

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Hyper-Process Modeling in Reconfigurable Manufacturing Systems with multivariate production: an automated learning approach

João Pedro Correia dos Reis



Doctoral Program in Informatics Engineering

Supervisor: Gil Gonçalves

April 4, 2019

Hyper-Process Modeling in Reconfigurable Manufacturing Systems with multivariate production: an automated learning approach

João Pedro Correia dos Reis

Doctoral Program in Informatics Engineering

April 4, 2019

Resumo

Atendendo à necessidade atual de produtos cada vez mais customizáveis com características específicas de acordo com a procura dos consumidores, as empresas da indústria transformadora necessitam aumentar a resposta e flexibilidade da sua produção. Para isso, e como forma de satisfazer a procura dos seus clientes, é necessário um entendimento mais detalhado sobre todos os processos existentes no chão de fábrica, juntamente com a capacidade de reconfigurar todos os equipamentos envolvidos. Neste contexto, o conceito de Sistema de Manufatura Reconfigurável (SMR) foi introduzido para abordar a mudança de paradigma de produção em massa para customização em massa, que exige uma mudança frequente na produção entre múltiplos produtos diferentes a custo mínimo.

Contudo, como forma de facilitar a mudança de produção entre múltiplas variantes de produtos, é necessário o conhecimento quer dos parâmetros possíveis de serem definidos e alterados no equipamento, quer do impacto que estes têm na qualidade final do produto, de acordo com o produto em questão. Neste sentido, normalmente um elevado número de experiências nos equipamentos testando diferentes parâmetros é necessário até que a qualidade final desejada seja atingida. Se estas experiências forem bem registadas e documentadas, algoritmos de aprendizagem automática para regressão podem ser utilizados para mapear esta relação entre parâmetros e qualidade final, onde esta qualidade normalmente é descrita por variáveis contínuas.

No entanto, existem duas restrições principais com a abordagem apresentada. A primeira restrição está associada à necessidade de um elevado número experiências para que se possa tirar o maior proveito dos algoritmos de aprendizagem computacional, e ajudar de facto os operadores de chão de fábrica onde cada experiência tem um custo muito elevado quer de pessoal, quer de material desperdiçado. A segunda restrição está relacionada com o uso e transferência dos modelos aprendidos para novos produtos. Se uma técnica de aprendizagem computacional for utilizada para modelar a relação entre parâmetros e qualidade para um produto A, um novo modelo é necessário para um produto diferente B. Posto isto, a principal hipótese apresentada na presente tese propõe que é possível o treino de modelos que mapeiem a relação entre parâmetros e qualidade final sobre novos produtos que não sejam conhecidos, onde não existe qualquer dado ou experiência realizada sobre o mesmo, e que apenas se baseia em dados adquiridos e modelos treinados anteriormente, de produtos passados. Isto significa que um novo produto pode ser integrado em chão de fábrica, sem que sejam feitas experiências sobre o mesmo, para determinar a relação parâmetro-qualidade, apenas usando informação existente de produtos anteriormente integrados. Em suma, este processo pode ser visto como a integração de novos produtos sem dados.

Baseado nisto, novas abordagens necessitam de ser exploradas e desenvolvidas para que se possa prestar a melhor assistência na integração de novos produtos sem dados. *Zero-Shot Learning* (ZSL) é um problema da aprendizagem computacional que tem como objetivo a interpolação entre múltiplas tarefas aprendidas e extrapolar a partir do conhecimento adquirido novas tarefas não conhecidas. Contrariamente ao processo tradicional de aprendizagem computacional, o propósito de ZSL é a reutilização de experiência passada e conhecimento para resolver um problema at-

ual. Normalmente os algoritmos de aprendizagem computacional focam-se na aprendizagem de tarefas isoladas que não podem ser diretamente utilizadas em tarefas diferentes. Nesse sentido, o propósito de ZSL no contexto da indústria transformadora é o de prestar assistência à integração de futuros produtos de interesse, como normalmente ocorre no paradigma de customização em massa, baseado em relações parâmetro-qualidade aprendidas sobre produtos integrados no passado.

Métodos recentemente apresentados na literatura abordam o problema apresentado de resolver futuros problemas de interesse, onde já um conjunto significativo de métodos foi proposto. Contudo, a grande maioria destes métodos tem como foco o problema de classificação de imagens, onde a correta classificação é feita para novas imagens, onde a classe correspondente dessa imagem não foi utilizada no processo de treino do algoritmo. À parte da classificação de imagens, apenas um pequeno conjunto de métodos genéricos foi proposto sendo aplicáveis tanto a problemas de classificação como regressão, onde os coeficientes dos modelos treinados são utilizados, como os pesos em rede neuronais ou os coeficientes de regressão polinomial. É baseado na aprendizagem da relação entre estes coeficientes que previsões de novos coeficientes é possível de ser feita, de acordo com novas condições propostas. Como estas abordagens são genéricas, existem algumas limitações que podem ser ultrapassadas. Ao invés de utilizar diretamente os coeficientes dos modelos treinados, dados extraídos desses modelos são utilizados na abordagem proposta. Neste caso, uma análise à forma desses dados é efetuada às variáveis contínuas (principalmente a qualidade final do produto) recolhidas dos processos industriais, como forma de criar um modelo estatístico dessas mesmas formas. Deste modo é possível gerar novas formas que representam a qualidade expectável de um novo produto de interesse. Assim, a relação entre parâmetros e qualidade para novos produtos não conhecidos pode ser prevista pelo modelo.

A abordagem proposta na presente tese foi testada em três cenários diferentes. Primeiro, foi feita uma comparação direta entre a abordagem proposta e um dos métodos genéricos. Para isso, foi definido um cenário baseado num conjunto de curvas geradas pela distribuição Beta onde o principal objetivo foi o de prever a forma de uma curva usando apenas os parâmetros α e β da distribuição e a modelação das restantes curvas. O segundo cenário aplica o método proposto a um conjunto de dados gerados através do simulador Abacus, tendo como contexto o processo de embutimento. Por último, um conjunto de dados extraídos de um cenário real de soldadura a laser foi utilizado. Todos os cenários utilizados têm como finalidade testar diferentes aspetos do método desenvolvido. Os resultados demonstram que a área de ZSL, e em concreto os métodos que exploram as propriedades de regressão como o proposto na presente tese, podem ser utilizados em contextos industriais. Posto isto, esta área pode ser particularmente útil para SMRs onde uma elevada resposta é necessária para ser possível produzir um maior número de variantes de produtos de forma eficaz e eficiente.

Abstract

Considering the nowadays demand for high customized products with specific features to attend customer needs, manufacturing companies are required to be highly responsive and flexible in their production. In order to meet all customer specifications, a deep and thorough understanding of each process used at the shop-floor is required, together with the capability of those machines to be quickly reconfigured. In this context, the concept of Reconfigurable Manufacturing Systems (RMS) was introduced to address the paradigm shift from mass production to mass customization, which demands for frequent changes in the production of multiple different products with minimized costs.

However, in order to easily change production outside an envelope of known product variants, the knowledge about how machine parameters impact the final product quality, according to the product in hands, is necessary. Hence, normally a large set of experiments using different machine parameters is performed until the desired quality is achieved. If these experiments are well documented and registered, machine learning techniques might be used to model the relation between parameters and quality in a regression setting, where the product quality is normally assessed by continuous variables.

Nevertheless, there are two main constraints with such an approach. The first is associated with the need for a significant number of experiments for machine learning techniques to be highly effective, where experiments are costly mainly due to material and personnel resources. The second constraint is related with the transferability of trained models to new product variants. If a machine learning technique is used to model parameters into quality for product A, a new model is required to be trained for a different product B. Hence, the main hypothesis of the present thesis proposes that is possible to train machine learning models that map the relation between parameters and quality for new unseen products where no data exists, based on previous acquired data and existing trained models from different products. This means that a new product can be integrated at the shop-floor without any data from its parameter-quality relation and solely by using existing information from previous processes. This can be viewed as a zero-data integration of new products.

Based on this, new approaches are required to be explored and developed that can assist on this idea of zero-data integration. Zero-Shot Learning (ZSL) is a machine learning problem that aims at interpolating among different learned tasks and extrapolating from the acquired knowledge to new unseen tasks. Contrary to the traditional machine learning setting, the main purpose of ZSL is to reuse past experience and knowledge to solve the current problem. Normally machine learning algorithms focus on isolated tasks that cannot be directly used for other tasks. Hence, the goal of ZSL in the manufacturing context is to assist on the integration of a future product of interest, as normally occurs in the mass customization paradigm, based on already learned parameter-quality relations of past integrated products.

Current state of the art techniques address such a problem of solving future problems of interest where already a significant amount of techniques have been proposed. However, most of them

are focused on image classification problems, where a label should be predicted from a previous unseen image, based on the assumption that the correct class of this image is not used during the training process. Apart from image classification, only a couple of generic methods were proposed so far that are applicable to both classification and regression settings that make use of model coefficients, like weights in Artificial Neural Networks or coefficients in Polynomial regression. It is based on learning the relation among these weights / coefficients that new ones can be predicted according to new provided conditions. Since these are general approaches, some limitations exist that can be overcome. Instead of directly working with model coefficients, data drawn from trained models is used in the proposed approach. In this case, a shape analyses of data collected from the industrial processes was performed in order to create a statistical shape model. From this shape model it is possible to generate new shapes that represent the predicted and expected quality for a new product of interest. This way, the relation between parameters and quality can be predicted for new unseen products.

The proposed approach was tested in three different scenarios. The first is where the technique is compared directly with an existing general framework that is applicable to both classification and regression problems. For that, a scenario was built based on the diverse set of curves generated by a Beta Distribution where the main goal is to predict curve shapes based on α and β parameters and previous fitted curves. The second one applies the proposed approach in a simulation dataset collected from the Abacus simulator of multiple deep drawing processes. Ultimately, the proposed approach was tested in a real dataset from a laser seam welding process. All these test scenarios explore and validate different properties of the proposed method. The results show that ZSL methods, and specifically those that leverage the properties of regression, such as the one proposed in this thesis, can be applied and used in the context of manufacturing. This is particularly useful for RMS where a high responsiveness is required to yield an increasing number of product variants.

Acknowledgements

First, I would like to thank Professor Dr. Norbert Link for all the fruitful feedback throughout my thesis and for the excellent collaboration in European projects, even outside the scope of my doctoral programme. Norbert is always an important piece in European projects with all the great ideas and positive thinking. We are missing now another European project. Not less important, I would also like to thank the ISRG team (Tarek, Johannes, Samuel, Jurgen and Alireza) for their support during my stay in Karlsruhe and all great machine learning discussions.

Obviously, I would like to thank and give my words of appreciation to my supervisor Professor Gil Gonçalves for all the support and encouragement during all the stages of my thesis. It was because of Professor Gil that I had my first steps in European projects and the whole world of Industry 4.0 from the point I didn't know what a "mote" was and I needed to look for it and buy a set of new equipment. It's very comforting to look back to the 6 years of research that we made, and all the hard and soft skills that I learned (yes, soft skills are very important for research and I learned that in the hard way), all the conferences, (social) meetings, summer schools and lab equipment to experiment like an old school scientist. If someone wants to do research, it should be Professor Gil's way! Ultimately, I should state that the skill I most appreciate in Professor Gil is the way he has to clear someone's mind of problems and simply give you a solution that you did not think of during the last month of research. It's very frustrating to think "How did I not think about this earlier? How did I miss this ... ". Sometimes all you need is your supervisor's words. Thank you for all Professor Gil.

To my parents, my brother Diogo, my niece Leonor and friends, that listened hours of my complaints of being a PhD student and helped me in ways they cannot even imagine. Specially Hugo "S", André "little", Tracy, Rui "Baka" (é este ano que o cavalito vai funcionar!), Diogo "Eskilo" (for all the creative thinking), Joana, João "Pinetree", Sara, Jorge, Ni, Pedro and above all Sandra for being always there, the most patient person in the world that always have the correct words for you, makes the correct questions that help you rethink your problems, and most importantly, always believed in me during tough times. To all the people from SAS, mainly Sara, Paulo, Catarina and Ana Cristina that greatly helped and supported me during my thesis in ways they don't know. To Professor Fernando Lobo Pereira for all the opportunities in SYSTEC and ISR-Porto, and last but not the least, to all SYSTEC-FoF team, mainly the ones that followed my work closely such as Rui (a seguir és tu a acabar, vamos lá!), Susana (que já me aturou muito em viagens de projetos europeus, e também não te livras do PhD), Luis, Liliana e Eliseu. To all the people that indirectly influenced or inspired me by speaking, writing or any other form.

João Reis

*"Ah, Putrefaction."**

Sherlock Holmes

*This is a quote from the Sherlock Holmes movie (2009) when Sherlock Holmes (SH) himself enters a room looking for clues about a crime. This reference represents his joy and happiness of sniffing the air and smelling putrefaction. By feeling this scent, he knew that something unusual happened and this could represent a step forward to his investigation. The feeling of happiness is not directly related with the putrefaction smell itself but with his true desire to solve crimes, which is what SH truly loves to do. Extrapolating this scene to real life, the feeling expressed by *"Ah, Putrefaction!"* is a metaphor for what everyone should feel about their work as a researcher (or something else), with an implicit joy for discovery and understanding, regardless of any condition and situation, like the one of putrefaction scent that is at least disturbing, but not for SH.

Contents

1	Introduction	1
1.1	Context	3
1.2	Motivation and Objective	3
1.3	Problem Definition	4
1.4	Thesis Structure	5
2	Background	7
2.1	Reconfigurable Manufacturing Systems	7
2.1.1	Characteristics	10
2.2	Machine Learning	11
2.2.1	Overfitting, Underfitting and Model Complexity	13
2.2.2	Model Selection & Hyperparameter Tuning	17
2.3	Hyper-Model	19
2.4	Statistical Shape Modeling	20
3	Literature Review	25
3.1	Transfer Learning	25
3.2	Introduction to Zero-Shot Learning	29
3.3	Related work in Zero-Shot Learning for Image Processing	30
3.3.1	Two-Stage approaches	32
3.3.2	Compatibility Functions approaches	38
3.4	Related work in Zero-Shot Learning for others applications	43
3.5	Discussion	46
4	Hyper-Process Modeling	53
4.1	Problem Definition	53
4.2	Proposed Approach	56
4.3	The Beta Distribution case	58
4.4	Practical Relevance	65
5	Process Assessment	69
5.1	Experimental Setup	69
5.2	Hyper-Process Model using Deep Drawing process	71
5.2.1	Scenario Description	72
5.2.2	Process Modeling	76
5.2.3	Process Model Generation	78
5.2.4	Results	85
5.2.5	Discussion	99

5.3	Laser Seam Welding Process	102
5.3.1	Scenario Description	102
5.3.2	Process Modeling	104
5.3.3	Process Model Generation	110
5.3.4	Discussion	116
6	Discussion	119
7	Conclusions	127
7.1	Contributions	129
7.2	Publications	129
7.3	Research Projects	130
7.4	Future Work	132
	Appendices	135
A	Deep Drawing - Source Process Models	137
B	Deep Drawing - Zero-Shot Learning Results	147
	References	165

List of Figures

2.1	Capacity vs functionality comparison between DMS, FMS and RMS Koren et al. (1999)	8
2.2	Diagram depicting the three main areas of machine learning (Supervised Learning, Unsupervised Learning and Reinforcement Learning) and examples of application scenarios	12
2.3	Relation between Capacity / Complexity and Error (LeCun et al., 2015)	15
2.4	Bulls-eye comparison of bias vs variance impact on Model performance / error	16
2.5	Model complexity and Error relation with variance and bias	17
2.6	10-fold cross validation example.	18
3.1	Summary of all three different classes that compose Transfer Learning. Pan and Yang (2010)	27
3.2	Publications for the ZSL area	49
3.3	Citations for the ZSL area	50
4.1	Comparison between ZSL for image classification and regression. a) Case where a latent representation for both images and classes is used, and a compatibility between these is learned (Akata et al. (2015)). b) Case where multiple models are used to learn a hyper-model that maps model coefficients θ into task descriptions $\mathcal{C}$. Upon new task descriptions c' , new model coefficients θ' can be estimated and a new model is created (Pollak and Link (2016)). This representation is also applicable in the model space view approach from Larochelle et al. (2008).	55
4.2	4 different types of functions generated by the Beta Distribution. In the top row, an Hyperbolic function is depicted in the left handed side with α and β values of [0.5,0.5] together with a Linear function in the right handed side with α and β values of [1,1]. In the mid row, two Exponential functions are shown with α and β values of [0.5,1] and [5,0.5]. In the bottom row, two Gaussian functions were generated using the α and β values of [5,10] and [15,5].	60
4.3	4 different graphs are shown, where the left sided graphs are the fits for Polynomial degree 5 and the right sided graphs are the curve fits for Exponential and Gaussian functions.	61
4.4	Graph plots of two test curves, where the dashed blue line is the ground truth from Beta Distribution and the orange line is the result of the ZSL approach applied. In the left column are the results for Beta Distribution with $\alpha = 4$ and $\beta = 6$, and in the right column the results for $\alpha = 12$ and $\beta = 4$. The first row corresponds to the HM approach with model degree of 3 and hyper-model degree of 3, to mid row refers also to HM approach but with model degree 6 and same hyper-model degree, and the bottom row is the HPM with 4 components and hyper-model degree of 4.	64

5.1	Idealized training and generalization error curves. The vertical axis represents the error and the horizontal axis the number of epochs in an ANN. (Prechelt, 1998) .	71
5.2	Exemplification of a deep drawing process in 3 main steps: 1) First position of the piece; 2) Punch force to form the flat metal sheet; 3) Final positioning and end of the process (Library of Manufacturing)	72
5.3	Abaqus 2D plot interface. On the left-hand side, the initial positioning of the steel sheet, and on the right-hand the final drawing of the steel sheet.	73
5.4	3D Scatter plot for the process parameters BHF and F, and the process quality maximum stress for the following process conditions: Metal sheet thickness 1.5, initial stress of 100 and saturation of 130 (above) and metal sheet thickness 1.5, initial stress of 175 and saturation of 355 (below).	76
5.5	3D Scatter plot for the process parameters BHF and F, and the process quality distance for the following process conditions: Metal sheet thickness 1.5, initial stress of 100 and saturation of 130 (above) and metal sheet thickness 1.5, initial stress of 175 and saturation of 355 (below).	77
5.6	3D Scatter plot for the Process Parameters Blank Holder Force and Friction, and the Process Quality Maximum Stress (above) and Distance (below) for the following Process Conditions: Metal Sheet Thickness 1.5, Initial Stress of 100 and Saturation of 130.	79
5.7	Graphical display of number of components and the average MSE for all the processes per number of components.	82
5.8	3D scatter plot for 3 deformable parameter regarding all 18 shapes from the deep drawing processes.	83
5.9	Mean values of MSE for all the processes per number of source models using LIN, POL2 and POL3 techniques. On the left, the Mean MSE for Euclidean distance (EUC) and on the right for the Euclidean distance with normalized process conditions (NORMEUC).	88
5.10	MSE values for all the 17 processes with an increasing number of source models. On the left, 3 graphs are displayed with LIN, POL2 and POL3 techniques from top to bottom for Euclidean distance (EUC). On the right, the same techniques are used for Euclidean distance with normalized process conditions (NORMEUC). .	89
5.11	Scatter Plot for the deformable parameters using NORMEUC strategy. The blue points are the source models, the green points are all the predicted values in the range of process conditions (Thickness = [1.5, 2], Initial Stress = [100, 250], Saturation = [130, 500]) and orange point is the predicted value for the target process conditions (process 15) that will be used to generate the corresponding shape. . .	91
5.12	Scatter Plot for the Deformable Parmaters in the SSM using EUC strategy. The blue points are the source models, the green points are all the predicted values in the range of process conditions (Thickness = [1.5, 2], Initial Stress = [100, 250], Saturation = [130, 500]) and orange point is the predicted value for the target process conditions (process 15) that will be used to generate the corresponding shape.	92
5.13	6 Scatter Plots of maximum stress and distance using the for degreee 2 Polynomial with NORMEUC strategy for shape generation of process 15. On the left there are the maximum stress plots with 4, 5 and 9 source models from top to bottom. On the right, the same organization is used but for process quality distance. The blue points represent the group truth of maximum stress and distance, and the orange points represent the generated shapes from HPM.	93

5.14	2 Scatter Plots of Maximum Stress and Distance using the for Degreee 2 Polyno- mial with EUC strategy for shape generation of process 15. On the left there are the Maximum Stress plots with 5 source models from top to bottom. On the right, the same organization is used but for process quality Distance. The blue points represent the group truth of Maximum Stress and Distance, and the orange points represent the generated shapes from the Hyper-Process Model.	94
5.15	Plot of two graphs presenting the parameter space for LASSO (left) and Ridge (right) regression. The blue areas represent the constraints in the optimization function imposed by the regularizers and the red lines are the contours for the least squares error function (Friedman et al., 2009).	96
5.16	Plots of the EUC (left) and NORMEUC (right) strategies for LASSO 2 where multiple values for λ were tested in order to minimize the MSE.	96
5.17	Plots of the EUC (left) and NORMEUC (right) strategies for LASSO 3 where multiple values for λ were tested in order to minimize the MSE.	97
5.18	Plots of the EUC (left) and NORMEUC (right) strategies for Ridge 2 where mul- tiple values for λ were tested in order to minimize the MSE.	97
5.19	Plots of the EUC (left) and NORMEUC (right) strategies for Ridge 3 where mul- tiple values for λ were tested in order to minimize the MSE.	98
5.20	Plot of all techniques tested for each of the strategies used to measure distance among process models: EUC on the left and NORMEUC on right. All the penal- ization λ used for LASSO2, LASSO3, Ridge2 and Ridge3 minimized the overall MSE.	99
5.21	Deformable parameter space for both POL3 (left), Ridge 3 (right) and LASSO 3 (bottom) using NORMEUC strategy.	100
5.22	Representation of weld width and penetration depth of two metal sheets by laser seam welding of AWL-Techniek.	103
5.23	MSE Prediction histograms for the three ANNs trained for each of the processes.	107
5.24	Histogram depicting the number of samples in relation to the MSE for the target and optimized quality.	109
5.25	Histogram depicting the number of samples in relation to the MSE for the desired and optimized parameters.	109
5.26	Optimization function for the hyper-model in the laser seam welding case for all the processes.	113
5.27	Scatter Matrix Plot of the real experimental data for process HC420LA - HC380LA	114
5.28	Scatter Matrix Plot for the generated process model for process HC420LA - HC380LA	115
6.1	Exploration vs exploitation: On the left, a solution that needs to experiment all possible solutions in order to find the global optimal (green dashed circle). On the right, by using the HPM for model generations and an iterative process, the optimal, or near-optimal, predictive model can be found, avoiding to perform all possible experiments.	125
A.1	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=130. For the process models, blue dots are the ground truth and in orange are the predicted values.	138
A.2	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and in orange are the predicted values.	138

A.3	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and in orange are the predicted values.	139
A.4	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and in orange are the predicted values.	139
A.5	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and in orange are the predicted values.	140
A.6	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and in orange are the predicted values.	140
A.7	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and in orange are the predicted values.	141
A.8	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and in orange are the predicted values.	141
A.9	Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and in orange are the predicted values.	142
A.10	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and in orange are the predicted values.	142
A.11	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and in orange are the predicted values.	143
A.12	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and in orange are the predicted values.	143
A.13	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and in orange are the predicted values.	144
A.14	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and in orange are the predicted values.	144
A.15	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and in orange are the predicted values.	145
A.16	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and in orange are the predicted values.	145
A.17	Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and in orange are the predicted values.	146
B.1	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=130. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	148

B.2	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	149
B.3	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	150
B.4	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	151
B.5	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	152
B.6	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	153
B.7	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	154
B.8	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	155
B.9	Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	156
B.10	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	157
B.11	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	158
B.12	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	159
B.13	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	160
B.14	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	161
B.15	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	162
B.16	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	163
B.17	Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and orange the generated shape from HPM.	164

List of Tables

2.1	Characteristics Comparison between DMS, FMS and RMS Koren and Shpitalni (2010)	8
3.1	Review table presenting the main methods for ZSL in Literature and their context in terms of Learning process and kind of Problem to address.	48
4.1	HM approach: Mean MSE, Standard Deviation MSE and Hyper-Model R^2 for Number of Components from PCA decomposition of 3, 4, 5 and 6, and the degrees of 3, 4, 5 and 6 for Polynomial Regression of the Hyper-Model.	62
4.2	HPM approach: Mean MSE, Standard Deviation MSE and Hyper-Model R^2 for Number of Components from PCA decomposition of 3, 4, 5 and 6, and the degrees of 3, 4, 5 and 6 for Polynomial Regression of the Hyper-Model.	63
5.1	Process Conditions for the Deep-Drawing process simulation in Abaqus	74
5.2	Mean Squared Error for all the 18 Deep Drawing processes using ANN.	78
5.3	Assessment of the number of components used in PCA for SSM to generate back the initial shape.	81
5.4	Process conditions for the deep-drawing process simulation in Abaqus.	84
5.5	Ranking of deep drawing processes by shortest distance from process with T=1.5, IS=100 and S=130, using 1) Euclidean distance and 2) Euclidean distance with normalized process conditions.	87
5.6	Process Conditions, Process Parameters, Process Quality and Number of Experiments	104
5.7	ANN Topology assessment in order to find the most suitable model for each process.	105
5.8	Best and Worse predictions for the best ANNs trained for the three presented processes	106
5.9	Process Parameter Optimization for the best 3 solutions in the test set that minimizes the distance between optimized and real quality.	108
5.10	Process Parameter Optimization for the best 3 solutions in the test set that minimizes the distance between optimized and real parameterization.	110
5.11	Process parameters and process conditions used for hyper-model training, and optimized process parameters according to the defined process conditions.	112
5.12	Comparison between the generated model and the process model trained with real experimental data	114
5.13	Process Parameter Optimization in the generated process for the best 3 solutions in the whole dataset that minimizes the distance between optimized and real parameterization.	115

Abbreviations

RMS	Reconfigurable Manufacturing System
DMS	Dedicated Manufacturing System
CPS	Cyber-Physical System
CPPS	Cyber-Physical Production System
ANN	Artificial Neural Networks
MLP	Multi-Layer Perceptron
ZSL	Zero-Shot Learning
SVM	Support Vector Machine
HM	Hyper-Model
HPM	Hyper-Process Model
HPM2	Hyper-Process Model 2
DoE	Design of Experiment
SSM	Statistical Shape Model
PCA	Principal Component Analysis
MSE	Mean Squared Error
EUC	Euclidean distance
NORMEUC	Euclidean distance using normalized process conditions
LIN	Linear Regression
POL2	Degree 2 Polynomial regression
POL3	Degree 3 Polynomial regression
LASSO2	Degree 2 LASSO regression
LASSO3	Degree 3 LASSO regression
Ridge2	Degree 2 Ridge regression
Ridge3	Degree 3 Ridge regression

Chapter 1

Introduction

The increasing number of product variants as a result of mass production to mass customization paradigm shift (Wang et al., 2016), is leading to the necessity of knowing in detail the machine process dynamics in order to quickly change between products being yielded in a small-lot fashion, or introduce a new machine in the shop-floor. This happens mainly because manufacturing companies are getting closer and closer to the end-customer and allow for customized products composed of multiple options and combinations, leading to a high number of product variations to manufacture. This forces the manufacturing companies to be much more responsive to market needs as a way to increase its market share and create new competitive advantages. However, in order to achieve this level of competitiveness, smarter and innovative ways to explore equipment skills and reconfiguration are required. Given the machine operations heterogeneity and production cycle speed, there's a demand for new techniques that intelligently can operate machinery according to diverse and new product requirements, and rapidly react to these requirements.

Normally, the operation of a certain machine is guided by a set of process conditions, like the type of material used or material thickness, and the requirements defined by the customer for final product quality. In order to meet these requirements, the correct process parameters need to be chosen to yield the desired process quality subject to the defined process conditions. This way, a good understanding of how process parameters influence the process quality needs to be developed. For a better understanding, an example is given showing how this information can be used in a manufacturing process.

Imagine a laser head mounted in a robotic arm with the objective of welding two metal sheets by issuing radiation from the laser head to a local area where the materials need to be joined. This radiation creates a melting zone on the laser focus in both sheets that solidifies when the focus is moved. This produces a continuous welding seam when the beam is moved along the overlapping sheets at controlled speed. To this intent, the process parameters (what can be changed in the machine resulting in different types of welds) are described by 3 independent variables: Laser Power (P); Focal Distance (F) from the surface and Robotic Arm Velocity (V). The assessed process quality that is achieved with specific process parameters are the Weld Width (W) and Penetration Depth (D) of the welded area, representing the final quality that needs to be observed.

If the influence of process parameters on process quality is explicit in a dataset through multiple experiments, machine learning techniques can be used to model this relation, building up a process model that maps machine parameters into final quality. Hence, such process models allow to infer which are the most suitable process parameters according to certain required quality. Additionally, on top of this information, the process conditions define in which context the process model is valid. For example, if a process model is trained using the process parameters and quality of two metal sheets, both with a 2mm thickness, such a process model becomes obsolete if these thicknesses change. This happens because the same parameters used in the metal sheets with 2mm each will lead to different quality in a new pair of materials with different thicknesses. In this context, different thicknesses represent different product variants. As a consequence, if a new product variant is introduced in the manufacturing process a new process model needs to be built.

Normally, the exploration of these parameter-quality relation is made by performing a Design of Experiment (DoE) - Full Factorial Design or Fractional Factorial Design - to know how process parameters impact process quality. Basically, these techniques present a strategy to know which the best value combinations of parameters are to experiment in the machine, so the number of experiments can be minimized, and the significance of data collected maximized. From these experimental findings, normally a dataset is built, and machine learning techniques can be used to train a process model, which is an abstraction and a simplified version of the real-world dynamics.

However, due to the mass customization paradigm and the need to often integrate new products in production lines, approaches such as full factorial design are considered costly mainly because they require 1) technical personnel to perform these experiments and 2) waste of raw materials due to destruction tests to assess process quality. From the perspective that mass production is shifting to mass customization, the need for quicker and rapid ways to introduce new machines and product variants to broaden the spectrum of available products to market, needs to be explored. This way, there's a huge potential yet to explore in such manufacturing systems.

Therefore, the current thesis proposes a new algorithm that increases the flexibility and re-configurability of a manufacturing system by intelligently leveraging already acquired shop-floor information. Most machines are hand-cuffed to operate under a small set of pre-defined configurations that need to pass through large periods of machine experimentation, where human resources are required, and a large amount of raw material is used for machine calibration. Thus, the main idea of the present thesis is to use information already available from a set of processes and use it to derive new knowledge about unseen processes yet to integrate that are required to be introduced in the production line. The key benefit of such approach is to reduce, or even completely remove, the need for large periods of machine experimentation under new unseen conditions (e.g. new types of material or new material thicknesses). If a machine yields a set of product variants, theoretically, adding a new process would not be hard task to perform. However, in practice is not so straight forward due to the high complexity of machines nowadays, where relating a high number of process parameters and its consequent impact is no ordinary job. This way, by exploring the latest machine learning concepts and techniques, it is foreseen that the relation between different existing processes can be learned efficiently and effectively, and therefore broaden the

amount of processes capable to be operationalized, and ultimately fully explore the capabilities of an equipment.

1.1 Context

The present thesis proposes a new algorithm that suits the context of Reconfigurable Manufacturing System (RMS), first introduced by Koren et al. (1999), where software platforms can be developed to reconfigure the operation of shop-floor equipment according to production requirements that can change in smaller periods than in normal circumstances, like Dedicated Manufacturing Systems (DMS), increasing its responsiveness to sudden market changes. Additionally, this thesis is framed in the latest concepts of Cyber-Physical Systems (CPS) applied to production systems, mostly called Cyber-Physical Production Systems (CPPS). Although there isn't an universal definition, CPS (Hellinger and Seeger (2011); Shi et al. (2011)) refer to the integration of computation and physical processes, which combine cyber aspects of computing and communication with the systems dynamics in the real world. These complex systems join concepts of robotics, sensor networks, intelligent mechanisms and control, by connecting several components in different layers, such as physical devices, sensors & actuators, control units and communication devices. The term CPPS characterizes CPS that are applicable in the industrial domain. Moreover, with the introduction of the Internet of Things (IoT) in the manufacturing environment, CPPS have a huge potential in terms of new business opportunities, mostly known as the 4th Industrial Revolution.

For this context, state of the art machine learning techniques will be explored and proposed to address some of the challenges of RMS. Taking advantage of data acquisition and processing from CPPS, a system can be developed to assist the integration of new product variants in the production line. With the benefits of digitalization in manufacturing systems, it is possible to record data from multiple machines and reconfigure / parameterize them according to new defined requirements from customers. Hence, these systems are suitable to apply machine learning algorithms in order to explore new relations among shop-floor equipment, high amounts of data acquired in real time and increase the knowledge about the manufacturing processes.

1.2 Motivation and Objective

By increasing the flexibility and reconfigurability of manufacturing systems it is expected to enable the customized production and the manufacture of small-lot products efficiently and profitably (Wang et al. (2016)). Such a belief led to a change in the manufacturing paradigm from mass production to mass customization, where it is stated that even one-off item can be produced profitably compared with the current approaches used in industry. As reported by Zhang et al. (2017), the impact of mass customization is mainly reflected in performance outcomes, like operational performance in terms of cost, quality, delivery, flexibility and innovation, as well as the added value to customers and their satisfaction.

Hence, the main motivation for the present thesis is to explore new algorithms that would enable mass customization. Concretely, the ease of integrating new products variants in production lines. Nowadays, most of the ramp-up phases when a new product needs to be manufactured require high times of experimentation and are led by a trial and error strategy. With information about previous product integrations, machine learning algorithms can be explored to reduce the number of experiments by assisting the strategy of machine calibration. This way, costs can be minimized due to the reduction of materials used mainly in destructive tests and human resources allocated to the calibration phase. Additionally, such calibration phases are not only applied when a production line needs to manufacture a new product, but also after equipment maintenance. Since some components might be replaced from the equipment, different machine responses are expected when comparing with its prior state. This way, the calibration phase consumes a high amount of resources, and an improvement in such process would result in high cost reduction.

The main objective of the present thesis is to propose and explore an algorithm framed in the context of Cyber-Physical Production System (CPPS), where existing information of the equipment dynamics can be used to speed-up the ramp-up process in a Reconfigurable Manufacturing System. Knowing the increasing importance of flexibility and reconfiguration at the shop-floor due to mass customization, the proposed algorithm aims at using data and available process models in order to guide the calibration process of a certain machine, reducing the periods of experimentation and wasted material.

1.3 Problem Definition

The capability of solving problems where no person has experience by relating previous learned tasks and yet, capable to quickly learn and solve the problem in hands, has been greatly explored in the Machine Learning community in the past decade. This concrete problem is denominated as Zero-Shot Learning (ZSL), framed into the broader area of Transfer Learning, and is defined by correctly solving a task where no training data is available, based on previous acquired knowledge from different, but related tasks.

So far, this kind of problem has mostly drawn all the attention from computer vision community due to its clear benefits for the field and direct application in innovative technologies. Nonetheless, small attention has been paid to regression problems, where instead of predicting a label from a specific class, the prediction is in form of continuous variables. To this point, no works have been proposed for regression problems, where only a couple of general frameworks that can be applied to both classification and regression were developed, but no algorithms really take advantage on the regression setting to achieve higher performances. Hence, it is the focus of the present work to propose a novel algorithm called Hyper-Process Model (HPM) to address ZSL for regression problems. Contrary to the classification setting, the main idea for HPM is not to correctly predict a value for a certain provided input, but rather to estimate a function to be used in a new task, based on a meta-description. This novel approach lies in performing a shape analysis to the existing data from previous learned tasks and correlates them with task descriptions. From

this perspective, the main idea is to predict the new shape of data based on a new meta-description, so a function could be learned from this generated shape.

In the context of RMS, the main idea is to use process conditions as meta-descriptors and based on existing process models that map machine parameters into process quality, generate new process models according to new unseen process conditions. This is achieved by exploring the different responses that different process parameters have for different process conditions, so new responses can be predicted. This way, it is possible to estimate a new relation between parameters and process quality to assist the operator when integrating a new product in the shop-floor.

1.4 Thesis Structure

This thesis is organized in 6 more sections. In Section 2, an introduction to Reconfigurable Manufacturing Systems is presented, together with some initial concepts of machine learning, more specifically in Supervised Learning problems. Additionally, an introduction to the concept of Hyper-Model and Statistical Shape Model is made. In Section 3 the area of Transfer Learning is introduced and a literature review is made for Zero-Shot Learning. Additionally, a final discussion about the state-of-the-art works is made and some final remarks are drawn. Moreover, Section 4 makes a first definition of the ZSL problem for regression (Section 4.1) and presents the proposed Hyper-Process Model (HPM) algorithm (Section 4.2). Then, a simple example using the Beta Distribution to demonstrate the potential of HPM is made (Section 4.3). Finally, some practical aspects and benefits of HPM are explored in Section 4.4. In Section 5 the main experiments to explore the advantages and limitations of HPM are presented. More specifically, Section 5.1 defines some ground rules for both experiments performed, being one related with the industrial process of deep drawing (Section 5.2) and the other with a laser seam welding (Section 5.3). Afterwards, Section 6 presents a discussion around the proposed algorithm and existing approaches for ZSL, together with the possible application to other practical areas and different types of problems. Finally, Sections 7 draw some conclusions about the whole work and present some new directions of research related with the problem of ZSL for regression, correspondingly. At the end of the present thesis, there are two different appendixes (A and B) where plots about the training process of models and final results of the proposed approach HPM are depicted.

Chapter 2

Background

2.1 Reconfigurable Manufacturing Systems

The old-fashion production paradigm was mostly handcuffed with mass production of a single product part without any product or family variant and with “fixed” or “rigid” system, not being easy to perform any changes in machines and production lines. This fact occurs because the corresponding systems were not designed to change the cycle times and be adaptive according to demand variation, market fluctuations and system failures. Otherwise, they operate on a single mode with fixed throughput and fixed product manufacture, being this production paradigm described by high production rates and few manufacture of product parts, meaning that customer demands should be higher than produced parts (full capacity operation), and therefore, the cost per produced part is very low. The type of manufacturing system that better describes this type of production is called *Dedicated Manufacturing System* (DMS).

Based on this, a new manufacturing concept was created to tackle some of the DMS limitations, aiming to increase the flexibility in producing product varieties within an envelope of options called *Flexible Manufacturing System* (FMS). This concept makes mostly use of the single-tool operation and general-purpose machine called *Computer Numerically Controlled* (CNC) to achieve the production of multiple products. The main objective of this approach is to easily reconfigure the system within a constrained set of possibilities, and therefore quickly respond to any production requirement changes. This reconfigurability is achieved by adapting the CNC machines to specific tasks that need to be executed at the shop-floor, and not building machines from scratch for that particular purpose. However, this leads to a much lower throughput of the system when compared with DMSs. Additionally, these machines are expensive, turning initial investment considerably higher. As a consequence of that, the cost per part becomes much higher, revealing not to be cost-effective, and consequently, not competitive enough for harsh market conditions.

Therefore, the author Yoram Koren first introduced the concept of Reconfigurable Manufacturing System in 1999 with the main intention to quickly respond to sudden market changes and aiming to improve the manufacturing system’s responsiveness (Koren et al. (1999)). It has the main purpose of putting together the advantages of both DMS and FMS in one single concept,

	Dedicated	RMS / RMT	FMS / CNC
System Structure	Fixed	Changeable	Changeable
Machine Structure	Fixed	Changeable	Fixed
System Focus	Part	Part Family	Machine
Scalability	No	Yes	Yes
Flexibility	No	Customized (around a part family)	General
Simultaneously operating tools	Yes	Possible	No
Productivity	Very High	High	Low
Cost per part	Low (for a single part, when fully utilized)	Medium (Parts at variable demand)	Reasonable (Several parts simultaneously)

Table 2.1: Characteristics Comparison between DMS, FMS and RMS Koren and Shpitalni (2010)

in a cost-effective way. On one hand, the RMS concept aims to combine the high throughput of DMSs and the flexibility in producing a set of product parts from FMSs, composing a reconfigurable way to change from one envelope of product parts to another, to quickly respond and react to changes that might occur. To better illustrate the differences between these three systems, Table 2.1 presents several comparison metrics and puts side by side all the concepts characteristics.

As seen, the RMS concept is focused on part family giving more flexibility to manage product or market fluctuations and system failures, contrary to the DMS which is based on the part and the FMS based on the machine. Saying that DMS is based on the part means that the machines used are very specific and built for a constrained purpose, being rigid to produce a certain product part, without room for maneuvering the extension of one single product part to a full envelope. Since the production is specific, normally this represents a very high throughput of the system, but there's no capability for changing the product manufacturing when needed. On the other hand, FMS is focused on the machine due to the use of general-purpose CNC machines that needs to be adapted a priori to produce a certain type of product or product parts. This leads to a very low system throughput, but machines can be easily adapted during the time of operation to change between an envelope of product parts. Figure 2.1 presents a graphical representation of system capacity or system throughput and functionality to accommodate new products or product parts.

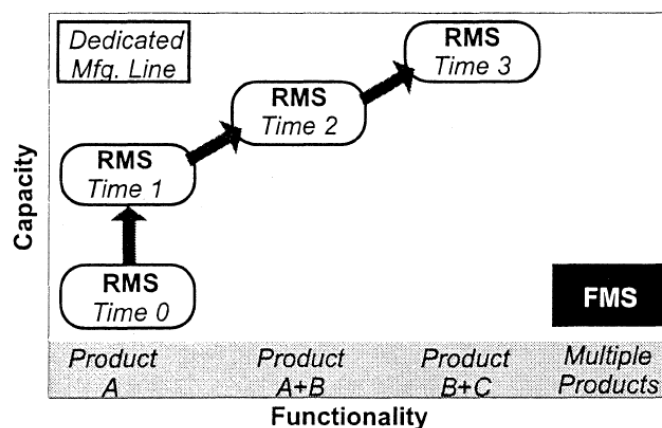


Figure 2.1: Capacity vs functionality comparison between DMS, FMS and RMS Koren et al. (1999)

As discussed, DMSs are capable of high throughput and capacity, but little capability to change functionalities according to new requirements. On the other hand, FMSs are capable of accommodating multiple products or product parts and easily change its configuration, but a lower capacity is achieved. The purpose of RMS is to maximize both capacity and functionality, being capable of evolving during the time through many generations, to reach its best configuration.

As a way to materialize the objectives of RMS, Koren and Shpitalni (2010) have developed two different device concepts to enable RMS to be more responsive and productive. One of them is the Reconfigurable Machine Tool (RMT) which is a machine with changeable and modular structure, ready to be adapted from one product envelope to another. The main purpose is to easily change its physical configuration to accommodate new features to fulfill different product requirements. Additionally, and as a complement to shop-floor equipment, the author created the Reconfigurable Inspection Machine (RIM) concept that aims to inspect the process and infer the quality defects in real time. In terms of manufacturing process and machine calibration during ramp-up, having such a system is of extreme importance since automated observations of the process can be done and machine parameterization can be automatically tested. Moreover, reliable datasets can be automatically generated and much more accurate models can be created, reducing the physical experimentation of the manufacturing process when calibrating the system.

However, despite all the benefits from such a system, which is based on decades of experience collected in different manufacturing paradigms as DMSs and FMSs, the RMS has some inevitable drawbacks, as any other system. As Koren and Shpitalni (2010) referred, *“If the system and its machine are not designed at the outset of reconfigurability, the reconfiguration process will prove lengthy and impractical”*. In other words, in an already deployed manufacturing system such as DMS or FMS, converting the whole system to become a RMS concept involves high costs, not only due to changes on machines and transportation systems (physical systems), but also due to software changes that are used for planning and machine parameter optimization (software systems). Up to now, process planners and process parameter optimizers for RMS are not mature enough and more research should be applied to create consistent enablers for that purpose. The author Elmaraghy (2007) already reported this issue saying that *“Modeling methods including classes, hierarchies, reconfiguration rules and knowledge, associations and relationships rules, functions, constraints, tables, and the like must be developed. Process plans representations that would facilitate their reconfiguration should be utilized”*.

Hence, one of the critical aspects to adapt existing DMS or FMS into RMS is the knowledge and information about the equipment. A more deep understanding about the existing equipment allows to explore more reconfiguration possibilities and also speed up the process of reconfiguration. By measuring the impact of machine parameters into the final quality of the product, a dataset can be created and machine learning algorithms used for system modeling. This way, the integration of new and already produced products can be made easily, and speed-up the ramp-up process shorting lead time. Moreover, new techniques should be developed to leverage the high amounts of data produced in factories from already sensorized machines or from the use of RIM, in order to relate these data and extrapolate relations among heterogeneous processes. This way,

a holistic view and a more broad perspective is grasped that could lead to better decision making about which products to manufacture, when to produce and how to produce them. The work of the present thesis proposes to explore these relations by using state of the art machine learning techniques.

In the next subsection, the main characteristics of the RMS concept will be formalized, enabling to better understand its underlying concepts and better frame the work of this thesis. Additionally, it allows to assess the impact of the proposed approaches and if the achieved results are pertinent in this context.

2.1.1 Characteristics

According with Koren and Shpitalni (2010), for a system to be designed based on both modular hardware and software that can be reliably and quickly integrated, it needs to be composed by six different characteristics. A system that contains these is considered to be highly reconfigurable, and are presented as follows:

- **Customization:** This characteristic is related with the possibility to customize the production considering a certain set of possibilities, mostly referred as a product or product part envelope;
- **Scalability:** The ability to scale the system's capacity by adding or removing machines to the production line to compensate for variations on product demand;
- **Convertibility:** This characteristic aims to easily convert the functionalities of machines and overall system to cope with new production requirements. The adaptation to new products is a key aspect that needs to be considered taking into account the mass customization paradigm that market is facing nowadays;
- **Modularity:** Hardware and software operational functionalities should be modular to easily accommodate new functionalities and be reconfigured at the system level;
- **Integrability:** Since RMS is a concept based on modular hardware and software which defines very well its interfaces, machines can easily integrate new modules;
- **Diagnosability:** The ability to assess the system's internal state turning possible an automatic detection of production defects. This allows to diagnose the root cause of a certain problem, and therefore, to apply the most suitable prescription or remedy solution.

To better comprehend the purpose of each characteristic, the author has divided them into two different groups, concerning the *Critical Reconfiguration* and *Rapid Reconfiguration* characteristics. Regarding the first group, it is composed by Customization, Scalability and Convertibility, which defines all the necessary flexibility to change, add and convert machines when new requirements need to be accommodated. The latter is composed by Modularity, Integrability and

Diagnosability and are capable to quickly respond to sudden system failures and market changes by easily integrating new machines with well-defined interfaces.

It is in the Critical Reconfiguration that this thesis have its most impact and contribution, mainly because it is essentially achieved with data-driven technology. By exploring the behavior of multiple machines and processes through well annotated datasets and real-time data, knowledge about the previous production can help understand how to integrate new products by a set of customer requirements. This approach is one of the key enablers to achieve both *Customization* and *Convertibility*. Additionally, the information about existing machine behavior allows to assess the capability of a manufacturing system to support new machines in current processes, and help in the integration of those by minimizing the calibration phase. This is mostly related with *Scalability* where the current system is scaled to add or remove machines according to specific product demands and market fluctuations.

For a complete overview about the RMS field and the latest trends towards the achievement of Industry 4.0, please refer to the review paper presented by Bortolini et al. (2018).

2.2 Machine Learning

According to Samuel (1959), machine learning is defined by "*the field of study that gives computers the ability to learn without being explicitly programmed*". This means that there are algorithms that are intended to build other algorithms from any kind of data available. From sociology to manufacturing, the purpose of this field is to build a surrogate model based on data available from a certain task in order to make predictions about the future. This surrogate model is a simpler version of any system dynamics (from real world, digital or both) that tries to capture its patterns and subtleties, composed by a set of rules that were learned directly from data.

The discipline of machine learning is often described as being arranged in three main groups: 1) Supervised Learning; 2) Unsupervised Learning; and 3) Reinforcement Learning. There are other important categories that could be discussed, but these are the main cornerstones that are often used to give a broad initial context about the area. Figure 2.2 presents a graphical representation of the three main groups together with some application areas. In the remaining of this section, a brief description about the categories will be made, along with a particular focus on understanding in more detail Supervised Learning due to its use in the present thesis.

Supervised Learning is the category that handles problems where outputs are known for specific inputs. The main goal is to find a function or set of functions that can translate input variables into a set of output variables so new predictions can be made for new inputs. This implies that both input and output variables are good representations and descriptors of the system that we want to model and these functions are a good generalization of the whole system. Hence, there are multiple algorithms that automatically learn the relationship among input and outputs. There is a large set of algorithms available for this learning process that are applicable for either regression or classification problems, or both. Some of the most used algorithms are Linear and Polynomial regression, Support Vector Machines (SVM), Artificial Neural Networks (ANN), Decision Trees

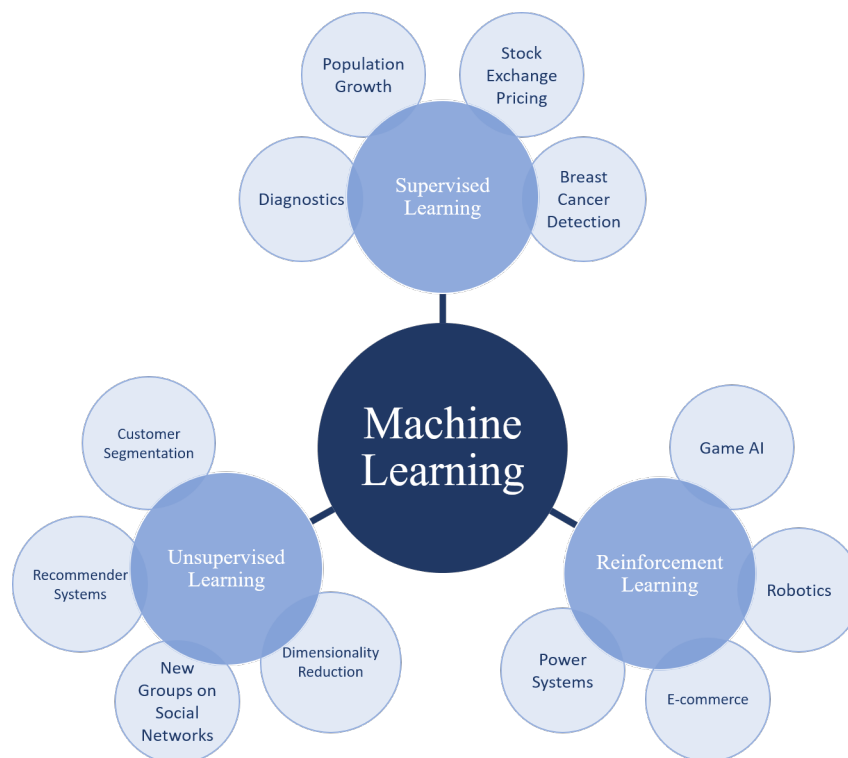


Figure 2.2: Diagram depicting the three main areas of machine learning (Supervised Learning, Unsupervised Learning and Reinforcement Learning) and examples of application scenarios

(DT) and Bayesian Networks (BN). For regression problems, some examples of application areas are the prediction of population growth, estimating life expectancy or prediction of stock exchange. In classification problems, some areas are related with diagnostics, image classification and fraud detection.

Contrary to Supervised Learning, Unsupervised Learning aims at finding clusters and groups of data by only dealing with input information. In this case no output information is available, so no input-output mapping is possible to be established. This way, relations between input data are assessed, normally by proximity, to form groups of data and recognize new patterns. To this process is called Clustering and is used to detect outliers (like deviant behaviors from common practice) and to assign a certain input into a found cluster of data. Some of the algorithms used are k-Nearest Neighbors (k-NN), DBSCAN and Spectral Clustering, and examples of application areas are recommender systems, customer segmentation and in social networks to recognize new communities. Additionally, in the category of Unsupervised Learning, there are also important algorithms that address the problem of dimensionality reduction. This particular problem aims at selecting a set of input variables from the whole input feature space. It is normally used to reduce the complexity of the problem and remove some important or redundant features that do not add a significant advantage when compared to the added complexity, and therefore avoiding the effects of Curse of Dimensionality (Bellman, 1957). Some algorithms used for this sub category are Principal Component Analysis (PCA), Generalized Discriminant Analysis (GDA) and

Autoencoders.

Reinforcement Learning is a machine learning category that aims at interacting with a system or environment in order to learn new rules and take actions based on a reward-based approach. The main idea is to iteratively learn how to fulfill a set of goals, based on observations and measurements (state of the system), and learn the most suitable actions to take that maximize a long term reward. This area of machine learning is different from Supervised Learning because 1) it exploits the environment in order to learn what to do according to the present state, 2) is a reward-based system that tells what to do and not to do, and 3) it evolves during time according to the taken actions and rewards obtained. Some of the algorithms used nowadays for Reinforcement Learning are Markov Decision Process, Q-Learning and Deep Q-Learning (making use of Deep Learning approaches) that are often used for game AI, robotics and speech recognition.

It is based on these three areas of machine learning that most of the Artificial Intelligence (AI) state of the art technologies are built. As far as data is available or a system with whom one can interact with to collect data, AI algorithms can be used to create technologies that assist and complement humans to perform their job more effectively and efficiently. In some areas like robotics, AI allows to develop systems that can replace the human in hard and demanding physical activities, minimizing the risk of injuries or even fatalities in borderline situations, such as rescue scenarios. Hence, it is important to develop new algorithms for the machine learning area so new systems could be applied to broaden applications and assist or replace humans in a more effective way.

Despite these three main areas of machine learning being the foundations of most of the greatest technologies that we assist operating and executing nowadays, we will focus now on the Supervised Learning category due to its importance for the present thesis. The proposed approach makes use of some of the algorithms for Supervised Learning and a better understanding is required at this stage for future reference. For that, a brief introduction about training and modeling in machine learning will be presented in the following section.

2.2.1 Overfitting, Underfitting and Model Complexity

As previously explained, Supervised Learning aims at mapping a certain input feature space X into an output feature space Y . The input feature space is normally represented by random variables where any value can be drawn by measuring a random phenomena. This way, these measures should map into a set of variables (often called dependent variables) that explain the phenomena that one is measuring. Based on this, a function $f : X \rightarrow Y$ can be estimated in order to map these measures into predictions based on available data collected from the system. Machine learning algorithms can then be applied to automatically learn a set of function parameters that minimize the difference between the function prediction and the true value to predict. In the supervised approach, both regression and classification problems are addressed. A regression problem is defined when the output feature space is characterized by p continuous variables $Y \in \mathbb{R}^p$, such as temperature, person weight or prices. Classification handles output feature spaces with discrete binary values $Y \in \{0, 1\}$ indicating whether or not that feature is true, like head or tails in a coin,

or a dog being present in a picture. In some cases, such as in probabilistic models, the output variables can assume values between 0 and 1, $Y \in [0, 1]$ representing the probability of that feature being true according to the input.

Some methods allow for a direct calculation of such parameters based on data, and others require a more iterative process. Most of the algorithms lie in the latter case, so frequently these parameters are found through an optimization problem where the distance between predicted and true values needs to be minimized. The main idea is to test multiple parameter values and by assessing the distance to the true value, these are updated in order to get closer to the desired values. A frequent approach for this parameter optimization is to use the Gradient Descent (GD) algorithm (Cauchy (1847)) that updates the parameter values in proportion to the negative gradient of a function. For example, in ANNs the GD algorithm is used in the well famous Backpropagation algorithm where the weights of a network are optimized according to the same philosophy, and was first introduced by Werbos (1982) and popularized afterwards by LeCun (1985), LeCun et al. (1988) and Rumelhart et al. (1985). Despite being a robust and efficient method, it is designed to find the local minima of a function. If a function F is convex, then the local minima is the global minima and the optimal solution can be found. Thus, the optimization algorithm for the iterative process of finding the suitable parameter values is of great importance. If this optimization algorithm performs poorly, the error of the model will be high and no suitable prediction can be made. Since the GD algorithm was designed to find the local minima, if the function is not convex, it would be difficult to converge towards the global optima. Hence, for non-convex problems other methods could be used like Stochastic Gradient Descent introduced by Robbins and Monro (1951), that have a high probability of finding the optimal solution.

From Linear Regression to ANN, the automatic learning and optimization of function parameters is of key importance to achieve a good generalization in any machine learning setting. We can define model generalization as an effective way of an algorithm to produce accurate predictions apart from the ones used for the training process. Hence, it is worthless to have a model that has a good performance on the data used to optimize the model parameters if then it performs poorly when tested with new data or in real world situations. A factor that greatly influences this generalization capability is the model complexity. Here model complexity can be referred as the number of parameters to be optimized in a certain model, e.g. number of weights and layers of an ANN or the polynomial degree. Model complexity will greatly influence the achieved performance. If a model is too simple for the problem in hands, there's the possibility that it will underfit the data and details about the system to model will not be grasped. Also, if the model is too complex, it will probably overfit the data where it tries too hard to predict exactly the true value and perform poorly when presented with new, independent data from the training process. Both underfitting and overfitting mean that the model poorly generalizes the system based on the available data.

One of the approaches used to control these are based on splitting the dataset into train and test set. Assuming the dataset is i.i.d. (identically and independently distributed), splitting it into train and test sets mean that both are i.i.d as well. Therefore, the machine learning algorithm should be trained using the training set and afterwards tested by predicting the values present on

the test set and calculating the error. This test error is normally called the generalization error and has U-shape regarding the model complexity, whereas the training error tends to decrease as the complexity increases. This creates a gap between the generalization and training error. The sweet spot for any model is when training error is small and when the gap between training and test error is also small. Based on this, if the model is not able to achieve a low training error, most probably the model is underfitting the data. If the gap between training and test error is large, probably the model is overfitting the data. Figure 2.3 illustrates the relations between these two types of errors, where the red line depicts the sweet spot for model complexity in order to avoid underfitting and overfitting.

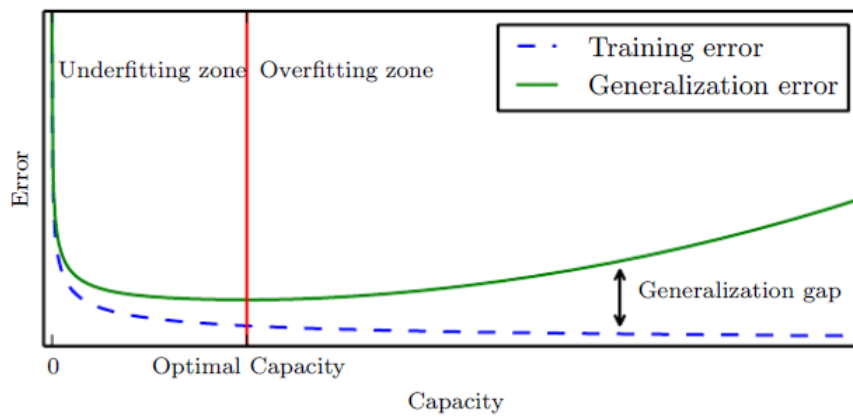


Figure 2.3: Relation between Capacity / Complexity and Error (LeCun et al., 2015)

For model complexity evaluation in terms of generalization there are two main measures used: bias and variance. Bias can be defined as the difference between the true value and the average of all predictions from the model measuring how far the model is from the ground truth. Variance can be defined as the variability between the average of all predictions and the model prediction measuring how sensitive the model is to new data. Based on Figure 2.4, imagine the center of the red bulls' eye region is the true mean value of our target random variable which we are trying to predict, and the red region indicates the variance spread of this variable. Every time we take a sample set of observations and predict the value of this variable we plot a blue dot. We predicted correctly if the blue dot falls inside the red region. In a nutshell, bias is the measure of how far off are the predicted blue dots from the true red region, while variance is how scattered are our predictions, meaning that if they are scattered, the variance is high, and if not, the variance is small (Faber, 1999).

When estimating a model $\hat{f}$ from a certain dataset $\mathcal{D}$, the expected squared prediction error for an instance x is defined as:

$$Err(x) = E[(y - \hat{f}(x))^2] \quad (2.1)$$

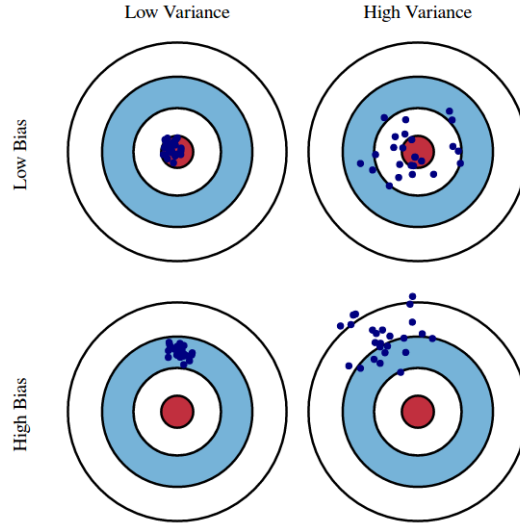


Figure 2.4: Bulls-eye comparison of bias vs variance impact on Model performance / error

This error equation can then be decomposed in the bias and variance components explained previously as:

$$Err(x) = (E[\hat{f}(x)] - y)^2 + E[\hat{f}(x) - E[\hat{f}(x)]] + \sigma_\varepsilon^2 \quad (2.2)$$

where the first term is the squared bias, the second is the variance and the last one is the irreducible error or noise that cannot be handled by any model, and comes from the error term ε in the prediction model $Y = f(X) + \varepsilon$, which is normally distributed with mean 0, such as $\varepsilon = \mathcal{N}(0, \sigma_\varepsilon)$. The only way to avoid such noise is only if $\sigma_\varepsilon^2 = 0$. As observed in Equation 2.2, in order to minimize the error both bias and variance need to be low. However, most of the times this is hard to achieve and a trade-off between the two components is required. As depicted in Figure 2.5, the bias tends to decrease as model complexity increases, because with higher model complexity it is possible to grasp more of the system details and have a better fit to the data. Contrary to this, if the model is too simple, it is not able to learn most of the details and will be rough approximation of the system, thus the high bias error. As for variance, it tends to increase as the complexity also increases because once all the details are learned, the model is not general and is very sensitive to new data, meaning that normally there are high variations of predictions between close input values. Both situations of high bias or high variance are not ideal and there is a sweet spot somewhere in between not-so-less bias and not-so-less variance which produces least prediction error in the long run.

Making a direct relation between bias-variance trade-off and Figure 2.3, when the training error is low and test error is high, it means that the variance is also high and the model is most probably overfitting the data. Since the training error is low, we can assume that the model grasps most of the system details, which implies a low bias, but due to high test error, the difference between the model and true test values is very high, leading to high variance. When training and

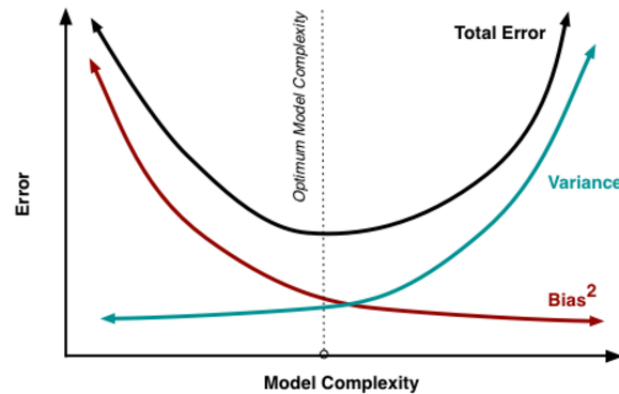


Fig. 6 Bias and variance contributing to total error.

Figure 2.5: Model complexity and Error relation with variance and bias

test errors are high but very close to each other, this implies an underfitting effect. From this case, we can assume a high bias because of the high error of training data and a low variance due to a narrow gap between train and test errors. The worst case scenario is when both train error and gap between train and test error are high. High train error implies a high bias, and a high gap between train and test errors implies a high variance. The best case scenario for a model generalization is when the train error is low, meaning low bias, and a small gap between the two errors, implying a low variance.

All the discussed factors of model complexity and bias-variance trade-off are important to understand in the context of supervised machine learning because the main goal of machine learning techniques is to serve as good generalizations of the system to be modeled. If these are not general enough, no good predictions can be made with new data and the whole purpose of training the model is lost. Based on this, it is important to use the best practices for training and avoid situations where during training the performance seems to be good, and once deployed for real world operation the performance is bad and is not usable. First understanding the relation between model complexity and the bias-variance trade-off is particularly useful to detect if the model is effective and will have a good performance once deployed.

2.2.2 Model Selection & Hyperparameter Tuning

In most machine learning applications there's not a method that fits all and can be applied to all possible scenarios. In that sense, normally a set of models is tested to assess which performs the best, together with a set of hyperparameters. Regarding the model assessment, one can test different methods like ANNs or SVMs, and additionally vary the hyperparameters of those methods. For instance, in ANNs the hyperparameter can be a learning rate, the number of hidden layers and neurons per layer. Therefore, there's a need to evaluate these models and compare which one is

more applicable according to the problem in hands. So, as previously described, the main idea is to avoid overfitting your model to the dataset by holding out a part of the data to test the performance of model. The model should be trained by using the training set, and then assessed in the test set, preferably with the same data distribution. One could think that for model selection and hyperparameter tuning, training a set of different models and versions of it with the train set and then choose the one with lowest test error would yield the best model to use, but it simply "fits" the test set. With such an approach, we are assuring that the best model is the one that performs the best for the test set, not the one that yields the best generalization. To avoid this test set fit, a third set should exist and is normally called the validation set. Similar to the test set, the validation set should follow the same data distribution as the test and train sets. Based on this, first the models are trained using the train set, and then assessed using the validation set. Then, the best model is chosen according to the validation set performance and a confirmation of the obtained performance is made using the test set. If the performances are similar, the chosen model is in fact a good generalization. If the performances are too different (normally the validation set is higher than test set) it means that the chosen model is fitting too much on the validation set and is not a good generalization.

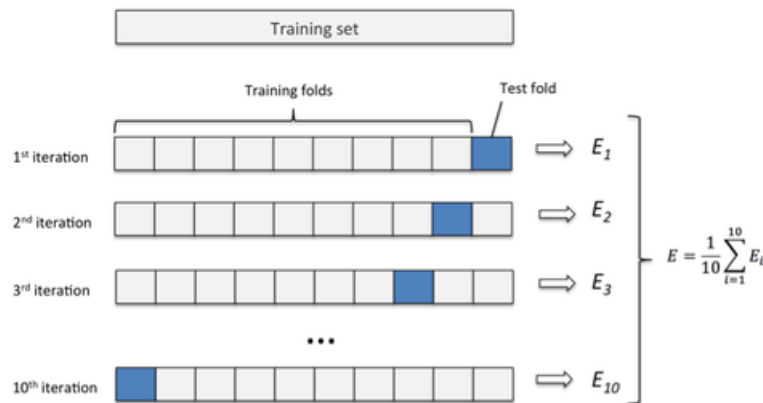


Figure 2.6: 10-fold cross validation example.

The described approach is mostly applicable in situations where the dataset has a significant amount of samples, and a representative amount exists in all train, validation and test set. However, there are application areas where data is very difficult to obtain, and a different approach needs to be taken. An approach first introduced and explored by Mosteller and Tukey (1977) called k -fold cross validation is one of the most effective techniques tackling the overfitting effects when a small amount of data exists. The method divides the dataset into k equally sized sets (normally $k = 5$ or 10), and iteratively selects one of the sets not chosen previously, trains the model with $k-1$ sets and tests its performance in the chosen set (evaluation). The process is repeated k times, until all the sets were chosen for testing. In the end of all iterations, the mean value of all performed evaluations is calculated. Is this value that should be used to compare different models or hyperparameters, where the model that presents the lowest error should be the used because it

ensures the best generalization capability out of all the tested models. Figure 2.6 presents a graphical representation of a 10-fold cross validation example, where there are 10 iterations of model evaluation using the train / test set strategy explained, and at the end of all iterations, the average of the evaluations is performed and the final model performance calculated.

The intuition of using different folds, repeat the process of evaluation and ultimately average all those test errors is to take the most out of small amount of data. Since dividing the dataset into 3 fixed sets could lead to a not enough of data in the train, validation and test sets, in k-fold cross validation there are only two sets (one fold for test and the remaining for training) and we can maximize the number of samples per set. In this particular case, since the test set is not fixed and the final evaluation is the average of all k iterations, there is not an issue with the model fitting too much the test set because it is variable in each fold.

2.3 Hyper-Model

The hyper-model concept was introduced by Pollak and Link (2016) where a model of models is built and applied to industrial scenarios. Complementarily, the authors introduce the notion of *condition* that are fixed quantities that govern a certain industrial process, like thickness in metal sheets for welding processes or deep drawing. This concept of condition is what defines each task in the context of ZSL, where different conditions mean different tasks. Assuming that a model is a set of base functions that transforms a certain input into an output, a model has always associated a condition that quantitatively describes the task to learn. Based on this, the main idea is to build a hyper-model to generate models for a whole continuum of conditions, aiming at mapping model parameters from the base functions into a set of conditions. This way, by providing a set of new conditions, it is possible to derive a new set of model parameters and build a new model for prediction in the context of those conditions. As one might have realized by now, this approach is independent from being a regression or classification problem. As far as the parameters of base functions and conditions are available, the hyper-model can be applied to both settings.

Defining these conditions as $\varsigma_n \in \mathbb{R}^c$ for the n^{th} model and c feature vectors, and a model as $z_n = f_{\lambda_n}(x)$ that maps an input x into an output z_n , being λ_n the process model parameters that build f_{λ_n} , is possible to create an hyper-model. Thus, the model can be represented as a linear combination of some base functions ϕ and the process model parameters λ_i

$$f_{\lambda_n}(x) = \sum_{i=1}^N \lambda_i \phi_i(x) \quad (2.3)$$

Based on this, the hyper-model allows to relate the model parameters with conditions represented by the following expression:

$$\varsigma = g_{\beta}(\lambda) \quad (2.4)$$

where β are the hyper-model parameters. Ultimately, also the hyper-model can be expressed as a linear combination of some base functions Ψ and the hyper-model parameters:

$$g_{\beta}(\lambda) = \sum_k \beta_k \Psi_k(\lambda) \quad (2.5)$$

This way, we formulate the problem as finding the hyper-model parameters to derive a transformation function that maps the model parameters λ to conditions ζ . However, as the authors state, a necessary condition for the previous formulation of an hyper-model is an homogeneous representation of all involved task functions, meaning that the base functions used for the models need to be the same, like using a degree 2 polynomial for all the models. This means that if we use different base functions we need to find a way to bring the model parameters into a the same common representation.

The method proposed to tackle this limitation lies in the idea of not using explicitly the parameters λ of the models, but instead, take a step back and directly use data. As the most suitable machine learning techniques to use highly depend upon data, if two different datasets have different properties, different techniques might be applied, like a Support Vector Regression in one dataset and Linear Regression in other. As stated by Wolpert and Macready (1997) from the No Free Lunch theorem, *"if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems"*. Hence, if different datasets have different types of complexities and properties, the same technique will perform well in some of these datasets, and worse in others.

In this context, the optimal solution for such a problem is to use the best algorithm possible for each dataset and take advantage on this to build the hyper-model. To that intent, our proposal is to use directly the data instead of the model parameters. In the next subsection we will detail the SSM approach that is suitable to explore the properties of data and therefore, a suitable candidate to achieve this common representation to train the hyper-model.

2.4 Statistical Shape Modeling

The Statistical Shape Model (SSM) (Cootes et al., 1995) is a widely used technique for image segmentation that analyses the geometrical properties of a set of given shapes or objects by creating deformable models using statistical information. As a mathematical transform to be applied to these set of shapes, the most common techniques are Principal Component Analysis, Approximated Principal Geodesic Analysis, Hierarchical Regional PCA (Mesejo et al., 2016) and Singular Value Decomposition, where non-affine modes of deformation are calculated.

The problem definition for this area of research is framed as a maximization problem to overlap a deformable model in the object / region of interest. The overall idea behind this method is to obtain the optimal affine (rotation, scale and position) and non-affine transformation parameters where a deformable model is built and matches a segment of an image. The same way this method assumes that there exist specific shape variations and these can be quantified forming a deformable

model, is the same way that we assumed that these variations also exist in different tasks, and a deformable model for a set of tasks can be derived.

There exist multiple recent examples that use such an approach, most commonly in medical imaging, like Shakeri et al. (2016) that proposes a novel approach of groupwise shape analysis that is able to analyze two study groups (healthy and pathological). The main idea is perform morphological study to predict neurodevelopmental and neurodegenerative diseases, such as Alzheimer's, by quantifying sub-cortical shape variations by using Statistical Shape Analysis. Another example is presented in Nguyen et al. (2016), where the authors used the SPHARM-PDM (SPHERICAL HARMONIC) framework, introduced by Styner et al. (2006), to analyze 20 patients undergoing bilateral sagittal split osteotomy (BSSO). In Shin et al. (2016) the authors used Statistical Shape Analysis to optimally obtain the landmarks for midsagittal reference plane for evaluation of facial asymmetry. Moreover, in Yates et al. (2016) the authors study the statistical mean and boundary models of the human spleen in an occupant posture, by using PCA to find the modes of variations and boundary models.

In a more formal way, a shape can be defined as *all the geometrical information that remains when location, scale and rotational effects are filtered out from an object* (Dryden and Mardia, 1998). Therefore, the information about a shape can be represented in different formats such as landmarks, parametric description and deformation-based (Zhang and Golland, 2016). On one hand, landmarks are a set of points used to describe the shape reliably, where methods can be used to calculate these automatically or be manually annotated, e.g. in a CT scan. On the other hand, the parametric descriptors are functional approximations of the shape, and therefore limiting shapes to a set of coefficients. Ultimately, the deformation representation is based on shape matching between images and a template, considering smooth constraints in the deformation field.

Defined as *a point of correspondence on each object that matches between and within populations* (Dryden and Mardia, 1998), a landmark is a point on a shape that as a direct correspondence in all the other shapes used to build a deformable model. Is this correspondence between points that allows the statistical processing to calculate the deformation of each shape in relation to its mean. Normally, the landmarks are described as a kn element vector x , where

$$x = [x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n]^T \quad (2.6)$$

in this case, for the dimensionality of the landmark representation space we have $k=2$, and n the number of landmarks of a given shape. Hence, to create a deformable model we need multiple shapes

$$x_i \in \mathbb{R}^k, i = 1, \dots, N \quad (2.7)$$

being N the number shapes available for analysis. Here we consider a shape as belonging to a specific model or task, and the landmarks of each shape are all the instances from a dataset, where both input and output features can be included. Therefore, each shape is the dataset to build a model. However, since we cannot ensure beforehand that all the datasets have the same size, we

also cannot assume that the datasets used to build the models are suitable to create the deformable model (remember that all the shapes need to have the same number of landmarks and they should match between each others). Instead, we can take advantage on the generalization capability of the models and sample a dataset per model, which will be considered a shape. This will guarantee that all the shapes have the same number of landmarks. For this intent, we need to ensure that the inputs provided to all the models are the same to guarantee the consistency of all shapes. Again, remember the assumption in the context of the present work of $P(X_i) = P(X_j)$ where the distribution of the input feature space is the same for all models, and hence is valid to use the same values of inputs to draw a new dataset from a model. Hence, to build the deformable model, only the output data should be used since no information is gained from using input data. Nevertheless, the proposed HPM algorithm can be generalized to regression problems with $P(X_i) \neq P(X_j)$, where input data can be included as far as the landmarks match between each other.

In order to build the deformable model, a mathematical transformation needs to be learned. One of the common techniques used to learn this transformation is the Principal Component Analysis (PCA) introduced by Flury (1988) which assumes a multi-variate Gaussian distribution. We will be describing the decomposition process using PCA, which is composed by the following steps:

1. Calculate the Mean Shape:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (2.8)$$

where N is the number of shapes;

2. Calculate the Covariance Matrix:

$$C = \frac{1}{N} \sum_{i=1}^N D D^T \quad (2.9)$$

where

$$D = ((x_1 - \bar{x}), \dots, (x_N - \bar{x})) \quad (2.10)$$

3. Calculate eigenvalues and eigenvectors of C :

$$C \phi_k = \lambda_k \phi_k \quad (2.11)$$

where ϕ are the eigenvectors and λ are the eigenvalues.

When there are fewer instances in the dataset when compared to the number of dimensions, the eigenvectors and eigenvalues can be efficiently calculated as follows:

$$\frac{1}{N} D^T D q_k = \mu_k q_k \quad (2.12)$$

multiplying both sides by D we obtain

$$C(Dq_k) = \mu_k(Dq_k) \quad (2.13)$$

From equation 2.13 we infer that $\phi_k = Dq_k$ and $\lambda_k = \mu_k$. The result of PCA is a decreasing order of the non-negative eigenvalues that represent the significance of each eigenvector (principal component). These eigenvectors are the modes of variations that allow to deform the model. With the most significant eigenvectors calculated, we can now approximate any training set, x' , using the following expression:

$$x' = \bar{x} + \phi b \quad (2.14)$$

where b are the parameters for the deformable model, and is given by the following expression:

$$b = \phi^T(x - \bar{x}) \quad (2.15)$$

The variation of b allows to change the shape of the deformable model, and normally they are constrained to $\pm 3\sqrt{\lambda_i}$ to provide similar shapes to those present in the original dataset.

Based on this, the common representation to be used in the hyper-model are the b values that allows to reconstruct the original shape, instead of using explicitly the model parameters. We consider the use of the statistical shape model concept the key to explore the best out of the each dataset properties and complexity. The approach taken to learn the deformable parameters for each shape can be seen as an unsupervised way to create a common space where multiple tasks have the same representation. If one uses different techniques to model the datasets as a way to increase generalization, as performed by HPM, the most organic step to take is to find a way to translate the different representations of model parameters into the same common space. Despite being an approach worth pursuing, taking the step to generate data from the trained models to form shapes, build a deformable model and ultimately use the parameters of this deformable model seemed the most effective, and above all, flexible way of creating this common representation.

In the following subsection the proposed algorithm will be detailed, which will glue together the presented methods of hyper-model and statistical shape model.

Chapter 3

Literature Review

3.1 Transfer Learning

The main purpose of Transfer Learning is to reuse past experience and knowledge to solve the current problem. Normally, machine learning algorithms focus on isolated tasks that cannot be inherently used for other tasks. From all the training data associated with specific task, the goal of Transfer Learning is to assist on the learning task for a future problem of interest. Therefore, this kind of domain proposes to solve the problem of transfer learning from different, but similar, tasks (Pan and Yang, 2010). Techniques that enable knowledge transfer represent progress towards making machine learning as efficient as human learning (Torrey and Shavlik, 2009). This statement comes from the fact that humans do not learn from scratch all the new tasks they need to perform. Otherwise, the reuse of past experiences and acquired theoretical and empirical knowledge plays a significant role in how fast a human can learn.

In a more formal representation, we introduce a notation for better understanding and further exploration of Transfer Learning methods. The first concepts that one needs to know for Transfer Learning is *Domain* and *Task*. Here *Domain* is represented by D , and consists of a feature space X and marginal probability distribution $P(X)$, where $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \in \mathcal{X}$. This means that if two domains are different, either or both feature space and the probability distribution are different. Therefore, we represent a domain as $D = \{X, P(X)\}$. For the *Task* definition, it is represented by T and consists of a label space Y and predictive function $f(\cdot)$, resulting in $T = \{Y, f(\cdot)\}$. In this case, the predictor represents a model that maps some input space into an output space, that can be trained using supervised machine learning techniques. For example, a probability predictor would take the form of $P(Y|X)$, where Y is the class or prediction and the X is the new instance to be classified or predicted. Based on these definitions, we can now define both source and target domain as D_s and D_t , correspondingly, and source and target tasks as T_s and T_t , correspondingly.

Once these first concepts are explained, then we can formally state that if two domains are different, for example the source and target domains $D_s \neq D_t$, then we can assume that either $X_s \neq X_t$ or / and $P_s(X) \neq P_t(X)$. In case of different tasks, once again they could be the source

and target tasks $T_s \neq T_t$, we can assume that either $Y_s \neq Y_t$ or / and $P(Y_s|X_s) \neq P(Y_t|X_t)$, in case the predictors.

According to Pan and Yang (2010), Transfer Learning can be categorized on 3 different settings according to the nature of knowledge transfer. There might exist cases where labeled data on the target domain could exist, cases where the marginal distribution between domains are not the same, or even the conditional distribution of predictors are different from source to target task, and so forth. Therefore, there's a need to frame each different context so the correct techniques and methods can be explored and applied. This way, these categories are: 1) Inductive Transfer Learning, 2) Transductive Transfer Learning, and 3) Unsupervised Transfer Learning.

Inductive Transfer Learning

For Inductive Transfer Learning, the idea is to improve the learning process of a certain target predictor $f_t(\cdot)$ in D_t , based on a single or multiple source tasks, independently from source and target domains being the same, assuming that $T_s \neq T_t$. The only requirement for Inductive Transfer is that data of target domain should be available. However, data from source domain, may or may not be available. Therefore, this leads to two different cases:

1. Labeled data is available on both source and target domains: The purpose is to use the tools available to transfer all the knowledge from the source task to the target task. Normally, multi-task learning (MTL) is applied;
2. Labeled data is not available on the source domain, but unlabeled data is, and there's available data on target domain: In this case, usually a self-taught learning approach is used.

Transductive Transfer Learning

In the case of Transductive Transfer Learning, the premise is that the source and target tasks are the same $T_s = T_t$, but the domains are different $D_s \neq D_t$, and in this particular case, no labeled data is available on the target domain. Based on this, we can have two different cases:

1. The feature space of both source and target domain are different $X_s \neq X_t$;
2. The feature space of both source and target domain is the same $X_s = X_t$, but the probability distributions are different $P_s(X_s) \neq P_t(X_t)$: This case is normally addressed with Domain Adaptation techniques, along with Sample Selection Bias and Covariate Shift.

In both cases presented of $X_s \neq X_t$ and $P_s(X_s) \neq P_t(X_t)$ the existing differences in input feature and marginal distributions are normally used to build a common feature space (often called latent representation) $\tilde{X}$ where a predictor can be trained using this common feature space on the source domain, and then use this predictor to label target input data to get the corresponding labeled data. It is based on this alignment on between tasks that unlabeled target input data can be classified.

Unsupervised Transfer Learning

Finally, for Unsupervised Transfer Learning neither source nor target data is labeled. Hence, the main purpose is to solve unsupervised learning tasks using the same premise as Inductive

Transfer, where source and target tasks are related somehow, and the learning process of the target task is improved by the source task. In this particular case, and due to nature of the Unsupervised problem, both input and output labels are not available, so it is applicable for Clustering and Dimensionality Reduction tasks.

For a more comprehensive understanding of the premises and context for Transfer Learning, Figure 3.1 presents a graphical summary and intuitive representation.

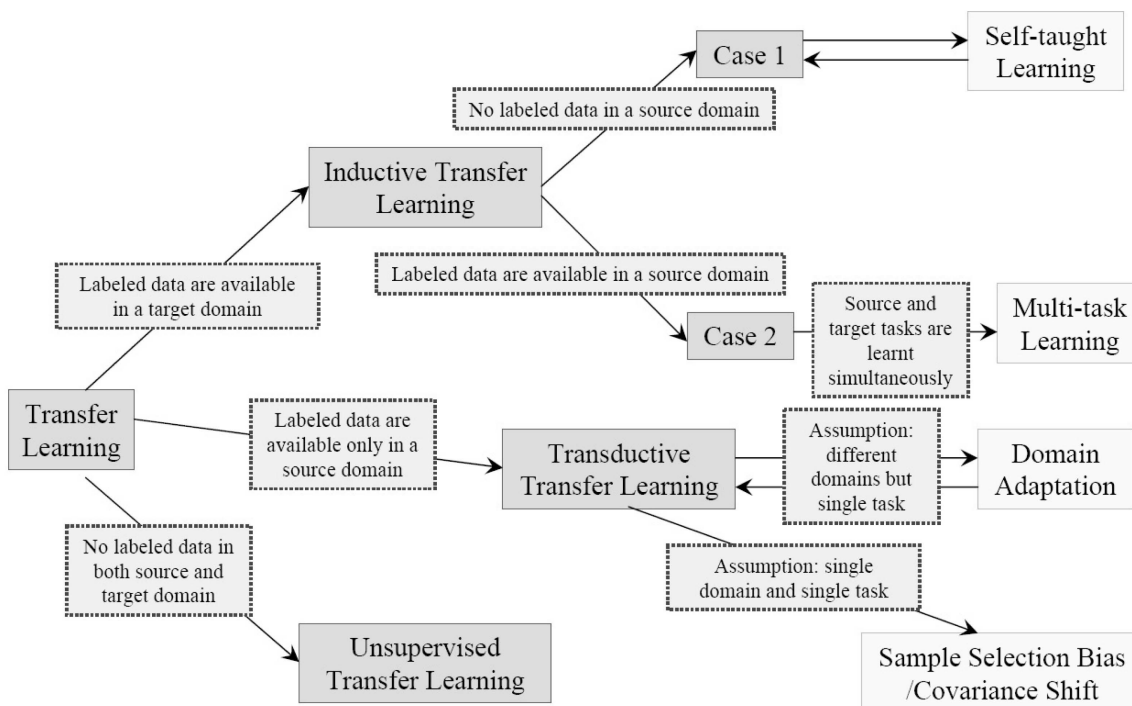


Figure 3.1: Summary of all three different classes that compose Transfer Learning. Pan and Yang (2010)

The traditional machine learning techniques assume that $D_s = D_t$ and $T_s = T_t$. It is called traditional machine learning because is more related with the well-established knowledge about the field. It implies that training and test data have the same distribution, and hence a predictor can be trained directly from available data without the need for knowledge transfer.

According to Pan and Yang (2010) and Weiss et al. (2016), so far, we have seen how to transfer knowledge from a source task to a target task, independent from being in the same domain. Additionally, there's the question related with "what to transfer" which is normally related with what part of knowledge is more suitable to transfer across tasks or domains. Hence, regarding what to transfer, there are four different approaches to do so: 1) Instance-based Transfer Learning: The instances of source task or domain are reweighted in order to improve the target predictor; 2) Feature-Representation Transfer Learning: These are classified in two ways 2.1) The first is related with the reweighting of features of the source domain to approximate it to the target domain. According to Weiss et al. (2016), this is also called the asymmetric transformation because only

makes use of source features; 2.2) The second is based on creating a common latent representation using data from both source and target domains, and is also called symmetric transformation; 3) Parameter Transfer Learning: This approach is commonly used when a model is already available on the source domain, and the parameters of such model are used in two ways: 3.1) To build the predictor for the target task, e.g. using Stacked Denoising Auto-encoders to build up a Deep Neural Network and use these weights to initialize the target predictor; 3.2) Ensemble techniques that make use of several weak learners to build up the best hypothesis for the target task; and 4) Relational-Knowledge Transfer Learning: This happens when there's an explicit or defined relation between source and target tasks.

3.2 Introduction to Zero-Shot Learning

One of the most intriguing and fascinating capability of humans is to generalize upon multiple and diverse tasks. When only presented with few examples, humans can quickly learn particular features of a certain object or task, distinguishing it from different classes of objects. The human capability to generalize allows to extrapolate and infer which kind of object it might be from previously seen examples of different classes. Take as an example a chair. It is very intuitive for a human to classify a chair when a flat surface from a particular hard material, e.g. wood or aluminum, sits on top of 4 legs. Based only on this example, a person can classify as a chair the same type of arrangement but with longer legs, larger top surfaces where you can seat, and when with a larger or smaller amount of legs. As presented by Biederman (1987), humans have the capability to identify and distinguish about 30,000 objects, and for each of these objects there was no need to show thousands of millions of images the same object in order to recognize and discriminate it, as it is often required in Deep Learning approaches such as in Convolutional Neural Networks (CNNs). In fact, a great majority of humans would become confused if such a number of images of the same object was shown. Instead, based on a small amount of images, humans can generalize by extracting certain key features of an object and form high level representations to distinguish among multiple objects. By relating all the information learned and internal object representations, it is easier to learn from small amount of pictures. This capability to extract particular features and properties of an object and then generalize to other unseen classes of objects is one of the greatest challenges in Artificial Intelligence nowadays.

Similar to this approach, in machine learning there's a particular area that is concerned about the presented 'chair' example. This area is called Zero-Shot Learning (ZSL) and aims at solving a certain task based solely on the task description when different, but similar tasks were already learned. These high-level descriptions aim at differentiating different tasks and from those, some information can be extracted. An example is on multi-task medical domains where the patients are grouped into tasks and described using demographic and disease data as presented in Oyen et al. (2012). Another application is in control problems, where dynamical system parameters such as spring, mass and damper constants describe the specific task (Isele et al. (2016)). Descriptors can also be derived from external sources such as Wikipedia by using unsupervised approaches like the ones used by Pennington et al. (2014); Palatucci et al. (2009); Socher et al. (2013).

A definition for this particular type of problem was first presented by Larochelle et al. (2008) where the authors first named it as Zero-data Learning and defined it as: *"Zero-data learning corresponds to learning a problem for which no training data are available for some classes or tasks and only descriptions of these classes / tasks are given."* Humans can imagine and visualize certain objects when reading a book or an article. Based on 1) a description of the object and 2) prior knowledge about the world and intuition, humans can materialize such imagined objects by drawing, sculpturing or even 3D modeling and recognize these if seen somewhere else. If this description about the object is present, based on all the learning throughout lifetime, humans can match their own interpretation and visualization of an object with the physical object

and determine if these are the same, or somehow similar in certain features. Normally, in such situations, intuition plays a significant role by matching an already learned object, problem or pattern and immediately recognizing it without great effort. Such concepts are the ones that ZSL is based on to build a set of algorithms and strategies for machine learning.

However, one might question the importance of ZSL in terms of its applicability and if it is really pertinent in the machine learning community. The main motivation behind ZSL is that, as depicted and explored by Larochelle et al. (2008), the number of tasks is far too large and data for each task is far too little. We have already seen some great advances in Artificial Intelligence where programs can reach superhuman capabilities in very specific tasks, such as playing world famous games like Chess, Go and even Dota 2. Despite all these great achievements in AI, these are not even close to the generalization of human capability and knowledge transfer from a set of tasks to new, unseen ones. Therefore, one of the advantages of ZSL is that one can create a specific classifier or predictor for a problem before trying to solve it. This means that a pre-assessment and understanding of a certain task can be made using such an approach. In case there are a large range of tasks to explore and pre-selection of set of tasks needs to be made (which is usually the case) one can see ZSL as a tool to design a decision support system where a ranking of the most relevant tasks to explore next can be presented. Take for example the case of developing new toxins where a large combination of molecular compounds needs to be tested. With ZSL a pre-assessment can be made on which combinations worth pursuing before trying these out in the lab. Hence, a ranking of molecular compound combinations can be made based on existing information about experiments already performed and a high-level description about the combination itself.

In the next subsections, the most relevant works about the ZSL will be presented and the techniques used detailed, so one can understand how the ZSL problem was addressed and specific scenarios were tackled. Hence, first the application of ZSL in image processing will be made, starting from Two-Stage approaches and then explaining the latest methods based on Compatibility Functions. Then, different application scenarios will also be discussed to broaden the perspective of this machine learning area. Ultimately, a discussion will be made about all the performed literature review.

3.3 Related work in Zero-Shot Learning for Image Processing

As an introduction to ZSL in the application context of Image Processing one of the first publications of this field is presented, even before the categorization of techniques in Two-Stage and Compatibility Functions. This work was presented by Larochelle et al. (2008) where the authors first make a definition of *Zero-Data Learning* in order to distinguish their work from others and address specific issues that were not addressed until that time. These are related with One-Shot Learning, Lifelong Learning and the "cold start" problem in recommender systems where tasks are objects that can be recommended to users. In their work, they present two different approaches to the problem: 1) input space view and 2) model space view.

The first approach uses a concatenation of the input x_t and the task / class description $d(z)$ and by using a supervised learning algorithm train a model $f^*(.)$ to predict y_t^z . Hence, for a new class z^* and input x^* , one could predict the output by using $f^*([x^*, d(z^*)])$. In case of a classifier that discriminates objects, being the outputs between 0 and 1, if one wants to discriminate between two classes z_1^* and z_2^* , the largest output of $f^*(.)$ for each of the class / task descriptions can be used, like in Maximum A Posteriori (MAP) when using probabilistic models. In this approach, SVM was used to train two different classifiers, one with Gaussian Kernel (SVM rbf) and the other with Polynomial Kernel of degree 2 (SVM poly).

The second approach is more model-driven, and is defined by $f_z(x) = g_{d(z)}(x)$. By defining a joint distribution $p(x, d(z))$ one can then set $g_{d(z)}(x) = p(x|d(z))$ and learn the parameters that calculate the probability of a certain input x being from class $d(z)$. However, a different way to achieve model space view is also presented. It uses the model parameters θ to train a model that maps class description into model parameters. Assuming a family of functions $h_\theta(x)$ parametrized by θ , if one defines a function $q(d(z))$ where the output is the same as the parameter space of θ , then the output for a particular x with class description of $d(z)$ is $h_{q(d(z))}(x)$. With this, the model space view is obtained by $f_z(x) = h_{q(d(z))}(x)$. This approach is similar to the one presented by Ghosn and Bengio (2003) where instead of using an explicit task description such as a 7 x 5 pixel image, this description was learned from the available tasks. Despite being more general, the authors use the $g_{d(z)}(x)$.

For this approach, they have used 4 different models: 1) Linear generative model from classification called LinGen, 2) Linear discriminator with a multi-class cost (softmax) called LinDisc-1-vs-all, 3) Linear discriminator using a binary cost (cross-entropy) called LinDisc-0-1, and 4) Non-linear version of LinDisc-0-1 was used by replacing $g_{d(z)}(x)$ by $\tanh(Wx_t)$, which corresponds to an hidden layer in a feed-forward neural network, and they called NNet-0-1. For testing such an approach, 3 different datasets were used: 1) Character recognition in license plates, considering characters from 0 to 9 and A to Z and 3 others accentuated characters, summing up a total of 40 classes and 200 samples per class; 2) Handwritten alphanumeric character recognition with characters from 0 to 1 and A to Z with 39 examples per class; 3) Molecular compound dataset provided by a pharmaceutical company where the main idea is to develop a system that could identify if a molecular compound x_t is active $y_t^z = 1$ in the presence of a biological agent z for each of the 7 provided agents. In the latter, the authors used two approaches for the input values, being one with 469 real-values features and the other with 3640 binary features. For (1) and (2) the task description is a direct derivation from the inputs, where a computerized version of the character used are encoded in the 7 x 5 pixel image. For (3) the task descriptor was provided by the company and is composed by a 25-dimensional features and do not come from the same space as the input, contrary to (1) and (2).

As for the experimental setup, the first test was to discriminate an input between 2 unseen classes during the training process. From an initial pool of learned classes, the idea is to randomly select a set of 2 classes not included in the initial pool in order to discriminate between them, assess the discrimination performance, and then train again a classifier with the initial pool of

tasks, plus the 2 randomly selected tasks. For the molecular compound, each of the 7 agents were tested using only the 6 remaining agents, where the AUL (Area Under the LIFT curve) was used to calculate the performance.

As main results, the authors assessed the performance of the proposed approaches against the number of classes used to calculate the output of unseen task / class in the training process. The results for the character recognition are mainly that classification error tends to decrease as the number of classes increases, where the SVM with Gaussian kernel provides almost perfect discrimination of unseen characters. NNet-0-1 yields also good results being the second-best technique. As for the handwritten characters all the models have nearly the same behavior by decreasing the error with an increasing number of classes. However, there's an interesting behavior for LinGen, which is a U-shaped error graph where two options are hypothesized: 1) Atypical classes that do not directly help to predict or classify unseen classes and 2) the model is too simple to grasp complexity of the problem. Additionally, the authors performed a test where they allowed the techniques to classify new instances not only on the test classes, but also in the training classes. Naturally, this yielded worse results because only predicting between 2 classes the output was constrained a priori, whereas in this scenario no constraint exists and much more classes could be chosen. In this setting, the authors observe an "unusual type of overfitting" where the model is overconfident classifying on the training classes when presented with test instances. Moreover, they observe that slight changes in the model would produce great changes in performance. In both (1) and (2) cases, there's no clear difference between input and model space view, apart from the SVM rbf that produced near perfect classifications. As for the molecular compound dataset, model space view performed better than input space view, and performed better than random ranking. Finally, one of the main conclusions of the presented work is that zero-data learning sits between purely human developed programs by defined the system rules, and data-based systems like the ones used in machine learning where rules are derived from data. This type of approach uses less data and less human labor in order to perform some decision making and help developing new intelligent systems.

3.3.1 Two-Stage approaches

Other interesting work that paved the way towards a more formal definition and theory of Zero-Shot Learning is the work of Palatucci et al. (2009). In their work a two-stage approach is presented where the same concept as task / class description is used but now called semantic feature space. This naming is mainly due to the application of ZSL to the problem of brain imaging classification into a specific word. For their approach, the first stage is related with mapping brain images X^d of dimension d into a semantic feature space F^p of dimension p , defining the following function $\mathcal{S} : X^d \rightarrow F^p$. Then, the second stage maps this semantic feature space F^p into the desired class label Y where another function is defined $\mathcal{L} : F^p \rightarrow Y$. Hence, the main idea is to train a classifier $\mathcal{H}$ that can map the input X^d into the correct class label Y using the two presented functions, where $\mathcal{H} = \mathcal{L}(\mathcal{S}(.))$ called Semantic Output Code (SOC) classifier. The main reason to separate the learning procedure in two stages and avoid training directly a function is that one

wants to predict class labels that are not present in the training phase. In a normal classification system setting, the classes used in the training process are the ones that will be predicted in the test phase, so if a new input is presented it can only be classified into one of the existing classes, and not new ones. Therefore, the goal is to train $\mathcal{S}$ with a set of inputs that map into certain class labels, and train $\mathcal{L}$ with a larger spectrum of class labels. So, $\mathcal{L}$ should be trained with as much as possible class labels and $\mathcal{S}$ with the existing dataset. As the authors refer, the SOC classifier can be useful when the knowledge base used for $\mathcal{L}$ covers more of the possible values for Y than the ones covered by $\mathcal{S}$.

The dataset used contains neural activity observed in 9 different human participants while watching 5 specific words from 12 different categories, summing up a total of 60 words. For each word, 6 different fMRI scans were made composed by 20,000 voxels where an average per voxel was calculated creating only one brain scan for a specific word. By using the *voxel-stability-criterion*, the brain scan dimensionality was reduced from the 20,000 voxels to 500. Two different knowledge bases were created for the semantic feature space for all 60 words, being one based on corpus5000 and the other on human218. With corpus5000, the semantic feature space is composed by 5000 co-occurrence vector of the most frequent words from Google Trillion-Word-Corpus, and with human218 a semantic feature space of dimension 218 was built representing questions related with size, shape, surface, etc. where users answered these on a scale from 1 to 5. As for the first stage, the authors used multiple output linear regression to learn $\mathcal{S}$ and 1-nearest neighbor classifier to learn $\mathcal{L}$. As for the experiments, three different questions were addressed: 1) *Can we build a classifier to discriminate between two classes, where neither classes appeared in the training set?* 2) *How is the classifier able to discriminate between closely related novel classes?* and 3) *Can we decode the word from a large set of possible words?* As for the question (1), $\mathcal{S}$ was only trained with 58 brain images, where a leave-two-out-cross-validation was performed, and $\mathcal{L}$ with all 60. This resulted in 3,540 comparisons and the approach achieved a performance of 80.9% for the human218 and 69.7% for the corpus5000. For question (2) the inputs for the two classes left out of $\mathcal{S}$ were used representing a bear and a dog and the classifier clearly distinguished between the two in almost all the 10 semantic questions selected for discrimination. Finally, for question (3) the authors expanded the knowledge base used to train $\mathcal{L}$ using mri60 (with 60 nouns) and noun940 (940 nouns), and tried to predict the correct word for the held-out input and semantic feature using again both human218 and corpus5000. For this experiment, the authors calculated the median and mean rank accuracy, where for noun940 the median rank accuracy is above 90% and mean about 80% for human218 and for corpus5000 the median is around 79% and mean of 70%. As for the mri60, the median rank accuracy was around 88% and mean 79% for human218 and similar median and mean were obtained for corpus5000.

One of the key differences between the presented work and the one presented by Larochelle et al. (2008) is that only the input of a brain image is required in order to classify the corresponding word, whereas in Larochelle et al. (2008) both input and task description are required, as seen in the expression used by the authors $h_{q(d(z))}(x)$. This is one of the greatest advantages of using a two-stage approach where classes can be learned in the latent space even when there's no input

available for all the classes. However, one of the disadvantages is the training of two different functions where the performance of the function in the first stage greatly influences the performance of the whole classifier, even if the performance of the function in the second stage is good. This error accumulation from the first to the second stage can invalidate the whole SOC approach for different application scenarios.

Socher et al. (2013) presented a similar two-stage approach called Cross-Model Transfer (CMT), where the main idea is to train a model that is able to map image features into a word vector space, and then have a second model that is trained to classify these word vectors into the correct label classes. Again, it is assumed that more classes are present in the second stage rather than in the first. In the first stage, based on the work of Coates and Ng (2011), the authors have extracted a set of unsupervised image features from raw image pixels in order to map these into a semantic space (word vector). Hence, for the semantic space, the authors have used an unsupervised model from Huang et al. (2012) composed by 50-dimensional word space. This is contrary to the approach presented previously, where the word vectors are drawn from corpus5000 and human218. In sum, the idea of the first stage is to train a model to map these unsupervised image features into the unsupervised word vectors.

As for the second stage, the authors first assessed if the presented image is from seen or unseen classes, so then labels can be chosen based on likelihood. The main motivation for such an approach comes from the analysis performed on the semantic features where images from unseen classes are close to related images from seen classes, but not as much as the images from the same seen class. For example, cats are near dogs, but not so close as other types of dogs, so a likelihood approach would yield good results. One of the main goals of this approach is not only to develop a solution that yields good results for unseen classes, but also perform well in images that belong to already learned classes. For that, two novelty detection strategies will be applied using outlier techniques. For the first strategy, each class is mapped into a Gaussian distribution and the outlier detector is an indicator if the marginal probability is below a certain threshold T_y , where y is a specific class. A low value of probability means it has low probability of belonging to that particular class, so values above the threshold correspond to images belonging to seen classes, and otherwise. The second strategy, instead of discretizing the images into seen or unseen classes, aims at providing an outlier probability so a weighted classification can be made. So, for this second stage, if a new image is from the seen classes, a regular classifier is used, otherwise, a Gaussian distribution is calculated and classes assigned based on their likelihood.

As for the experiments performed, the authors have used the CIFAR-10 with 10 classes and 5,000 $32 \times 32 \times 3$ RGB images each. In the first assessment, the performance only providing seen classes was made, together with using also unseen classes. This means that for the training process, only 8 of 10 classes were used, being the left-out classes reserved for zero-shot analysis. For the seen classes, the left-out classes are "cat" and "truck" and an accuracy of 82.5% was achieved. As for the unseen classes (zero-shot learning itself), different pairs of left-out classes were tested. Lower accuracies are obtained if the two classes are related and higher accuracies if the two classes are not related. For instance, the lowest accuracy of around 50% was achieved when the left-out

classes were "cat" and "dog" and the highest around 90% when the classes were "cat" and "truck". This happens because if one of two related classes is in seen and the other in unseen classes, one can be a good semantic distinction for the unseen class and increase the probability of correctly classify the image. The authors also performed the same kind of test in CIFAR-100, where 100 classes with 500 32 x 32 x 3 RGB images each were used, together with the CIFAR-10 with 6 left-out classes for zero-shot learning analysis. For the seen classes, a performance of 52.7% was achieved, where 16.6% is choosing a class by chance. As for the unseen classes, with a 2-layer neural network for mapping image features into semantic space, a performance of 42.2% was achieved.

Another interesting work worth referring that is also related with this two-stage approach was first introduced by Lampert et al. (2009) and then further extended by the same authors in Lampert et al. (2014), where two different techniques were presented: 1) Direct Attribute Prediction (DAP); and 2) Indirect Attribute Prediction (IAP). The authors propose a probabilistic model as a way to address the problem of predicting the class labels of images, where the test classes were not seen / used in the training process. Hence, training classes $Y = \{y_1, \dots, y_K\}$ are disjoint from $Z = \{z_1, \dots, z_L\}$ test classes. For the DAP technique, a probabilistic model was used to estimate the probability of binary-value attributes given a certain image, so unseen images at test phase could also have an estimate into this attribute space. Hence, the authors modeled $p(a|x) = \prod_{m=1}^M p(a_m|x)$, where a_m is the attribute representation and x the corresponding image. Moreover, a probabilistic model was also trained to estimate the probability of a certain attribute set is from a specific unseen class. This model was defined as $p(z|a) = \frac{p(z)}{p(a^z)} [[a = a^z]]$, where a^z is the attribute representation for the unseen class z , and $[[a = a^z]]$ is the Iverson's bracket notation (Knuth (1992)), where $[[P]] = 1$ if condition P is true, and 0 is false. By combining both stages, the final probabilistic model is expressed as:

$$p(z|x) = \sum_{a \in \{0,1\}^M} p(z|a)p(a|x) \quad (3.1)$$

The predictions from image to unseen class were then made using Maximum A Posteriori (MAP):

$$f(x) = \arg \max_{l=1, \dots, L} p(z = l|x) \quad (3.2)$$

As for the IAP technique, instead of a two-stage approach, an additional stage was used. First a mapping between image and training classes is performed, as a regular multiclass classifier, estimating $p(y_k|x)$ for each training class y_k . Then, a mapping between training classes and attributes is made $p(a_m|y) = [[a_m = a_m^y]]$, resulting in a model that maps images in attributes as $p(a_m|x) = \sum_{k=1}^K p(a_m|y_k)p(y_k|x)$. Hence, by using Equation 3.2 it is possible to predict the class label for a new image, where the corresponding class was not used in the training process, addressing this way the ZSL problem.

To test these techniques in a ZSL setting, the authors created the Animals with Attributes

(AwA) dataset, composed by over 30,000 images, 50 animal classes and 85 semantic attributes. Additionally, the authors also tested these in existing datasets such as the aPascal/aYahoo and SUN Attributes datasets. The first is composed by 12,695 images from the PASCAL VOC 2008 data set (training set) with 64 binary attribute and 2,644 images collected from Yahoo search engine (test). Both train and test sets have disjoint classes, where PASCAL has 20 classes and Yahoo has 12 classes. As for the latter, it consists of 14,340 images from 717 classes for fine-grained scene categorization. For each image, there are 102 binary attributes that describe material, surface, lighting conditions, functions and so forth.

As for the experimental setup, using the AwA dataset, 40 classes were used in the training process and the remaining 10 in the test phase (5-fold cross validation). For the SUN Attributes dataset a 10-fold cross validation approach was used, meaning that approximately 637 classes were used for training and 70 classes for the test phase. Both fixed and random cross validation class splits for the datasets were used, where the fixed class splits were defined to maximizing the diversity in the training set to increase the probability of choosing the right unseen class for test images. For DAP, Support Vector Machine (SVM) was used to predict images into binary attributes, where one SVM was trained per attribute $a_1, \dots, a_M$, being M the number of binary attributes. In order to get the probability estimates, the SVM prediction was transformed using a sigmoid curve for Platt scaling. For IAP, SVMs were also used with Platt scaling to predict training classes from images, now with a one-versus-rest, and by using $p(a_m|x)$ the corresponding attributes can be estimated. Then, as in DAP, Equation 3.2 can be used to perform a MAP prediction for the test class. The chosen metrics for evaluation were the normalized multiclass accuracy (MC acc.) on the test set - how well the prediction for any test class can be used to retrieve images from the images of test set by using the area under the ROC curve (classAUC) - and the accuracy of individual attribute prediction in terms of AUC for the test set (attrAUC).

The authors compare both DAP and IAP with two other methods from ZSL. Both have the same principle of training a classifier for the training set, and then estimate the most similar test class by using the trained classifier for the test images. This way, a test image is classified into a train class, and then the most similar test class is chosen using two different similarity criteria: 1) Hamming distance (CT-H) and 2) Cross Correlation (CT-cc). In the overall cases, the DAP and IAP are better than CT-cc and CT-H, where there's not much difference between both DAP and IAP. In terms of test set class prediction, the results are not as good as a supervised approach but are above random chance. For AwA with fixed train and test class splits, the MC acc. is of 41.4% for DAP and 42.2% for IAP, as the CT-H and CT-cc have accuracies of 30.7 and 30.8%. For classAUC, DAP achieved a retrieval of 81.4%, IAP achieved 80%, CT-cc 73.4% and CT-H also 73.4%. For the attrAUC, the DAP achieved 72.8% and IAP achieved 72.1% (attrAUC is specific for DAP and IAP). For random splits, these results for MC acc. decrease to 37.1% for DAP, 34.1% for IAP, 27.7% for CT-cc and 27.3% for CT-H. For classAUC, DAP achieved a performance of 80.4%, IAP achieved 76.3%, CT-cc and CT-H achieved 72.4% and 72.8%, correspondingly. For attrAUC, the performances for DAP and IAP are 70.7% and 69.7%. For the other dataset results, please refer to the actual of Lampert et al. (2014).

The work of Qiao et al. (2016) presents an algorithm that was greatly inspired by DAP where the authors explored the relations and dependence between the attributes to increase the performance of the system. For this purpose, the authors considered a chain of dependent attributes where the joint probability of each attribute for a specific class is calculated, contrary to DAP which calculates the marginal probability. However, due to high amount of attributes it is difficult to calculate these joint probabilities, so first a clustering algorithm is applied to organize attributes into sets. Only after this process the probabilities are calculated for each cluster. Finally, the classes are predicted using MAP estimation, as used in DAP.

As for the datasets used, the authors have tested and compared their approach with the AWA and aPascal-aYahoo. Despite being an interesting work where different properties of attributes were explored, the results did not significantly improve the state of the art when compared with DAP. The best accuracy achieved in the AWA (aPascal-aYahoo) dataset is 44.14% (24.4%) while for DAP the accuracy was 42.5% (22.6%).

An interesting and pivotal work that explores the latest machine learning trends of deep learning is the one performed by Frome et al. (2013) where a composition of an ImageNet and a neural network to predict adjacent words in a text from a skip-gram language model was used. This constitutes a deep visual-semantic model that makes use of both visual and semantic embeddings from trained networks, and thus the name of the proposed method Deep Visual-Semantic Embedding Model (DeViSE). For that purpose, a joint learning is proposed using the hinge rank loss to yield the best dot-product similarity between visual model and word vector representation. One key characteristic of the experiments performed is related with a high scale tests, where the authors take the method to the extreme of training with 1K classes and test with 20K different classes not used in the training procedure.

Regarding the datasets, the authors have used the ILSVRC 2012 1K for training the ImageNet and 5.7 million documents (5.4 billion words) from Wikipedia to train the skip-gram model. As for test set, the ILSVRC 2012 20K was used. However, the authors have defined several experiments to assess the proposed method. First 3 different levels of difficulty for the zero-shot learning scenario were setup, ranging from the easiest with 1.589 labels not used in the training procedure that are within two tree hops from the training set called "2-hop", to a "3-hop" using the same approach, and the most difficult using the 20K labels. Additionally, the authors have tested the two scenarios where training data is included in the test set (called DeVISE+1K), and a purely zero-shot scenario where train and test sets (DeViSE-0) are completely disjoint. Finally, these two strategies were compared with the ImageNet baseline from ILSVRC 2012, which although not possible to use in a zero-shot setting, the closest predicted labels were compared to the ground truth. As evaluation metrics, the authors have predominately used the hierarchical precision@k that allows to evaluate how close a prediction is based on label hierarchy provided by the ImageNet repository, with $k = \{1, 2, 5, 10, 20\}$.

The results show that in most cases the proposed method surpassed the ImageNet baseline being the best results from DeVISE-0 with 0.217 for hierarchical precision@k=10 and 0.233 for k=20 in a 2-hop dataset; DeVISE-0 with 0.214 for k=10 and 0.236 for k=20 in a 3-hop dataset;

DeViSE+1K with 0.092 for $k=10$ and 0.101 for $k=20$ in the full 20K label dataset. These results depict a great potential for future real applications where one of the advantages of zero-shot learning is clearly extrapolating label predictions outside what the model was trained for.

3.3.2 Compatibility Functions approaches

First introduced by Akata et al. (2013) and then extended and generalized by the same group Akata et al. (2016), the Attribute Label Embedding (ALE) is presented as an alternative that outperforms some of the DAP method limitations (Lampert et al. (2014)). The authors state that ALE overcomes the limitations of being 1) a two-stage learning approach for ZSL problem, by 2) assuming the attributes on AwA are independent among themselves and 3) is not extendable to other sources of side information. Regarding (1) the problem is associated with not assuring that both attribute (first stage) and class prediction (second stage) are optimal because the learning process is not performed jointly, but separately. Hence, perhaps the prediction of attributes might be optimal, but not for class prediction. On (2) by assuming that attributes are independent, like "has stripes" and "has paws" for the AwA dataset, no additional information can be leveraged to increase the performance of the system, and the authors explored a hierarchical method to address such issue. Finally, limitation (3) is related with only using the attributes available on AwA and no other complementary information such as textual descriptions that can be automatically processed. This last aspect is particularly interesting when little training data is available and other sources might increase the performance of the system. Additionally, the authors explain also how ALE supports an incremental approach, specifically if class labels become available the system is able to accommodate them and increase its performance. To this particular case, the authors call few-shots learning.

Apart from the proposed ALE algorithm, the main contribution from the authors is a framework for learning label embedding with attributes in a ZSL problem. Hence, first a label embedding should be defined as a set of attribute vectors that correspond to a class label. The same way images can have some attributes that define stripes and color, the classes themselves can also have these attributes. Complementary to the previously described approaches, this means that both images and class labels have a latent representation of its own, instead of only the input image. To these latent representations, we should call image embedding to the image latent representation and label embedding to class latent representation. Hence, assuming that an image embedding is defined by $\theta : X \rightarrow \tilde{X}$ and a label embedding by $\phi : Y \rightarrow \tilde{Y}$, the prediction function can be defined as such:

$$F(x, y; W) = \theta(x)'W\phi(y) \quad (3.3)$$

where W are the parameters that should be learned to predict the correct class y for the input x . Based on this, the authors define an optimization problem to minimize the empirical risk and learn

the model parameters in order to maximize the compatibility between image and label embeddings as such:

$$\arg \min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n=1}^N \Delta(y_n, f(x_n)), \quad (3.4)$$

where N is the number of samples and $\Delta(y, z) = 0$ if $y = z$ and 1 otherwise. Considering this formulation, the label embedding $\varphi(y)$ for each class can be learned as well from the data, the same way as W . The label embedding for all classes is defined as Φ and as a matrix of stacked $\varphi(y)$, where each row is a class. The only restriction is that the dimension of the embedding should be found, but a strategy such as cross-validation can be used. Another option is to define the label embedding a priori as side information, as normally occurs in the previous algorithms for the image embedding. The same way features can be engineered by man to describe an image like "has paws" or "has stripes" as image embedding, features for label embedding can also be engineered by man as well or learned independently from the ALE learning process. This way, multiple forms of label embeddings can be explored, from textual corpora to class hierarchies. To this setting the authors also call "learning with side information".

The authors define the ALE algorithm on top of the presented framework. For that, an already existing algorithm called Web-Scale Annotation By Image Embedding (WSABIE) proposed by Weston et al. (2010) was used as a baseline to formulate ALE. Hence, the WSABIE algorithm defines an optimization function based on ranking, where it enforces the correct labels to rank higher than incorrect ones, and is expressed as follows:

$$R(\mathcal{S}; W, \Phi) = \frac{1}{N} \sum_{n=1}^N \frac{\beta_{r_{\Delta}(x_n, y_n)}}{r_{\Delta}(x_n, y_n)} \sum_{y \in \mathcal{Y}} \max\{0, \ell(x_n, y_n, y)\} \quad (3.5)$$

where $r_{\Delta}(x_n, y_n) = \sum_{y \in \mathcal{Y}} \mathbb{1}(\ell(x_n, y_n, y) > 0)$ is an upper-bound on the rank of label y_n ; $\beta_k = \sum_{j=1}^k \alpha_j$ where $\alpha_j = 1/j$ which can be seen as a quantity of penalty going from k to $k+1$; and $\ell(x_n, y_n, y) = \Delta(y_n, y) + \theta(x)'W[\varphi(y) - \varphi(y_n)]$. Hence, the introduced function allows to learn both W and label embedding Φ at the same time. For the optimization process, the authors use Stochastic Gradient Descent (SGD) as a convex-function is not guaranteed. Contrary to WSABIE, the ALE algorithm does not learn the label embedding, but instead, this embedding is given a priori as side information. Hence, the proposed algorithm is only designed to update the W parameters and the label embedding is $\Phi = \Phi^{\mathcal{A}}$ where the $\Phi^{\mathcal{A}}$ is the given label embedding. This way, this latent space can be precomputed as desired and introduced in the algorithm to find the optimal or near-optimal W .

Additionally, the authors propose a variation of the method when few labeled data is available on the test set for both image and label embedding called "few-shots learning". This is intended to address the problem of incrementally update the model with new data, as in streaming settings, and improve the model's performance. For this case, the optimization function was changed to contemplate a regularizer as follows $R(\mathcal{S}; W, \Phi) + \frac{\mu}{2} \|\Phi - \Phi^{\mathcal{A}}\|^2$ and now both W and Φ are updated simultaneously using the SGD.

One of the greatest contributions of this work is that ALE algorithm is compared with a handful of algorithms, apart from the DAP algorithm already stated. It is easy to see that if anyone is capable of defining a label embedding, the ALE algorithm can be readily used since these attributes are seen as side information. Therefore, the authors explore other kinds of embeddings such the Hierarchical Label Embedding (HLE), first proposed by Tsochantaridis et al. (2005). This embedding is a vector with hierarchical structure information of the image classes, like observed in the ornithology of birds. This vector is filled with hierarchical information from y and z classes, where $\xi_{y,z} = 1$ if $z \prec y$ or $z = y$ for each class, with $\prec$ meaning that z is an ancestor of y . Other explored embedding for comparison with ALE is the Word2Vec Label Embedding (WLE) proposed by Frome et al. (2013). This embedding makes use of co-occurrences of class names in textual corpora to produce word embeddings to be explored as side information for the ALE algorithm. For the ZSL problem, these 4 algorithms were considered: DAP, ALE, HLE, WLE.

For the few-shots learning, two more algorithms were considered. These are considered data-independent with fixed mapping from classes to a latent space, and by using the $R(\mathcal{S}; W, \Phi) + \frac{\mu}{2} \|\Phi - \Phi^A\|^2$ optimization function, both W and label embedding Φ are learned simultaneously. Hence, the On-Versus-Rest embedding (OVR) is proposed where the latent space is simply an identity mapping, where it is equivalent to independently train a classifier per class. For this case, the label embedding dimension is the same as the number of classes. Moreover, the Gaussian Label Embedding (GLE) was also proposed, but instead it uses Gaussian random projection matrices. For this algorithm, the number of dimensions needs to be optimized the same way as WSABIE algorithm. The two presented algorithms, together with WSABIE, are not suitable for the ZSL problem because the only way to include test classes that are disjoint from train classes in the learning process is when label embedding are seen as side information, where only W is learned. However, the authors could have explored the label embeddings from the few-shots learning in the ZSL setting, since any embedding is seen as side information.

The authors tested the algorithms in two datasets: 1) the already presented AwA dataset and 2) the CUB-200-2011 (CUB) data with 11,800 images of 200 bird classes. Based on these, they define the input and output embeddings (or image and label embedding). For both datasets, the image embeddings were calculated based on the provided images where SIFT descriptors are extracted and represented in a Fisher Vector. For the image embeddings, there are two versions of 4K and 64K dimensions. Regarding the label embeddings, three different types were developed: 1) Attributes that describe each class; 2) Hierarchical structure that represent each class; 3) Word2Vec based on English-language Wikipedia.

For the ZSL experiment, a 5-fold CV approach was used with 40 classes for training and 10 for testing. For the CUB dataset a 4-fold CV was used with 150 classes for training and 50 to test. The authors also test three different types of embeddings encoding: 1) Continuous between 0 and 1; 2) Binary being either 0 or 1; 3) Binary being either -1 or +1. The first assessment using only the ALE algorithm indicates a significant difference between continuous and binary encodings, favoring the continuous encoding. Also, the regularized version of the optimization function was used where it seems that only the ℓ_2 -normalization benefits the performance and not the mean-

centering μ parameter. As a direct comparison with DAP and ALE, ALE has a better performance with 48.5% classification accuracy compared with 40.5% from DAP.

As for the comparison between the three different proposed embeddings, ALE, HLE and WLE were compared in both datasets. In this experiment, ALE (AwA: 48.5%; CUB: 26.9%) perform better than HLE (AwA: 40.4%; CUB: 18.5%) and WLE (AwA: 32.5%; CUB: 16.8%) in both datasets. Additionally, the authors tested the concatenation of ALE and HLE embeddings, where the best results were obtained with 49.4% for AwA and 27.3% for CUB. Finally, the authors compared the ALE with WSABIE, OVR and GLE for few-shots learning, but since the main purpose of this section is to report on ZSL problems, please refer to the paper for the results.

The work presented by Akata et al. (2015) proposes a new approach called Structured Joint Embedding (SJE). The difference between SJE and the previously presented ALE algorithm is mainly on the optimization function, where the authors preferred the unregularized structured SVM as follows:

$$\frac{1}{N} \sum_{n=1}^N \arg \max_{y \in \mathcal{Y}} \{0, \ell(x_n, y_n, y)\} \quad (3.6)$$

where the loss function ℓ is the same as presented in ALE. For the optimization, again SGD was used, and the regularization is performed by early stopping when using the validation set of cross-validation. The authors also present an additional approach to that based on multiple output embeddings. The algorithm learns the best transformation W for a specific output embedding so, according to the given input embedding the best class is selected based on a confidence in each of the embeddings. As a certain output embedding can benefit more some classes than others, this approach uses multiple output embeddings and learns the best according to the provided input embedding. For this case, the authors, instead of using Equation 3.3 have updated the compatibility function as such:

$$F(x, y; W) = \sum_{k=1}^K \alpha_k \theta(x)' W_k \phi_k(y) \quad (3.7)$$

for K output embeddings and $\sum_{k=1}^K \alpha_k = 1$. As for embeddings, the authors used the input embeddings from a CNN presented in the DeVise (Frome et al. (2013)), Fisher Vectors (FV) used in ALE (Akata et al. (2013) and features extracted from GoogleLeNet (GOOG) Szegedy et al. (2015). From the output embeddings, the features used were human engineered attributes, hierarchical features and text corpora also used in ALE (Akata et al. (2013). The datasets used were the AwA, CUB and Standard Dogs (Dogs), where Dogs do not have human annotated attributes.

When comparing with the results from ALE with best of 49.4% for AwA and 27.3% for CUB, the results obtained in SJE are far better, where 66.7% for AwA and 50.1% for CUB with GOOG input embedding was achieved. As for the output embedding, the continuous version of human engineered attributes was used. The authors also analyzed the effect of learning from multiple embeddings at the same time with different input embedding transformation W . They tested two combination approaches: 1) Concatenation of embeddings by merging features from each one

(cnc); 2) Combination of embeddings (cmb) by using Equation 3.7. Since the authors are particularly interested in performing ZSL with output embeddings obtained from unsupervised learning, they tested the combination of embeddings using only unsupervised embeddings against the unsupervised plus supervised embeddings. Supervised embeddings are the attributes present in the datasets that were produced and annotated by humans.

The main conclusion is that attributes engineered by man significantly increase the performance of the system, where for AwA the best combination among only unsupervised embeddings achieved 60.1% and both unsupervised and supervised achieved 73.9%. As for the CUB, only with unsupervised embeddings the best performance was 29.9% and with both was 51.7%. This represents a better result than only using supervised embeddings as presented previously. The best performance obtained for the Dogs dataset using the unsupervised embeddings (human annotated attributes are not available) is 35.1%. In almost all experiments, the cmb approach performs better than cnc, but the difference is not very significant.

The approach presented by Xian et al. (2016) is called Latent embeddings (LatEm) and is a direct extension of the SJE where a nonlinear piece-wise compatibility function is explored, opposed to the linear one used in SJE. This nonlinear compatibility is explored by learning a collection of linear models, where each linear model maximizes the compatibility among image-class embedding pairs. Based on this, the new compatibility function proposed is as follows:

$$F(x, y; W) = \arg \max_{1 \leq i \leq K} \theta(x)' W_i \phi(y) \quad (3.8)$$

where K is the number of linear models to use and is a free parameter in the algorithm that needs to be optimized according to the problem in hands to achieve the best results. As can be seen, by linear model we refer the K matrices W that need to be learned. For the optimization routine, the same method as SJE where SGD is used. Hence, the authors present two different approaches to optimize such a parameter and select the best model: 1) Find the best K using a cross validation strategy by trying out 2, 4, 6, 8 and 10 linear models; and 2) pruning based strategy. As the first approach is relatively standard where all the models are tested with the presented values and the best is chosen, the latter proposes a more efficient approach while achieving similar results. The pruning approach starts with a large number of linear models and tracks the linear models that best suit both image-class pair. After this, a histogram is built with the number of times a linear model was chosen. After 5 rounds over the training data, the linear model is removed if it was chosen less than 5% of the times. As the authors refer, the intuition behind such an approach is that models that do not frequently maximize the compatibility between input and output embeddings are not of great importance and do not increase the performance, while increasing its complexity. The greatest benefit of the pruning approach when compared with cross validation is that only one model needs to be trained due to its adaptation during time to choose the best K value.

The datasets used are the same as the ones used in SJE (AwA, CUB and Dogs) so a direct comparison between the two approaches was made. In this work, only the GOOG input embeddings as presented in SJE was used since it was the one that produced the best results. For the

output embeddings, the same one was used, namely the human annotated attributes, text corpora and hierarchical features. By only using one type of embedding at a time and not making any sort of embeddings combination, the LatEm approach surpasses the SJE in all embeddings for all datasets, but for CUB with human annotated attributes. However, the best results are reported when the authors combine all the unsupervised embeddings with the supervised ones. This way, for the AwA dataset the performance of LatEm (SJE) was 76.1% (73.9%). For CUB the best result was still from SJE, where 47.4% from LatEm was not enough to surpass the 51.7%. However, for the Dogs dataset that only has unsupervised embeddings, LatEm performed better with 36.3% (vs 35.1%).

Some other interesting works were also proposed for the image classification problem in ZSL and worth mentioning, such as the joint latent similarity embedding (JLSE) (Zhang and Saligrama, 2016), the convex combination of semantic embeddings (CONSE) (Norouzi et al., 2014), the semantic similarity embedding (SSE) (Zhang and Saligrama, 2015), the embarrassingly simple approach to zero-shot learning (ESZSL) (Romera-Paredes and Torr, 2015), the synthesized classifiers (SYNC) (Changpinyo et al., 2016), the semantic autoencoder for zero-shot learning (SAE) (Kodirov et al., 2017), the simple exponential family framework (GFZSL) (Verma and Rai, 2017), the zero-shot classification with discriminative semantic representation learning (DSRL) (Ye and Guo, 2017), the feature generating networks (FGN) (Xian et al., 2018b) and the gaze embeddings (GE) for zero-shot image classification (Karessli et al., 2017). For a comprehensive survey of ZSL methods for image classification please refer to the work presented by (Xian et al., 2018a).

3.4 Related work in Zero-Shot Learning for others applications

One of the most interesting applications of ZSL outside image processing and classification domain is related with object identification using haptic devices presented by Abderrahmane et al. (2018). In their work, the authors use the DAP algorithm to recognize a set of objects by grasping those with a robotic hand with tactile fingertips. The main idea behind the ZSL setting is to be able to correctly recognize an object that the system was not trained for. Hence, from cutaneous and kinesthetic information of the robotic hand, the system should correctly say which object is holding, e.g. a plastic bottle, lamp or cup of tea, without any prior information about this specific object. For that, the authors used the attribute-based approach also presented in DAP, where, in this case, classes have associated a set of attributes that describe the object.

In order to test the proposed approach, the authors use a PHAC-2 dataset containing information about 60 different objects Y , with 24 annotated attributes A describing those objects and both haptic X_{b1} and kinesthetic X_{b2} information. Specifically, for the haptic information, data like temperature, heat flow and pressure were used, together with gripper aperture and height for the kinesthetics. The main idea is based on these sources of information, select the most probable attributes that describe the object, and from these attributes select the corresponding object. Therefore, since there were two different sources of information, the authors proposed two DAP-based methods to combine attribute classifications for each X_i . One of them is called MAXDAP and

simply selects the attribute with highest probability from both sources of information. The other method averages the attribute probability from both sources and from those, selects the highest. To train the classifier, an SVM was trained per attribute and the outputs transformed using a sigmoid function to provide a probability measure. The results show a classification accuracy 39% over 5 random splits with 50 objects in the train set and 10 on the test set.

Complementary to these results, the authors have setup an experiment with a real robotic hand with the same kind of sensing devices described before, where 20 objects were used together with 11 attributes that describe each object class. They have collected data from the robotic hand by grasping each of the objects 10 times in different positions for a period of 20 seconds to acquire information mainly on temperature. As for the train and test split, the authors randomly split the objects into two disjoint sets. The tests were performed using three different methods: 1) Local DAP (LDAP) where only one grasp was made; 2) Data-Fusion Multi-grasp DAP (DF-MDAP) where multiple grasps were performed and "super-grasp" was calculated based on the mean values from all grasps; 3) Similar Classification for Multi-grasp DAP (SC-MDAP) performs multiple grasps, and for each one gets a classification using LDAP until an object is classified with the same label k times. The results show that SC-MDAP is the best approach for this setting, followed by SC-DAP and then LDAP, leading to almost 100% accuracy in the test set with only 4 to 5 grasps. Despite the variety of methods tested in the present work, none of them represents a general ZSL framework that can be used for other applications, being only slight variations of DAP specific to the problem in hands of object recognition using haptic robotic hand.

One work that explores this task description topic in a different way where no data exists in the target task (both input and output) for a Reinforcement Learning scenario is presented by Sinapov et al. (2015). The main purpose of the authors is, based on task descriptors, learn the expected benefit that a certain source tasks will have on the learning process of a target task. For this case, a Reinforcement Learning method was chosen where an agent should learn the best strategy to win the Atari game of Pac-Man. Meta-data features were defined from each task to estimate its relatedness, where 192 variations of the task were used. The main idea is to select a target task, and the source task that most benefits the knowledge transferability should be chosen. The task descriptors are features specific from the game of Pac-Man such as the number of ghosts, ghost type, number of nodes in the maze, number of pills and so forth, totalling 15 features. As for the learning process, the authors used Markov Decision Process and used an Atari game simulator.

As previously stated, there is no information about the target task and still this transferability benefit should be calculated. For that case, the authors propose to learn a regression model $B(T_i, T_j)$, with a pair of tasks (T_i, T_j) , where the descriptions of each task (both source and target) are mapped into this transfer benefit. Hence, the expression used to calculate this relatedness is as follows:

$$x_k^{ij} = \frac{f_{i,k} - f_{j,k}}{\max(f_{i,k}, \epsilon)} \quad (3.9)$$

where $F_i = [f_{i,1}, f_{i,2}, \dots, f_{i,n}]$ and $F_j = [f_{j,1}, f_{j,2}, \dots, f_{j,n}]$ for a pair of tasks (T_i, T_j) . The goal of

such a regression model is to predict this relatedness and by trying out all possible source tasks, it is selected the one with highest value as such $T^* = \arg \max_{T_i} B(T_i, T_j)$. The authors also define a loss function where the difference between the optimal source task and the selected one is calculated $loss(T_i) = B(T^*, T_j) - B(T_i, T_j)$. Additionally, the authors define a ranking evaluation procedure based on the Discounted Cumulative Gain (DCG) (Järvelin and Kekäläinen, 2002) where the source tasks are ranked in decreasing order, and the DCG is calculated as such:

$$DCG_p(R_j) = rel_1 + \sum_{k=2}^p \frac{rel_k}{\log_2(k)} \quad (3.10)$$

where R_j is the ranked list of source tasks and $rel_k = B(T_k, T_j)$. A 10-fold cross validation strategy was used to evaluate the performance of the proposed approach, where 172 tasks were used to train and the remaining ones for testing. For the regression model, the authors tested Linear Regression (LR) against M5P Model Tree by calculating the loss and DCG. Linear regression performed the best and allowed to achieve faster convergence in the training procedure for the agent playing Pac-Man compared with the baseline where no transfer exists in which the source task was used as a “warm start” for further training. The authors also tested the approach of Multi-Stage Transfer by transferring knowledge from two source tasks, one after another. The main conclusion is that no better performance was achieved in transferring multiple tasks and ultimately selecting the best source task is key.

In line with the previous work in the sense that task descriptors are used, Isele et al. (2016) present a novel algorithm where the model parameters of a policy-based approach in a Reinforcement Learning setting is predicted based on a set of defined task descriptors. This work makes use of the same principle as Larochelle et al. (2008) and Pollak and Link (2016), but the methods used to achieve it are different. The main goal of the present work is to jointly learn a sparse encoding of both model parameters $\theta^{(t)}$ from a policy π_θ and task descriptors $\phi(m^{(t)})$ in an latent representation, where $m^{(t)}$ is the task description for task t . Hence, in order to learn this sparse encoding the authors defined the policy parameters as $\theta^{(t)} = Ls^{(t)}$ and the encoding of task descriptions as $\phi(m^{(t)}) = Ds^{(t)}$. Both L and D should be learned in order to reconstruct back the θ and $\phi(m^{(t)})$, where $s \in S$ should be a shared coefficient. To this joint learning the authors call *coupled dictionary learning* and to the whole algorithm Task Descriptors for Lifelong Learning (TaDeLL). The rationale behind such algorithm is that similar task descriptions have similar policies, so information can be learned from these two different spaces.

Therefore, the authors perform an adaptation to the Policy Gradient, first introduced by Sutton et al. (2000), where both L and D parameters are optimized.

$$\arg \min_{L,D,S} \frac{1}{T} \sum_t [-\mathcal{J}(\theta^{(t)}) + \rho \|\phi(m^{(t)}) - Ds^{(t)}\|_2^2 + \mu \|s^{(t)}\|_1] + \lambda (\|L\|_F^2 + \|D\|_F^2) \quad (3.11)$$

where $\mathcal{J}(\theta^{(t)})$ is the expected average return based on a set of trajectories from policy parameters θ , and ρ , μ and λ are the regularizers. It can be seen that the expression forces the values of

s to be zero, which is the principle of sparse coding algorithm, from the applied L_1 -norm. After the learning process of L and D , by giving a new task description t_{new} the algorithm is able to first estimate a sparse encoding $\tilde{s}^{(t_{new})}$ and then estimate the model parameters $\tilde{\theta}$ based on this sparse encoding.

The authors perform a set of tests in three different simulated environments: 1) Spring Mass Damper (SM); Cart Pole (CP); and 3) Bicycle (BK). For each of the domain 40 different tasks were tested, where another 20 tasks were used to tune the regularizers. The TaDeLL algorithm was compared with other algorithms, such as PG-ELLA, GO-MTL, single-task learning using PG. Additionally, the authors also tested a different version of TaDeLL called TaDeMTL where the learning is performed in an offline Multi-Task Learning fashion. From the tests performed, TaDeLL performed the best in all the testing scenarios, representing a promising approach for the RL field, and more specifically for Lifelong Learning. Ultimately, TaDeLL was compared with the previously presented work of Sinapov et al. (2015) in terms of time spent on computational performance. Since the proposed approach does not have to test all the possible source tasks to assess the best one to transfer, the processing time is drastically reduced.

Despite the presented approach being promising for RL problems, there is a major negative aspect that is important to highlight. The authors never state the values used for the task descriptors, which are also the conditions where the dynamical systems were executed. This kind of information is peremptory mainly due to experiment reproducibility. This way, it is difficult to compare the presented results with other algorithm implementations and can lead to future inconsistent and unfair conclusions about the approach.

Ultimately, the paper presents a general framework that is extensible to classification and regression problems by using their learning algorithm. As previously described, this kind of approach already explored first by Larochelle et al. (2008) and around the same time by Pollak and Link (2016), where a model of models was built making use of the model parameters applied to industrial scenarios.

In the next section, an holistic perspective will be taken from all the presented works and have a discussion about major contributions to the ZSL area. This perspective will make a comparison in terms of developed learning frameworks and the types of problems addressed, namely Classification and / or Regression. Additionally, an overview about the progression of the ZSL area in the Literature will be presented, focusing on number of publications and citations.

3.5 Discussion

As one could notice from reading the previous section, there are already a considerable amount of algorithms that address the area of ZSL. One of the main reasons is the sole idea of predicting correctly without previous knowledge. This central idea of being able to correctly classify an image without being trained to identify that class, or make an estimation of a target value from a task description, is one of the main captivating problems for machine learning. This kind of approach truly represents a significant step for AI field mainly due to restrictions about techniques

that were hand-cuffed to what they were trained to do, and no extrapolation or projection was possible outside the box called "training data". The presented methods are clearly a huge step towards more general and human like applications.

As a first comment to all detailed works and literature review, we should highlight the work that is considered the first to explore and define the ZSL area. We are talking about the work from Larochelle et al. (2008) that instead of ZSL, the authors used the term Zero Data Learning. This work was not only tied to the problems of character recognition in license plates, handwritten digits or molecular compounds, but also present two general frameworks that are applied to both classification and regression problems. It is based on the idea of Zero Data Learning that we consider that the ZSL topic made its first steps.

The second comment to all the works is related with the clear tendency towards the problem of image classification. Some other interesting works in different application areas were also reported, but the great majority of works are in fact related with image classification. In this context, we would like to emphasize the work performed by Zeynep Akata and her team from Max-Plank Institute, where new methods were introduced that could significantly improve the state of art. It was mainly due to the introduction of bi-linear compatibility functions that such an improvement was possible. Additionally, an effort was made by these researchers to standardize the topic of ZSL for image classification in a review work (Xian et al., 2018a) about evaluation protocols adopted for the ZSL problema and datasets used by the main methods proposed. Moreover, the authors evaluated these methods using the same evaluation protocols for a reliable comparison and proposed new datasets as benchmarks for further use by further researchers from the ZSL topic. This is a particularly useful work to perform in order to harmonize the developments being made and put side-by-side the major contributions to the topic so far.

As stated, the topic of ZSL is clearly biased towards the image classification field. As the name of the topic states, "Zero-Shot Learning", the central idea is to correctly classify or predict the label of a certain input sample, without being trained to do so. Naturally, the most suitable field is image classification because a "shot" in terms of classification is made without "zero" data from that class, and it is not a blind shot because there's the "learning" part involved. Despite image classification being the central topic of ZSL, as soon as promising results are published, the tendency is to apply these methods outside image classification. These steps already started to be taken in the works of Isele et al. (2016) and Abderrahmane et al. (2018) applied to policy estimation in reinforcement learning and haptic control for object identification. In the case of haptic control, the term "shot" in ZSL is still valid, but not so much for the policy estimation. In this case, the central idea is not to give a classification "shot" from a single input sample, but rather provide a new predictor for a task where no training data was available. This way, we should recall again the work of Larochelle et al. (2008) where the first term "Zero Data Learning" was used, which is more suitable to the policy estimation case. However, despite not being the central role of policy estimation to give a prediction based on a single input, we consider it should belong to the topic of ZSL. There are many opportunities to apply existing ZSL techniques, and develop new versions of those in multiple application areas. It is just the case of correctly formulate to problem

regarding the context of ZSL.

Additionally, another consideration should be made about general frameworks for ZSL contemplating both classification and regression problems. For classification, there are already a significant amount of existing works to address the problem of image processing, but not so much for regression or that contemplates both. Here we consider a natural situation to develop specific and tailor-made techniques for image classification because in most of the cases, in order to increase the performance of a certain system, a more application specific methods need to be built. It is difficult to design a general framework that can be applied to multiple problems and yet have state of the art results in most of these problems. Since the focus in the past few years has been on image classification, there was no need or interest in developing general frameworks that also work on regression problems. With some recent applications of ZSL to other areas, it is expected to assist the development of specific methods to address regression problems, like the one presented by Isele et al. (2016).

Table 3.1: Review table presenting the main methods for ZSL in Literature and their context in terms of Learning process and kind of Problem to address.

Method	Problem	
	Classification	Regression
Larochelle et al. (2008)	Image Label Prediction; Molecular Compounds	Applicable
SOC (2009)	fMRI	
CMT (2013)	Image Label Prediction	
DAP / IAP (2014)	Image Label Prediction	
JACF (2016)	Image Label Prediction	
DeViSe (2013)	Image Label Prediction	
ALE (2013)	Image Label Prediction	
SJE (2015)	Image Label Prediction	
LaTem (2016)	Image Label Prediction	
JLSE (2016)	Image Label Prediction	
DSRL (2017)	Image Label Prediction	
FGN (2018)	Image Label Prediction	
GE (2017)	Image Label Prediction with Human Gaze Embeddings	
HZSL (DAP) (2018)	Haptic Control	Applicable
TaDeLL (2016)	Lifelong Learning	
CONSE (2014)	Image Label Prediction	
SSE (2015)	Image Label Prediction	
ESZSL (2015)	Image Label Prediction	
SYNC (2016)	Image Label Prediction	
SAE (2017)	Image Label Prediction	
GFZSL (2017)	Image Label Prediction	

Based on this last consideration, Table 3.1 depicts an overview about the main algorithms presented in the previous section framed into its real application problem. Thus, we assessed these

methods in terms of applicability to either or both classification and regression problems. As previously stated, as soon as promising results start to be published in the context of image classification, it is normal to draw the attention from researchers working on regression based problems. Hence, since there are already some work developed in this context, we made an evaluation if, first, the classification techniques are general enough to be applied in regression, and second, if there are any specific method developed only for regression.

As main conclusions about this assessment made through Table 3.1, we first conclude. as already reported, that there's a tendency to develop new methods and algorithms for image classification, which was also motivated with the latest advances in Deep Learning research for image classification and generation. Secondly, there is a small amount of works that are general frameworks and could be applied to both classification and regression. Once again, it is difficult to develop a method that is general and still have a good performance in specific application areas, such as image processing. Last, but not the least, and up to our knowledge, we could state that there is not yet a specific method for ZSL that addresses regression problems. One of the reasons might be due to the lack of formal definition of ZSL for regression problems. As it exists in classification problems, regression requires a more well-established definition of 1) specific goals and purpose that ZSL is addressing and trying to solve and 2) the underlying conditions and assumptions for that problem.

Finally, regarding the application of ZSL to industry, up to our knowledge there are no works developed so far. Most of the works focus on using benchmark datasets to test their new approaches and increase state of the art performance, but not so much in applying those to industrial scenarios.

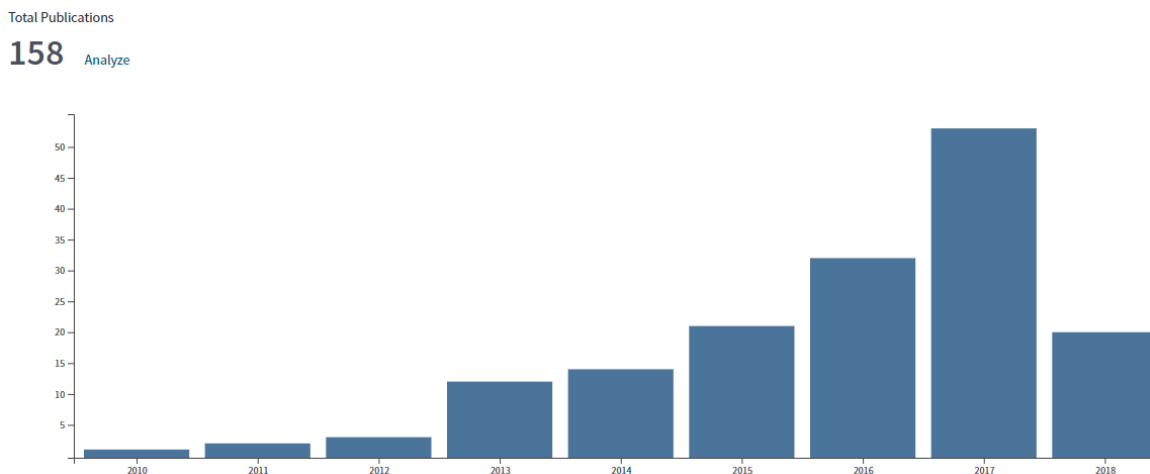


Figure 3.2: Publications for the ZSL area

A final assessment performed regarding the literature review is related with the number publications and citations in the area of ZSL. For that assessment, a search was performed using the Institute for Scientific Information (ISI) Web of Knowledge website that searches for scientific

works in the Web of Science platform. All the searches were made in the Web of Science Core Collection, that searches for indexed works in SCI-EXPANDED, SSCI, A&HCI, CPCI-S, CPCI-SSH, ESCI, CCR-EXPANDED, IC. The query used mainly contemplates the keywords "zero shot learning" and "zero data learning" that were searched mainly in the Title, Abstract and Keywords of the scientific publications. A timespan filter was applied to restrain the works prior to 2000 in order to avoid any misuse of the keywords in a different context. Based on this, the full query was defined as follows:

TITLE: (" zero shot learning " and " zero shot ") OR TITLE: (" zero data learning " and " zero data ") OR TOPIC: (" zero shot learning " and " zero shot ") OR TOPIC: (" zero data learning " and " zero data ") NOT TITLE: ("non-returnto-zero data" and "non-return-to-zero data") NOT TOPIC: ("non-returnto-zero data" and "non-return-to-zero data")

One could notice also that the keywords "non-returnto-zero data" and "non-return-to-zero data" were used to avoid the return of specific works. In literature, the term "non-return-to-zero" is used to refer to a specific topic in Telecommunications that has a significant amount of works. This was detected during all the assessment of works using the above query without the exclusion of this keyword.

This query returned a set of 158 publications, summing a total of 1,126 citations (672 without self-citation), with 583 citing articles (489 without self-citation), resulting in an h-index of 19. For a clear overview about the progress of the field in the past few years, Figures 3.2 and 3.3 present the number of publication and citations, correspondingly.

Sum of Times Cited per Year

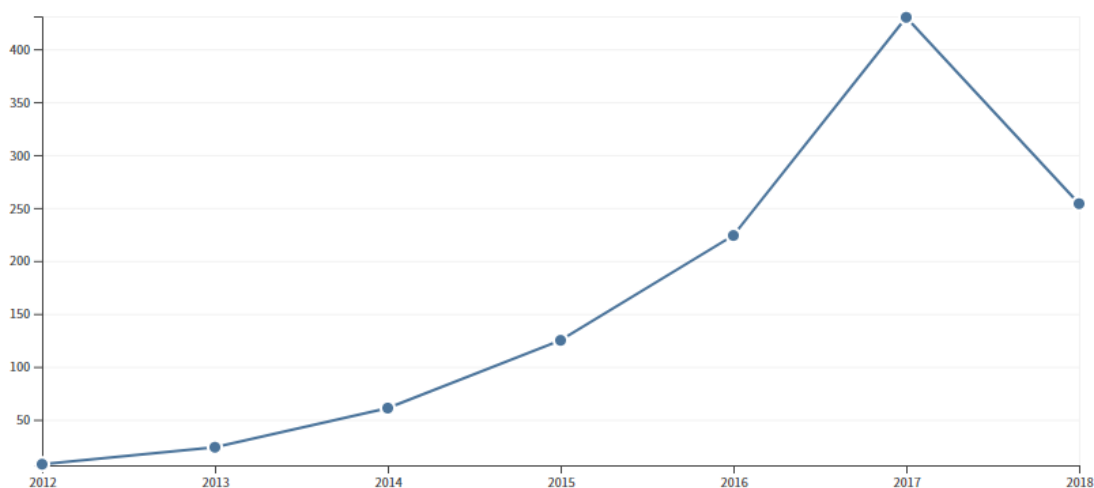


Figure 3.3: Citations for the ZSL area

As can be seen, in terms of publications, the growth is near to exponential since 2010, with 2 / 3 works published in the year of 2010 and, seven years later, the number of publications increased

to almost 60 only in 2017. Regarding the number of citations, from 2012 to 2017, the growth was clearly exponential, with 5 to 10 citations in 2012 and approximately 450 citations only in 2017. This supports our previous considerations about the growing interest in the field through the publication of good and promising results, which will probably lead to more research and pivotal publications in the area. If this behavior continues to persist, by 2020 the number of publications will be around 300 and the number of citations around 1,800, which is a tremendous growth in only 10 years of scientific research.

Chapter 4

Hyper-Process Modeling

So far, we have seen ZSL approaches that try to solve the problem of classifying new instances from classes that were not used in the training process. This means that a trained algorithm tries to correctly label a new instance from a class without being trained to do so. One of the key aspects to achieve a good performance is to have an additional feature space (often called latent space) for each class, where normally a meta-description is used, apart from input and output feature spaces. Is this property that inspired the development of the proposed approach.

Despite the good results achieved in the works presented in the previous section, none of them are either designed or applied to regression problems. As already explained, regression maps certain inputs into a set of continuous output variables, while in classification the output is either 0 or 1, or in a range between 0 and 1 such as in probabilistic models. Nevertheless, regression is used in these works to help, e.g. to map the inputs into a continuous latent space such as the weights of a linear classifier, as presented by Larochelle et al. (2008). However, the application of the problem is never regression. Most of the works are related with classification ranging from image classification to molecular compound matching or object recognition from haptic data. Here, we present an approach called Hyper-Process Model (HPM) that addresses the problem of ZSL for regression. Although, it should be stated that some of the works presented in Section 3.2 are general enough to be used in regression applications, but were not designed to take advantage on its inherent properties, such as output continuous variables.

For the remaining of this Section, we will first make a definition of the ZSL problem in a regression setting, and then describe all the methods used to build up the HPM. Finally, we will present and describe the proposed algorithm.

4.1 Problem Definition

As a first step, we would like to define the problem of ZSL to regression. Up to our knowledge, this is the first work that makes such a definition. Related with image classification, most of the ZSL techniques take advantage on the difference between input images, which is something normal where two different objects are displayed. Assuming inputs for a certain class / task as $X_i \in \mathcal{X}$

for class i , we can say that these techniques assume $P(X_i) \neq P(X_j)$ where marginal distributions among inputs from different classes are not the same. This means that the difference between the images can be learned to separate both from different classes. Contrary to this, for ZSL in regression problems, the inputs for different tasks could be the same and the responses might be different according to their specific task. For example, the amount of traffic in different parts of the city can be the same $P(X_i) = P(X_j)$, where i and j represent different parts of the city, but the air quality might be different due to different amounts of vegetation. If one part of the city has more vegetation, the air quality is higher, and vice versa. Most of the works first try to map the input into a latent space, which normally is a task descriptor, that can be generally expressed as follows $\mathcal{G} : X^n \rightarrow F^k$, where $X \in \mathbb{R}^n$ are inputs and $F \in \mathbb{R}^k$ are the task descriptors. In order to successfully learn the differences between tasks or classes, there should exist some difference between the task inputs like cubes and spheres, or cats and houses. Therefore, the assumption of $P(X_i) \neq P(X_j)$ is implicit in the context of image classification, which does not hold true for regression. Hence, this draws the first difference between how ZSL works for classification and regression, where it is not assumed that the marginal distribution of inputs from different tasks is different, and hence the proposed technique is applicable for problems where $P(X_i) = P(X_j)$.

Additionally, another key difference between ZSL for classification and regression is that multiple image classes are learned at the same time as a multi-task learning fashion. For the particular case of ZSL in classification, we have already seen from Section 3.2 that this learning normally occurs in two different steps: 1) Learning a mapping between inputs and task descriptions, and 2) Task descriptions into class labels. This means that one classifier should learn the differences between images and correctly predict the corresponding task description, and another classifier should handle the predicted task descriptions and correctly classify those into the desired class labels. Ultimately, the goal of ZSL in classification is to provide a new unseen image and correctly predict the label from a class not used in the learning process. Opposite to this idea, for the regression setting, the main idea is to build a whole new predictive function suitable for the new unseen task, where multiple inputs can be fed as a regular regressor. Therefore, for each source task, a regressor needs to be previously learned and together with the task description, a new function should be derived for a target task. The only work that uses the same approach is the already depicted technique called Model Space View presented by Larochelle et al. (2008). Additionally, the same principle was applied to solve a concrete problem in the area of manufacturing systems, a method called Hyper-Model (HM) Pollak and Link (2016) was used. Despite these techniques are in fact applicable for regression, the proposed approach overtakes some limitations of such approaches. These will be presented later in this section and will be highlighted and explained with a theoretical example.

In sum, we can define the ZSL for regression problem in the context of this work as the generation of a predictor that can be used in a new, unseen task, based on 1) task descriptions for both source and target tasks and 2) a set of predictors, one for each source task. Hence, we should define a task description as $c_i \in C$ for task i , where C are all the source task descriptions, and the predictors as $f_i \in F$, where F are a set of predictors. For the latter, we should define a function as

$f_i : X \rightarrow Y$, where X and Y are the input and output feature spaces, correspondingly. Additional to all of this, we should also define a function that maps the task descriptions into a latent space $\mathcal{L} : \mathcal{C} \rightarrow Z^p$, where Z is the latent space in a p -dimensional space. If each of the predictors of the source tasks has a set of trained parameters θ and all the predictors have the same number and corresponding parameters, this approach would be identical to the one presented by Larochelle et al. (2008) where Z^p represents the same as θ , so the parameters of the new function θ' would be predicted by $\mathcal{L}$ by providing the target task description c_t , where t is the target task. However, the key difference between the proposed approach and the one presented by Larochelle et al. (2008) is that the feature space Z^p is not a set of function parameters. This feature space is independent from the function parameters and in fact do not assume that parameters should have the same representation for all the functions used to model source tasks. For example, in order to learn a predictor that maps task descriptions into function parameters, one should choose the same type of machine learning technique to use, such as degree 2 polynomial, to train all the source tasks. For Larochelle et al. (2008) and Pollak and Link (2016), this implies that all source tasks will be trained using the same technique, not exploring the possibility of using the best machine learning technique for each source task. We interpret this as a limitation, where different tasks might have different complexities, and therefore certain types of functions might be more suitable to some tasks, and not to others. In the proposed approach we make use of a widely known technique from the computer vision area to address such a limitation and create a common feature space for different machine learning techniques.

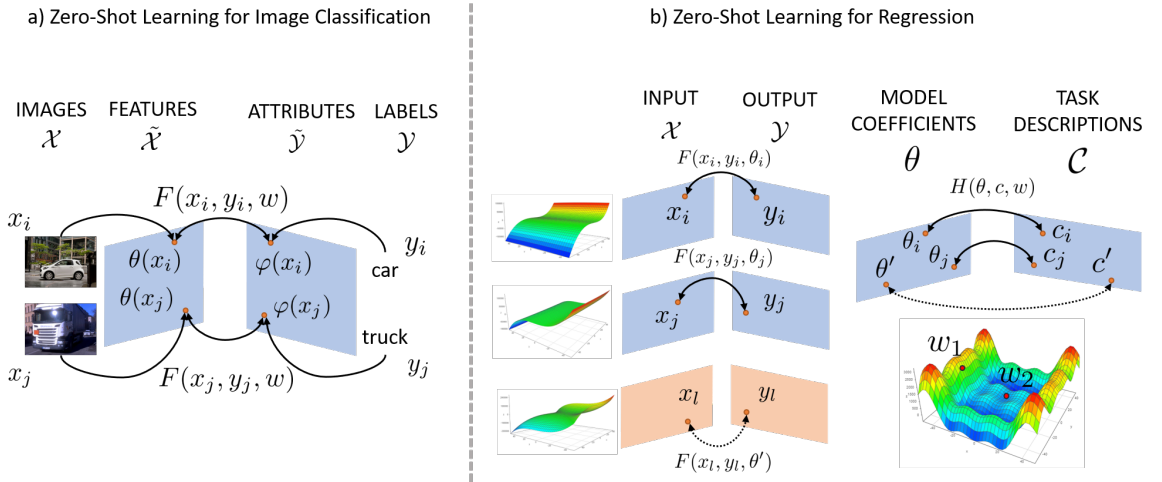


Figure 4.1: Comparison between ZSL for image classification and regression. a) Case where a latent representation for both images and classes is used, and a compatibility between these is learned (Akata et al. (2015)). b) Case where multiple models are used to learn a hyper-model that maps model coefficients θ into task descriptions $\mathcal{C}$. Upon new task descriptions c' , new model coefficients θ' can be estimated and a new model is created (Pollak and Link (2016)). This representation is also applicable in the model space view approach from Larochelle et al. (2008).

For a more complete explanation, Figure 4.1 makes a visual comparison between two approaches as a way to clearly make a distinction of between regression and classification for ZSL, in particular, to image classification. This way, on the left-hand side is a representation of the SJE approach (Akata et al., 2015) that makes use of two latent spaces, namely image embeddings and class embeddings as presented in Section 3.3. In this setting, the main idea is to present an unseen image from an unseen class during training, and correctly estimate its label.

Contrary to this, the goal of ZSL for regression is to estimate a new model by making use of an unseen task description and previous knowledge about already existing models. Particularly for the Hyper-Model approach (Pollak and Link, 2016) this learning is simply the mapping between coefficients and task descriptions of source models used to estimate the target model. This way, on the right-hand side two stages can be clearly seen. One is related with training the source models to derive the best models' coefficients θ for each task and the other is to train the hyper-model using those coefficients and existing task descriptions. Once a new task description is available, the model coefficients θ' can be estimated and a new function can be used $F(x_l, y_l, \theta')$, where x_l are the new input values that can be used to predict y_l (orange boxes on the bottom represent the new generated function). Hence, this visual separation allows to clear draw the main differences between classification and regression settings for ZSL, where one tries to label unseen instances in a class not used during training, and the other tries to estimate a whole new function based on previous acquired knowledge of existing functions and task descriptions. Particularly for the present thesis, instead of source model parameter space, a different kind of feature space will be used.

In the next subsection, the proposed approach will be presented taking into account the definition previous detailed, which is composed by the Hyper-Model concept proposed by Pollak and Link (2016) for process models in manufacturing applications, and the Statistical Shape Model (SSM) first proposed by Cootes et al. (1995) for image segmentation.

4.2 Proposed Approach

The main intention of the present section is to, first, clearly present the full algorithm of Hyper-Process Model (HPM) from the point of using the models trained with different techniques, to the final derivation of the new model to be used as a predictor in a new task. Secondly, it is intended to be reproducible for other researchers, where a step by step description of the algorithm is presented and explained. For that, most of the equations, notations and notions presented earlier are used, being the algorithm description just an organized way to present the approach.

Hence, Algorithm 1 presents all the steps required to implement the solution for different contexts of application. The first thing to notice is that the algorithm itself is divided into two different parts, being one of them oriented towards the shape analysis and the other to the hyper-modeling process. Based on this, the algorithm starts to introduce all the parameters necessary for its execution. As described, all the trained models are required along with the corresponding conditions (which are the task descriptions from ZSL). Moreover, the target condition is required

in order to generate the new model. Additionally, one should also specify the number of landmarks to use per shape, together with two more vectors that define the minimum and maximum values for the input features space. These minimum and maximum vectors are required so one could generate the input values to sample from the trained models. Finally, we assume to have m trained models to deal with.

For this algorithm, the SSM first comes into place because the Hyper-Model is dependent on the common representation of models to be trained. Hence, the first step (line 3) is to generate the input values X according to the minimum, maximum and number of intended landmarks per shape. Since we assume that no information can be drawn between the different inputs from the various models (as stated by $P(X_i) = P(X_j)$), the same input values are used for all the models. Therefore, the shapes S_i are built only considering the values from the output feature space, as presented in line 5, where i is a specific model.

Algorithm 1 Hyper-Process Modeling

- 1: **procedure** HPM($F, \zeta, \zeta', n, min, max$)(F is a set of source models, ζ is a set of conditions associated with each source model, ζ' is the target condition to be used for model generation, n is the of data points per shape, min and max are vectors of size r (assuming $X_i \in \mathbb{R}^r$) with minimum and maximum values for the shapes, correspondingly. Finally, m is the number of source models.)
 - 2: *Statistical Shape Model:*
 - 3: Define the input to sample from existing models: $X \leftarrow GenerateInput(min, max, n)$
 - 4: **for** $i = 1 \rightarrow m$ **do**
 - 5: Get shape: $S_i = f_i(X)$
 - 6: Get the mean shape: $\bar{S} = \frac{1}{N} \sum_{i=1}^N S_i$
 - 7: Get eigenvectors from PCA decomposition: $\phi \leftarrow PCA(S)$
 - 8: Get deformable parameters from PDM: $b = \phi^T (S - \bar{S})$
 - 9: *Hyper-Model:*
 - 10: Train the hyper-model: $h : b \rightarrow \zeta$
 - 11: Get the deformable parameter for new shape: $b' = h^{-1}(\zeta')$
 - 12: Get new shape: $S' = \bar{S} + \phi b'$
 - 13: Train a model for the new task. $f' : X \rightarrow S'$
 - 14: **return** f' .
-

The next step is to calculate the mean shape from all the generated shapes (line 6), where Equation 2.8 is used. In order to get all the eigenvectors to build the deformable model, a decomposition needs to be performed on all the generated shapes and PCA is applied (line 7). One should emphasize again that each shape is a vector of kn elements, where k is the number of features and n is the number of landmarks to use. Therefore, PCA is performed on a $m \times kn$ matrix S composed by all the shapes from source models, where these shapes are stacked in rows. Finally, the last step for the SSM is to derive all the deformable parameters for all the models (line 8). This way, Equation 2.15 is used. These are the parameters required to generate back the initial shape based solely on the deformable model. In order to get a good shape reconstruction the number of components chosen when performing PCA is critical, being a trade-off between reconstruction

and complexity. On one hand, if too little components are chosen, the greater the reconstruction error will be but less dimensions are required, and thus, less complex the problem is. On the other hand, if all the components are chosen, the reconstruction error will be minimum, but the complexity of the problem is far too great to deal with. In these situations, a good rule of thumb is to use the number of components (ordered by decreasing values of model variance) that attend for a cumulative sum of variance of at least 95%.

After building the deformable model, together with all the deformable parameters, the hyper-model is ready to be trained. For this case, and as presented by Pollak and Link (2016), one should train a hyper-model using any machine learning technique that seems suitable for the problem, by mapping deformable parameters into conditions. One might think at this stage that would be more suitable to map conditions into deformable parameters instead, because we can use the model to predict the parameters based on new conditions. However, in most of the cases, the dimension of the deformable parameters are greater than conditions, so the modeling needs to be made according to line 10. Only in the cases where 1) the dimension of parameters is the same or lower than conditions and 2) multiple models are trained where each one only has an output variable different from the others, the model can be trained as follows $h : \zeta \rightarrow b$. The implication of building a hyper-model that maps deformable parameters into conditions is visible in line 11, where the technique used needs to be invertible in order to get the new deformable parameters according to the specified new conditions. As an alternative, the level set where the model surface intercepts with the hyper-plane for the intended target condition can be calculated, as performed in the work of Pollak et al. (2011), or formulate a minimization problem where the distance between the predicted and target conditions should be minimized (Reis et al., 2017; Reis and Gonçalves, 2018b,a).

Once the deformable parameters are obtained from the hyper-model according to the target conditions, the next step is to generate a new shape based on Equation 2.14 as presented in line 12. The last step is to train a model to map the initially generated input values into the generated shape, which corresponds to the output values for that specific condition.

4.3 The Beta Distribution case

The main goal of the present section is to present a theoretical example for the use of HPM algorithm. For that, it would be ideal to have a set of simple functions with different properties, only with one input and one output so the algorithm could be well understood, and also different observed output values for the same input. On one hand, simple functions are suitable for this case because a visual feedback can be simply depicted, and on the other hand, with different types of functions the importance of the HPM can be better grasped.

As the name of the section indicates, for this scenario will be using a set of functions produced by the Beta Distribution. This distribution has two different parameters that by providing an input

$x \in \mathbb{R}$, a response $y \in \mathbb{R}$ is observed. By varying these parameters, the shape of functions will also vary. The expression for the Probability Distribution Function (PDF) is as follows:

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \quad (4.1)$$

where x is the 1-dimensional input, α and β are the parameters of the distribution and $B(\alpha, \beta)$ is defined as:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt \quad (4.2)$$

By using the Beta Distribution PDF it is possible to get a set of heterogeneous functions suitable to demonstrate a ZSL problem and apply the HPM to address it. For this case, 25 different functions were generated based on all combinations between α and β values of 0.5, 1, 5, 10 and 15. These set of values were chosen because they generate a set of 4 different types of functions. These are depicted in Figure 4.2 where 6 different graphs are shown and grouped by Hyperbolic and Linear functions (top row), Exponential functions (mid row) and Gaussian functions (bottom row). The goal for this example is to predict curves outside the 25 functions generated by the presented α and β values. Therefore, in the training phase only the curves generated by the 25 set of Beta Distribution parameters will be used, where in the test phase 16 different combinations of α and β parameters will be used in HPM algorithm to generate the corresponding curves. Here, the task descriptions will be the α and β parameters to train the hyper-model. The set of values for α and β parameters in the test phase are 4, 6, 8 and 12. In order to generate the curves, 20 input values between 0.01 and 0.99 were used to ensure a fair representation of each curve. As for the machine learning techniques used, polynomial regression was chosen with multiple degrees for training both source models and hyper-model in HPM algorithm. In this case, cross-validation was not an option due to the limited number of existing data, specifically 25 for training and 16 for testing. Additionally, as one can see, the chosen test curves from Beta Distribution are not biased to achieve a better performance on HPM, where all the α and β parameters lie between the parameters of training curves. Hence, we consider this experiment fair to compare both approaches.

In order to clearly highlight the advantages of HPM in a simple way, this approach will be compared with the one presented by Larochelle et al. (2008) and Pollak and Link (2016) where the coefficients of the base functions from the source models are used to train the model of models (hyper-model). As described earlier, we have stated the limitations of this approach by saying that the modeling technique needs to be the same for all the source tasks, where different complexities in the datasets might exist. This means that certain techniques might be better than others for specific datasets. Hence, we should call the approach from Pollak and Link (2016) simply as Hyper-Model (HM).

To highlight the differences between these approaches, Figure 4.3 shows the fit of two curve types from Beta Distribution. The dashed blue lines are the training curves and the orange lines are the corresponding fits. On the left side are the fits from a Polynomial degree 5 and on the

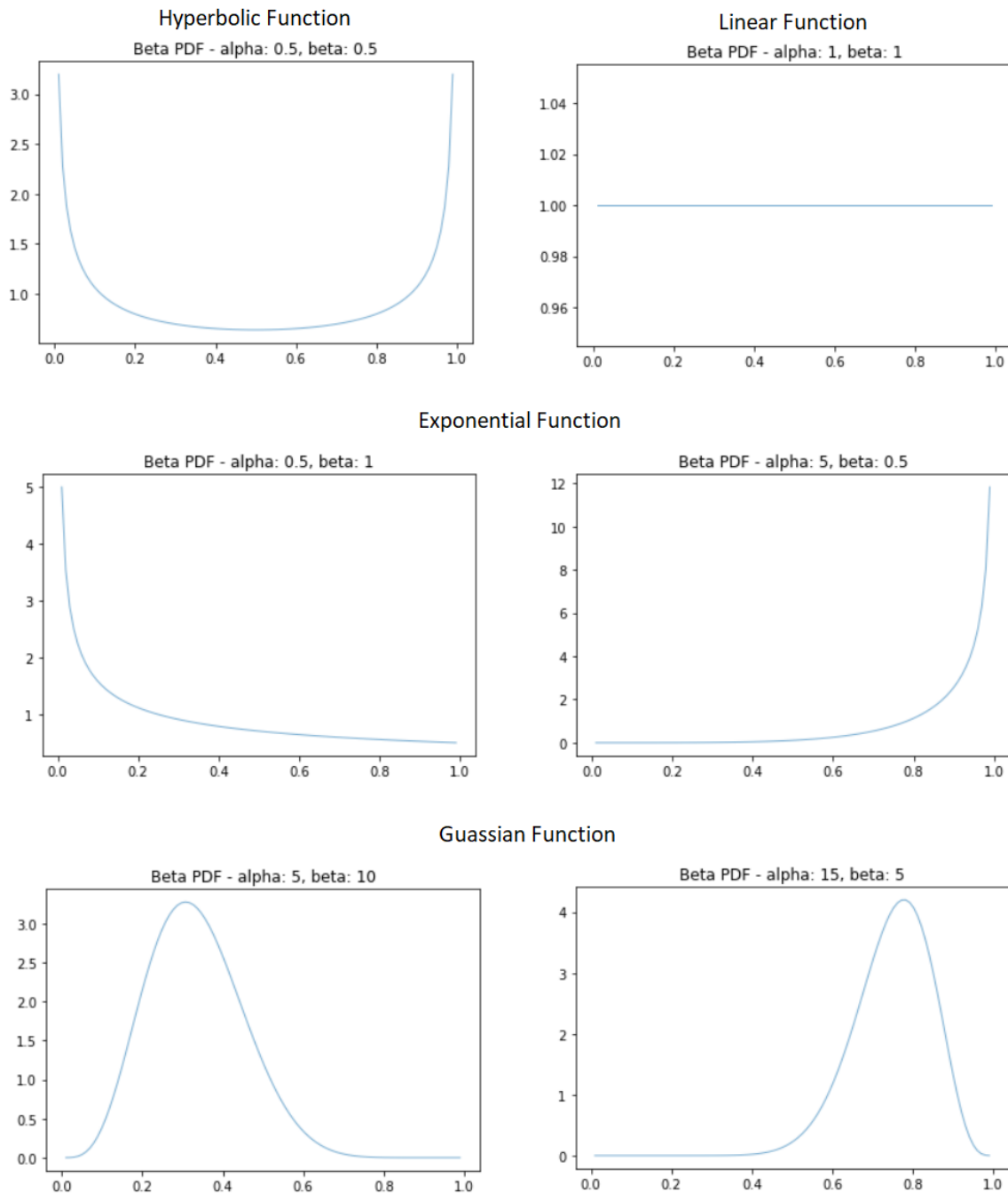


Figure 4.2: 4 different types of functions generated by the Beta Distribution. In the top row, an Hyperbolic function is depicted in the left handed side with α and β values of [0.5,0.5] together with a Linear function in the right handed side with α and β values of [1,1]. In the mid row, two Exponential functions are shown with α and β values of [0.5,1] and [5,0.5]. In the bottom row, two Gaussian functions were generated using the α and β values of [5,10] and [15,5].

right side the Exponential and Gaussian curve fitting results. The main idea is to clearly see that a technique that tries to fit all curves have a lower performance than the most suitable curve fitting

techniques for each task. As can be seen, on the left are the curves that the HM will use, and on the right side the shapes that the HPM will use to build the hyper-model.

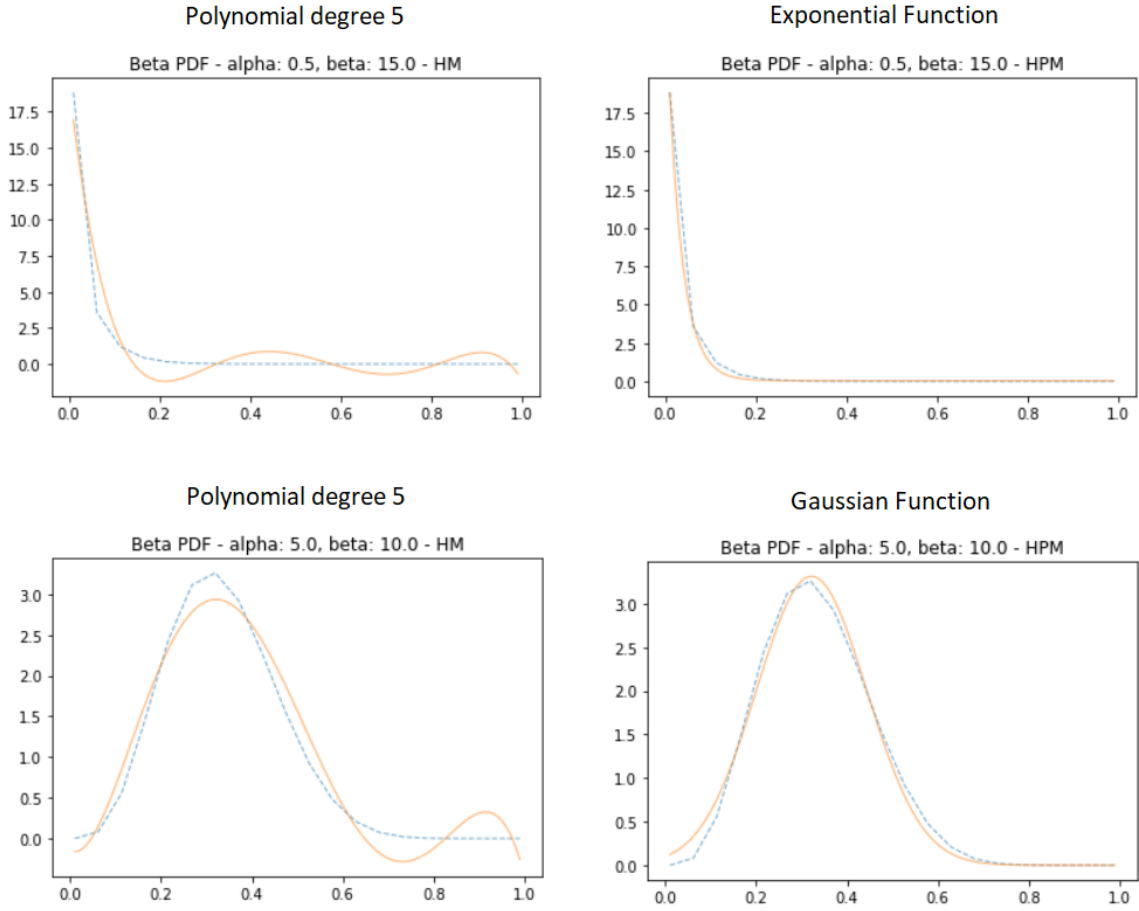


Figure 4.3: 4 different graphs are shown, where the left sided graphs are the fits for Polynomial degree 5 and the right sided graphs are the curve fits for Exponential and Gaussian functions.

At this point, in order to better differentiate between HM and HPM, we will briefly define its differences. For the HM, the models of models is trained by mapping model parameters into task description / conditions. Therefore, a function $h : \lambda \rightarrow \zeta$ should be trained, where λ are the model parameters from a single technique for all the models and ζ the task description. Contrary to this, the HPM approach uses deformable parameters from a deformable model proposed by the SSM concept based on a set of shapes, where a shape corresponds to each model. Hence, a function $h : b \rightarrow \zeta$ should be trained, where b are the deformable parameters and ζ the task description.

As stated before, the training phase for the hyper-model is composed by 25 parameters and task descriptions from the source tasks where machine learning techniques can be used to train models that fit these curves and are good generalizations. As in the HM approach, the same technique needs to be used for all tasks, so polynomial regression with varying degrees was chosen due to its flexibility to fit a large spectrum of curves. As for the HPM approach, Exponential, Gaussian

Table 4.1: HM approach: Mean MSE, Standard Deviation MSE and Hyper-Model R^2 for Number of Components from PCA decomposition of 3, 4, 5 and 6, and the degrees of 3, 4, 5 and 6 for Polynomial Regression of the Hyper-Model.

MODEL DEGREE	HYPER MODEL DEGREE	MEAN MSE	STANDARD DEV. MSE	HYPER-MODEL R^2
3	3	0.48	0.116	0.942
4	3	0.545	0.171	0.782
5	3	0.52	0.185	0.799
6	3	0.488	0.183	0.703
3	4	0.613	0.381	0.984
4	4	0.553	0.398	0.968
5	4	0.652	0.288	0.954
6	4	0.799	0.226	0.913
3	5	0.971	0.711	0.995
4	5	1.074	0.852	0.973
5	5	1.258	0.882	0.984
6	5	1.469	1.047	0.971
3	6	2.676	2.394	0.999
4	6	5.111	5.773	0.997
5	6	6.367	6.321	1
6	6	6.691	6.899	0.999

and Polynomial degree 7 were used to model these curves. By analyzing the shape of the curves, it is possible to choose the best technique to fit the data, representing a clear advantage for HPM.

For both approaches, different types of hyper-parameters were tested. Regarding HM, different degrees for the polynomial regression were chosen to model the source tasks, while for HPM the best models were trained according to data shape and different number of components were tested when performing PCA. In both cases, the degree and number of components tested were 3, 4, 5 and 6. As for training the hyper-model in HPM and HM, the polynomial regression was again used with degrees 3, 4, 5 and 6. For evaluation metrics, the mean of Mean Squared Error (MSE) for each of the test cases was calculated, together with the standard deviation of the MSE. This represents a total of 32 tests performed, 16 for each approach. To test the generalization capabilities of each approach, the generated curves from each approach were tested against the ground truth from the Beta Distribution with each curve containing 100 datapoints. As previously explained, for the training process only 20 datapoints per curve were used to train the source models. This way we could assess the performance of the proposed approach in a more broad, robust and generic scenario. For the HPM, 100 landmarks per shape were used to take advantage on the generalization capabilities of using the most suitable technique for each dataset according to its properties.

As presented in HPM and HPM2, and introduced by Pollak and Link (2016), the definition of the hyper-model states that it should be trained by mapping model parameters into conditions / task descriptions. This was first formulated as such because normally the number of parameters is greater than the number of conditions. However, as already discussed, it is a challenging problem to find an inverse for some of the techniques used in machine learning. As discussed, another way

Table 4.2: HPM approach: Mean MSE, Standard Deviation MSE and Hyper-Model R^2 for Number of Components from PCA decomposition of 3, 4, 5 and 6, and the degrees of 3, 4, 5 and 6 for Polynomial Regression of the Hyper-Model.

NUMBER COMPONENTS	HYPER-MODEL DEGREE	MEAN MSE	STANDARD DEV. MSE	HYPER-MODEL R^2
3	3	0.45	0.202	0.92
4	3	0.49	0.194	0.818
5	3	0.492	0.193	0.683
6	3	0.489	0.202	0.615
3	4	0.464	0.326	0.964
4	4	0.32	0.198	0.951
5	4	0.45	0.181	0.9
6	4	0.638	0.193	0.859
3	5	0.677	0.6	0.99
4	5	0.601	0.467	0.974
5	5	0.838	0.61	0.965
6	5	1.173	0.896	0.962
3	6	2.208	1.88	0.997
4	6	3.473	3.313	0.996
5	6	5.127	4.672	0.996
6	6	6.276	6.323	0.994

to handle such a problem, is to use a search algorithm and find the optimal or near-optimal parameters that minimizes the distance between predicted and target conditions. As for this example, we would like to avoid both problems, and directly map conditions into model parameters. For this case of Beta Distribution multiple models are trained where each maps all conditions into only one model parameter. This implies that the number of models trained is the same as the output feature space dimension. Hence, the hyper-model is a composition of multiple ones, where each model parameter is predicted independently from the remaining. This should not be mistaken as Ensemble regression, where multiple weak learners are trained with the same input and output features, and then the outputs of all learners are combined to produce a final prediction.

As for the results of both approaches, Tables 4.1 and 4.2 present all the 16 tests performed per approach, where in bold font the best result is depicted. In this case, the best result should be considered as the minimum value for Mean MSE for all 16 tests performed in the test set. Complementarily to the evaluation metrics, also the Coefficient of Correlation R^2 for the hyper-model is presented, so the reader can have a clear idea of its performance depending on the polynomial regression degree. Just before commenting the achieved results, one should highlight that for HM the number of models parameters used to train the hyper-model is the value *Model Degree* in Table 4.1 plus 1 because of the bias term used for training. On the contrary, the *Number of Components* in Table 4.2 is exactly the number of parameters used to train the hyper-model.

As for the results, the first thing to refer is that the best set of parameters for HPM performs better than the ones in HP, with a Mean MSE of 0.32 for 4 components and hyper-model polynomial degree 4 and 0.48 for model polynomial degree 3 and hyper-model polynomial degree

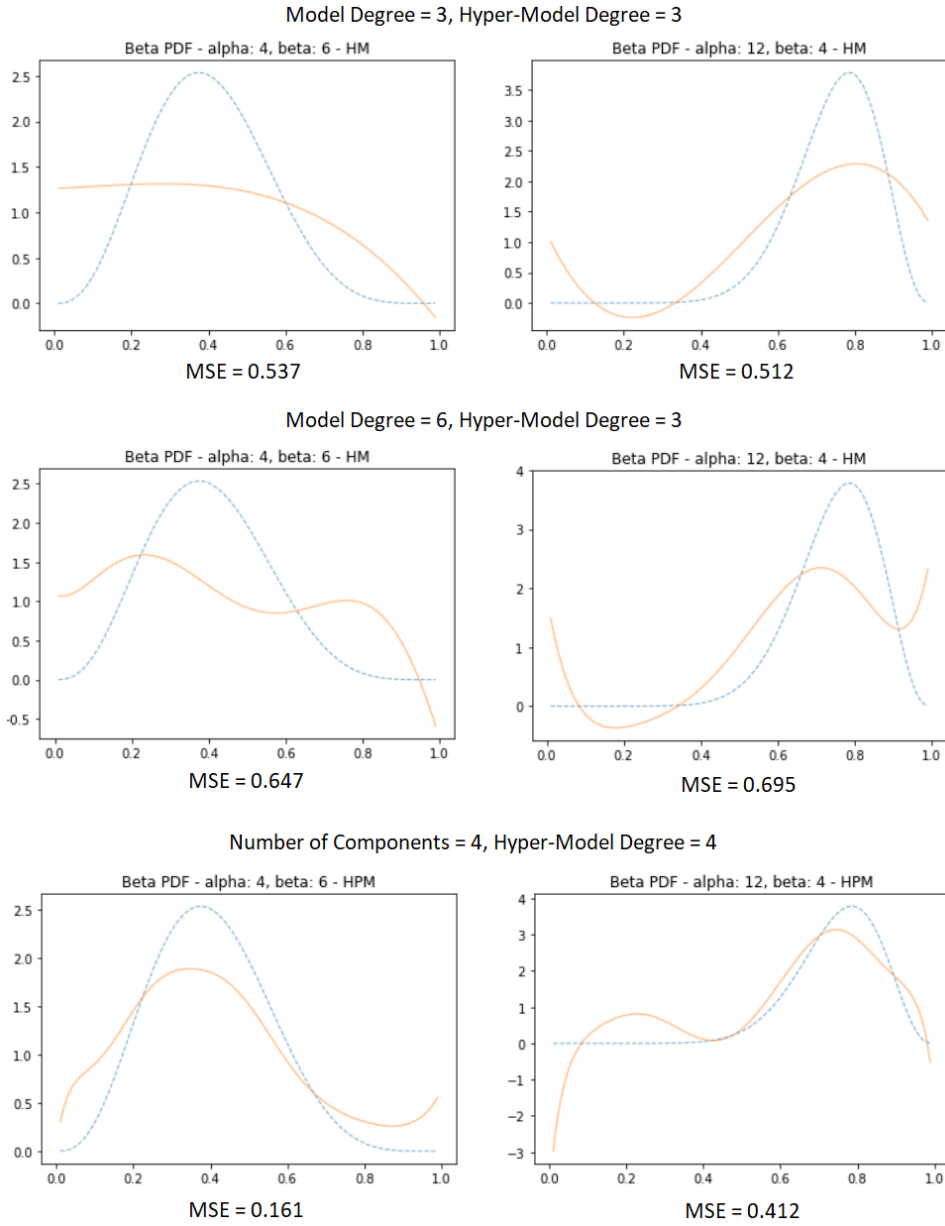


Figure 4.4: Graph plots of two test curves, where the dashed blue line is the ground truth from Beta Distribution and the orange line is the result of the ZSL approach applied. In the left column are the results for Beta Distribution with $\alpha = 4$ and $\beta = 6$, and in the right column the results for $\alpha = 12$ and $\beta = 4$. The first row corresponds to the HM approach with model degree of 3 and hyper-model degree of 3, to mid row refers also to HM approach but with model degree 6 and same hyper-model degree, and the bottom row is the HPM with 4 components and hyper-model degree of 4.

3 (which is very similar to Mean MSE of model degree 6 and hyper-model degree of 3), correspondingly. This supports our hypothesis that by using the SSM concept that can take advantage on different techniques by dealing with shapes instead of model parameters, a better performance

can be achieved when comparing with HM. The second thing to notice is that by making a direct comparison between each of the tests from both approaches where the model degree and number of components are the same together with the hyper-model degree (we might call it as the same setting), for all tests the HPM performs better. This means that the HPM is more effective than HM for the same problem. Additionally, as previously explained, the models parameters to use in HM to train the source models corresponds to the model degree plus 1, so for the same setting, the HPM is more efficient because it uses a lower dimension for the problem.

For a better interpretation of the above results, Figure 4.4 presents two test scenarios for the Beta Distribution with $\alpha = 4$ and $\beta = 6$ on the left and $\alpha = 12$ and $\beta = 4$ on the right, where the dashed blue line is the ground truth from Beta Distribution and the orange line is the result of the ZSL approach applied. In each row, different settings were used: On the top, HM was used with model degree of 3 and hyper-model degree of 3; On the mid row, again the HM approach was used with model degree of 6 and hyper-model degree of 3; On the bottom row, the HPM approach used tested with 4 components and hyper-model degree of 4. Additionally, below each image the MSE is depicted to assess the distance between the predicted curve and the ground truth. As can be seen, the best results are yielded by the HPM approach and, for the HM, it can also be seen that the MSE increased as the model degree increased from 3 to 6. It can be clearly seen the effect of increasing the polynomial degree used to fit the source models as in the mid row the predicted curves are more complex and irregular than the ones in the top row. Despite being more complex, they do not produce better results and are not a better suit for the presented ground truth.

Hence, for the Beta Distribution scenario we have shown that HPM, for the proposed conditions, is, first, more effective than HM due to better performance in all the corresponding settings / tests performed, and second, because it addresses the same problem with less complexity than the HM approach. This is what we were referring previously as by taking advantage on the regression setting, the performance of ZSL techniques could be improved.

4.4 Practical Relevance

In the previous section, the HPM algorithm was depicted in a more abstract context where a detailed description was made and explored in a theoretical case. Although this description is more interesting in a theoretical context, in this section we would like to highlight some of the interesting applicabilities and show the potential for further exploration in a more practical context. We will first discuss the importance and benefit of such algorithms (both HPM and HPM2) in the context of manufacturing systems where multiple machines and processes for the same machine exist. Secondly, a simple example is presented so a better understanding could be grasped by the readers, and once again, start seeing the potential of the proposed approach.

Normally, the operation of a certain machine is guided by a set of process parameters and process quality that dictate the final result of a certain product. The process parameters are just a set of values that the operator can introduce in the machine that will lead to different machine responses. These will influence the final result for a certain product (e.g. a shape of metal piece),

where a certain deviation from customer specifications is calculated, and ultimately, the quality is assessed. In order to achieve that, the correct process parameters that would yield the optimal process quality need to be chosen for specific process conditions. These conditions are the ones defined by Pollak and Link (2016) and are fixed quantities that govern a certain industrial process. These conditions can be metal sheet thicknesses, room temperature, humidity, or other conditions that are not controllable by the operator in the machine parameters, but still influence the end result.

One of the responsibilities of a machine operator is to explore how the parameters impact the final quality by performing a set of experiments, where no system that automatically performs these experiments is used. Hence, the strategy used is based on trial and error, and once the optimal machine parameters are found, the machine is ready for production. If all these experimental findings are well documented, a dataset can be built and machine learning techniques can be used to train a process model. A process model is defined by a simplified version of the machine dynamics where parameters are mapped into quality. These process models can greatly guide and assist shop-floor operators to perform experiments in a more informed way by providing an initial guess of parameters according to the desired end quality of a product.

In a more practical manner, assuming that a machine can produce different types of products, such as using different sizes and types of materials, it would be useful to have some sort of way to assist the operator once a new unseen product needs to be produced in that same machine. As already explained, normally the operator should perform a set of experiments until it finds the correct set of parameters that would yield the best quality. However, it is difficult to know which parameters use first, and once performing the experiment, what parameters use next. Often, this kind of decisions are based on prior acquired operator knowledge about that machine or other machines with similar physicochemical dynamics. If the operator does not have such experience, random trial and error experiments are performed, and long times of calibration are required because normally machines have a high number of possible parameter combinations to explore.

Defining the possible set of products to yield in a machine by processes, where one product is a specific process for that machine, and m a specific process from a pool of M processes, a model that maps parameters into quality is called process model and it can be expressed as $z_m = f_m(x_m)$, where z_m is the process quality and x_m are the machine parameters for a specific condition. Additionally, one should also define a condition as $c_m \in C$, where these can be different material properties, dimensions, topology and so forth. In this context, a ZSL problem can be defined as deriving a process model f' for a new product variant, where 1) process models f_m for different product variants are available, together with the corresponding conditions C and 2) the condition for the new product variant c' . Since all the processes are yielded in the same machine, the input feature space for each process x_m is the same, and therefore, no information can be learned from these. For this case of predicting a process model based on a new process condition, the HPM algorithm can be used. By building a common feature space between all processes using the SSM concept, and from all conditions C , a hyper-model can be trained. Hence, a new solution in this common feature space that represents deformable parameters in a deformable model can be derived, and

a new shape generated. Finally, this will be used to train the new process model f' . In order to assist the operator with an initial guess of what machine parameters to use, a minimization problem can be defined where a search algorithm can be used to find a solution (machine parameters) by specifying a target process quality defined by the customer, as already explained.

In the context of the multiple processes, the HPM is able to interpolate among these because there's an implicit relation between the processes. This relation naturally exists because all these processes belong to the same machine, so the physicochemical phenomena is the same but for different conditions. However, in a manufacturing company there are different types of machines and it would be also useful if the same principle could be applied to different machines. If these machines have the same type of parameters (same feature space) $X_i = X_j$, but with different range of values $P(X_i) \neq P(X_j)$, one could use the approach proposed by HPM2 to address such a problem. Normally, in order for different machines to have the same type of parameters, these are scaled versions of a specific machine, like different sizes of a press machine or welding gun. The parameters remain the same but only with different ranges, like higher press forces or higher voltages for welding. Contrary to the previous scenario, information can be acquired by also considering the process parameters in the ZSL problem, as proposed by HPM2. Now, the usefulness of this approach is related with the integration of new machines in the shop-floor, where a new process model can be derived and used to accelerate the ramp-up phase process and start production sooner.

If one intends to apply these approaches to machines that have different types of parameters $X_i \neq X_j$, they will not work properly. This is similar to the case when process models are trained using different machine learning techniques and model parameters do not have the same representation. Hence, for different machine parameters also a common representation should be found, and only then the HPM2 algorithm be used. However, this common representation for different machines is a much harder problem because each parameter influences differently a physicochemical phenomena, and similarities between the processes phenomena needs to be derived. This last case is also out of scope for the present thesis.

Chapter 5

Process Assessment

The main objective of the present chapter is to assess the proposed methodology in two practical applications. This way is possible to correctly understand the context where the HMP can be used and what are the real benefits of using such an approach. For that, two different scenarios are presented: 1) Deep Drawing; 2) Laser Seam Welding. The first scenario is related with a sheet metal forming process where a metal sheet blank is shaped according to a forming die by performing a mechanical punch. This way is possible to shape objects such as cans, boxes, bottles and basically any irregular form that is possible to produce with mechanical punching. The Laser Seam Welding process is composed by two major components - Robotic Arm and Laser Mounted Head - and aims to weld two metal sheets using laser technology. Contrary to the typical Resistance Spot Welding process where a set of ring shaped welds are made, the Laser Seam Welding joins the metal sheets by creating a continuous line of melted material, yielding more robust and reliable products.

Based on this, this chapter will start to define an experimental setup that is common to both scenarios ensuring that all the techniques follow the same approach and in the end they are fairly compared with each other. Only afterwards the suitability of the proposed methodology will be tested and assessed in the proposed scenarios. For each one, a two-part process will be predominantly described. The first is training the process models based on experimental data using ANNs and the second is to apply the HPM algorithm to generate new process models based on existing ones. This two-part process simulates what could happen in the shop-floor and contemplates all the steps from data collection using a Full Factorial Design approach to the use of the generated process models.

5.1 Experimental Setup

Since the proposed approach is based on machine learning techniques, a reliable training process needs to be defined in order to assure the most trustworthy results. As presented in Chapter 2, one of the main concerns in the training process is to guarantee the generality, avoiding both underfitting and overfitting. One of the way to evaluate the presence of these effects, is by specifying

a well defined and widely used metric. The metric used is the Mean Squared Error (MSE) and is defined by the following expression:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (5.1)$$

where n are the number of datapoints, Y_i is the output for the i th datapoint, $\hat{Y}_i$ is the i th prediction.

The main advantage of monitoring both training and validation sets is to avoid overfitting the training data, and consequently, losing model generality. Particularly for ANNs where the number of epochs should be specified, it is particularly useful to monitor the training evolution. This evolution is an indication if the model is converging towards a suitable state and learning is being achieved. If not, normally both training and validation sets do not decrease the MSE and get stuck in a high error value. So, in the case of overfitting during the training procedure, normally the MSE for training set continues to decrease but the validation MSE stops decreasing or even starts increasing (Prechelt, 1998). It is when validation set reaches the lowest value that the model achieves its best generalization capability. Figure 5.1 presents an idealized curve shape for both training and validation MSE, where the vertical axis represents the error and the horizontal axis the number of epochs in an ANN. As can be seen, in order to get the best performance out of an ANN, normally an early stopping is required at the point where the validation error starts to increase. Based on this, there are two ways to achieve this: 1) Early Stopping; 2) Model Check Point. In the first approach, the idea is to set a high number of training iterations / epochs and if the MSE for the validation set does not improve during a certain amount of iterations, the training process is interrupted. Although it seems a good approach, one of the biggest problems of this approach is to set a suitable number of iterations to check if the model does not improve, being one more hyperparameter to optimize for modeling. When dealing with non linear datasets, and also due to random initialization of ANNs, the model could take same time until converging to a suitable state yielding low MSE. In this case, if low number of iterations for early stopping is defined, the model could have not yet converged. If this number is too high, the model starts to overfit the data. Similarly to the first approach, the Model Check Point also sets a high value for training iterations, but the best model considering the validation set is always saved. Hence, this allows to have a model in a generality sweet spot, right between underfitting and overfitting. Therefore, for this experimental setup, the model check point approach was used.

Additionally, as described in the HPM algorithm, when generating shapes from process models, the number of landmarks should be specified. Although, this is not sufficient and a strategy to select the best input values from all the parameters is required. For that purpose, and to simulate how normally a manufacturing company designs its experiments at the shop-floor, the full factorial design method will be used. Factorial designs are experiments to be performed in manufacturing equipment where a range of values of the process parameters are combined in other to grasp the distribution of the observations made. The experimental design consists of making an observation at each of a large set of possible combination values from these parameter. Each of these values

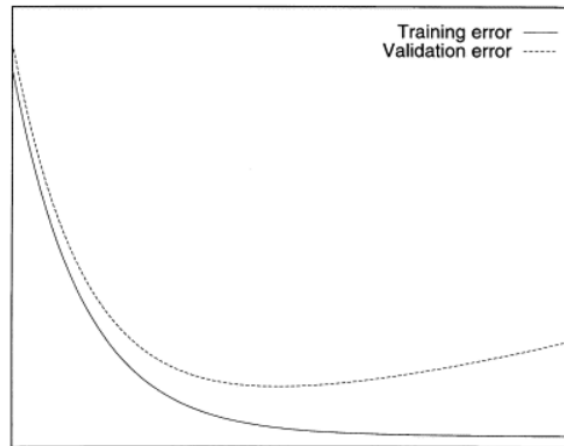


Figure 5.1: Idealized training and generalization error curves. The vertical axis represents the error and the horizontal axis the number of epochs in an ANN. (Prechelt, 1998)

are called levels, and the parameters to be tested are called factors. Every combination tested in the machine is an experiment.

5.2 Hyper-Process Model using Deep Drawing process

The main purpose of the present section is to make use of the HPM algorithm to, first, generate a new model upon new conditions and second, be able to predict the final quality of the product. To this intent, a simulated scenario with a high number of models are used in the context of the deep drawing process. Hence, a description of this process will be made, followed by the process modeling phase where multiple source models are trained using ANNs, and finally the generation of unseen models using solely the new conditions.

The deep drawing process is a widely used manufacturing procedure that aims at forming a metal sheet blank into a cup or box-like shapes using a press machine. This way is possible to form any flat metal sheet blank into the desired shape, which occurs under a combination of tensile and compressive conditions leading ultimately to the manufacture of light weight and low density products. For that, a great knowledge of the process is required since a high number of parameters can be optimized, from blank-holder force / pressure, punch radius, die radius, material properties just as elasticity, to coefficient of friction used (Dwivedi and Agnihotri, 2017). Pots and pans for cooking, containers, sinks, automobile parts, such as panels and gas tanks, are just some examples of the products manufactured by sheet metal deep drawing (Library of Manufacturing).

As can be seen in Figure 5.2, this process is mainly composed by 4 main components: blank (light gray) - metal-sheet to form; punch (dark gray) - tool that forces the forming of the metal-sheet and molds it; die (black) - base plate that supports the metal-sheet during forming; blank-holder (purple) - applies pressure in the metal-sheet extremities against the die. Normally, the process is composed by three main steps. The first is the initial setup of the machine where the

metal sheet is flat and the force is applied in its extremities to hold the sheet and guide the flow of the material. This force is made from a pressure pad against the die. The second step is mainly composed by continuously applying a punch force downwards to mold the metal sheet into the desired final shape. Finally, the last step is when the punch stops applying pressure and the final shape is achieved.

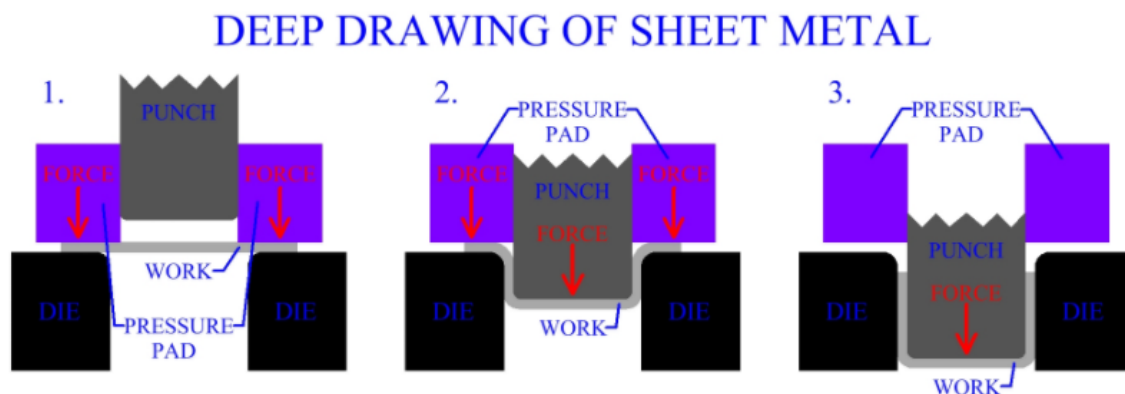


Figure 5.2: Exemplification of a deep drawing process in 3 main steps: 1) First position of the piece; 2) Punch force to form the flat metal sheet; 3) Final positioning and end of the process (Library of Manufacturing)

One of the most important parameters to measure in the deep drawing process is the maximum stress that influences the final quality of the product. This is a critical aspect to assess, e.g. in the aerosol industry where can-shaped products will be subject to high pressures like in beverages or paint sprays. If this can is not well produced and is subject to high stresses, it can leak during the filling process, or even worse, during the customer usage in its daily life routines.

5.2.1 Scenario Description

The scenario explored in the present thesis is related with the generation of process models that map a set of process parameter values into a set of product quality measures. This way, in order to better assess the proposed HPM algorithm, a considerable number of process models need to be trained, together with the corresponding process conditions that explicitly describe a certain process. To that intent, a simulator called Abaqus was used in order to prepare a dataset that describe the deep drawing process, where a combination of multiple process parameters and conditions was used.

Abaqus is a simulator for finite element analysis (FEA) and computer aided engineering (CAD / CAM). For this case, the "Complete Abaqus Environment" or Abaqus/CAE version 6.14 was used where a simulation of mechanical component models can be executed and an analysis of its performance can be done. For the present work, a 2D simulation was used with a metal sheet and three more components that allow to shape the sheet. These components are namely the punch,

pressure pad and die according to the deep drawing description in the previous section. As can be seen from Figure 5.3, on the left side is the initial position of the process where the metal sheet is flat and the pressure pad is applying force to the sheet against the die. On the right side is the final position of the process, where the final shape is achieved and the sheet molded. Additionally, the points of stress can be seen in the final shaped sheet, where the blue regions represent the lowest points of stress and the red / orange represent the highest points of stress. As can be seen, the sheet part that was molded is the one where the stress have increased, with a green and orange color. In this case, blue represents that the sheet was not subject to any pressure, green represents acceptable and secure values of pressure required for molding, and orange / red colors represent dangerous and undesired pressures that result in fragile product segments and could lead to further leaks or unintentional bends. In this context, high values of stress indicate fragile regions where an increased pressure can deform the metal sheet or even make a hole to release the pressure, e.g. from inside a beverage can.

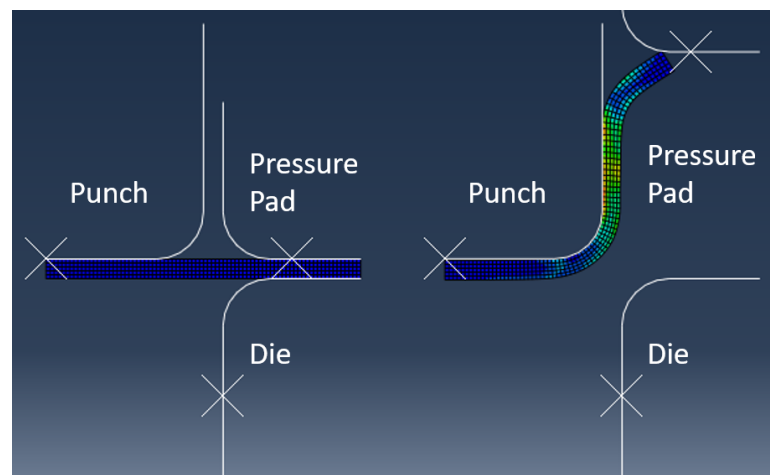


Figure 5.3: Abaqus 2D plot interface. On the left-hand side, the initial positioning of the steel sheet, and on the right-hand the final drawing of the steel sheet.

As for the process parameters and quality explored in this scenario, there are two parameters that can be changed that yield different quality in the final product for the same conditions. These parameters are the blank holder force (BHF) - force applied by the pressure pad into the metal sheet against the die - and friction (F) - amount of friction in the die that allows the metal sheet to move more smoothly or roughly. A minimum value for BHF leads to wrinkling in the metal-sheet, while a maximum value for BHF leads to fracture where the blank can break. Additionally, the lack of friction between the blank sheet and both die and blank holder can lead to earring on top of the formed piece. This way, it is very important to optimize these parameters during a calibration phase whenever a new piece needs to be yielded. As for the process quality, we have assessed two different measures. One of them is the maximum stress (MS) observed during the whole process and the other one is the distance (D) of the metal sheet from the initial to the final positions. This

means that the process models to be trained will map process parameters in process quality, so we can predict the maximum stress and metal sheet distance according to the parameters used, for specific process conditions.

Complementary, we have defined three process conditions that change the relation between process parameters and quality. These are the metal sheet thickness (T), initial stress (IS) and saturation (S). As for the IS and S, these are parameters used in the plasticity model during Abaqus simulation of the process, where the isotropic hardening law of Hockett-Sherby was used (Hockett and Sherby, 1975). Therefore, the IS is the parameter for initial value of the yield stress while S is the flow stress saturation value. Assuming that a machine learning algorithm is used to map process parameters into quality, different values of process conditions imply a different data distributions, and hence a new predictor needs to be trained. From Table 5.1 all the combination of deep drawing conditions are depicted, where 18 processes will be assessed to be used the proposed algorithm.

Table 5.1: Process Conditions for the Deep-Drawing process simulation in Abaqus

	Metal Sheet Thickness (T)	Initial Stress (IS)	Saturation (S)
Process 1	1.5	100	130
Process 2	1.5	100	165
Process 3	1.5	100	200
Process 4	1.5	175	227.5
Process 5	1.5	175	288.75
Process 6	1.5	175	355
Process 7	1.5	250	325
Process 8	1.5	250	412.5
Process 9	1.5	250	500
Process 10	2	100	130
Process 11	2	100	165
Process 12	2	100	200
Process 13	2	175	227.5
Process 14	2	175	288.75
Process 15	2	175	355
Process 16	2	250	325
Process 17	2	250	412.5
Process 18	2	250	500

In order to obtain the 18 models, the Abaqus Simulator was used with each of the process conditions and a set of values for process parameters BHF and F. For each process condition, 6 different values for BHF were used: 0.066, 0.1, 0.133, 0.167, 0.2 and 0.23; as for F, 9 different values were used: 0.12, 0.23, 0.34, 0.45, 0.56, 0.67, 0.78, 0.89 and 1. This means that a dataset of $6 \times 9 = 54$ datapoints per process condition was created. Taking in mind that each datapoint is a single simulation in Abaqus, it was required a total of $54 \times 18 = 972$ simulations to create the ground truth and baseline for the current scenario. The values for BHF and F were chosen based

on empirically evaluating the maximum of the process parameters where the maximum stress was so high resulting in undesired deformations of the metal sheet or a fracture.

All these 18 process conditions imply different relations between process parameters and quality. This means that a set of process conditions (T, IS and S) is a high level description for one process, where multiple process parameters have a unique mapping into a set of process quality. If we use a different set of process conditions, the mapping between process parameters and quality will also change. This way, for each set of process conditions we need to model this unique relation. Hence, Figures 5.4 and 5.5 depict a 3D scatter plot of the impact of BHF and F in the both maximum stress and distance. In both images, two different datasets from different process conditions are presented, where conditions of $T = 1.5$, $IS = 100$ and $S = 130$ are presented in the above row and conditions of $T = 1.5$, $IS = 175$ and $S = 355$ are presented. As can be seen, the maximum stress observed in the processes is nonlinear according to the process parameters BHF and F (Figure 5.4). Regarding the distance between initial and final position of the metal sheet in the die, this process quality is linear according to the process parameters (Figure 5.5).

As detailed in the beginning of the present section, the main idea of using a high number of different deep drawing processes is to assess the performance of the proposed algorithm. The main hypothesis to be validated is related with the decrease of Mean Squared Error of the generated process model, when compared with the ground truth dataset, once the number of process models increase. This means that, for a specific process model to be generated according to the specified process conditions, its precision of prediction will increase and will get closer and closer to the real process model as soon as more source models are used. In the context of Zero-Shot Learning, the intuition is that by increasing the number of source tasks, the interpolation among these tasks will become increasingly better and the overall error for the target predictor generation will decrease. This will allow to study the applicability of HPM from problems with small number of models to more complex ones with a significantly higher number of source models available.

This way, and according to the proposed algorithm, the first step for the HPM is to train a set of process models using a machine learning technique from the available datasets and therefore build a deformable model based on SSM. This deformable model requires a set of shapes that will be generated from the trained source models. The next sub section will depict the machine learning algorithm used for modeling as well as the whole training procedure. Complementarily, a preliminary study will be made to assess the suitability of all the 18 process models to train the hyper-model in the HPM. Afterwards, two different test cases will be explored where different process models will be incrementally added in different orders during the hyper-model training. In this context, incrementally added means that first we will use 4 different process models to generate a different process model A, then increasing this number to 5 and generating the same process model A, and so forth until there are no more process models left to add. Additionally, different incremental adding orders mean that the process models to add will be chosen using a proximity measure based on the process conditions. This will allow to assess the importance of process model proximity in terms of MSE minimization for the generated process models. Intuitively, if close models to the one being generated are added to the hyper-model, the MSE

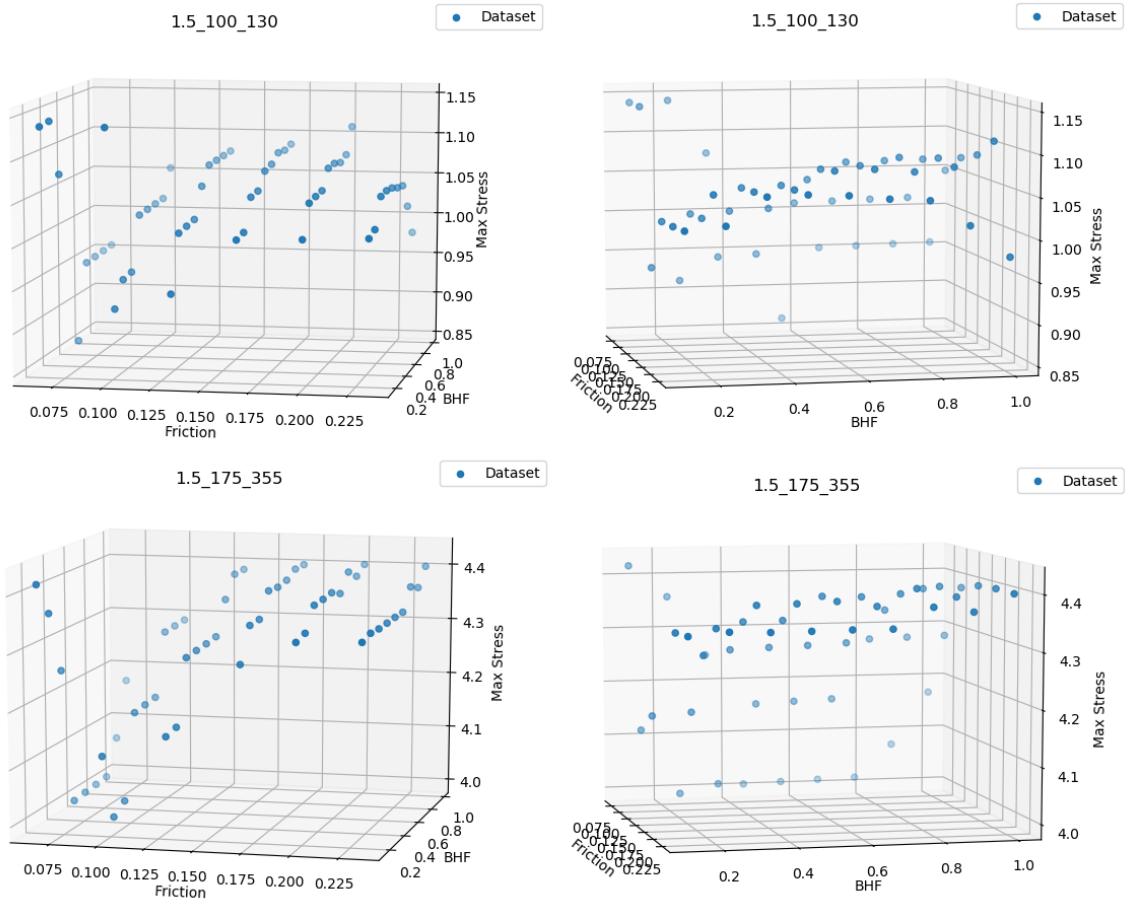


Figure 5.4: 3D Scatter plot for the process parameters BHF and F, and the process quality maximum stress for the following process conditions: Metal sheet thickness 1.5, initial stress of 100 and saturation of 130 (above) and metal sheet thickness 1.5, initial stress of 175 and saturation of 355 (below).

should decrease because a more complete picture of the deep drawing dynamics can be grasped.

5.2.2 Process Modeling

The present section aims at presenting the process of training a machine learning algorithm to map the process parameters into process quality for the deep drawing case. As explained in the previous section, 18 process models will be first assessed to know if all of them are suitable to generate the corresponding shapes for the SSM. The process parameters of blank holder force (BHF) and friction (F) are the independent variables (inputs) for the model, and process quality of maximum stress (MS) and distance (D) are the dependent variables (outputs). This way, is possible to predict the quality of the process based solely on the possible parameters to use.

As for the chosen algorithm for the modeling process, feedforward multi-layer perceptron was used, also widely known as Artificial Neural Networks (ANN), with backpropagation for

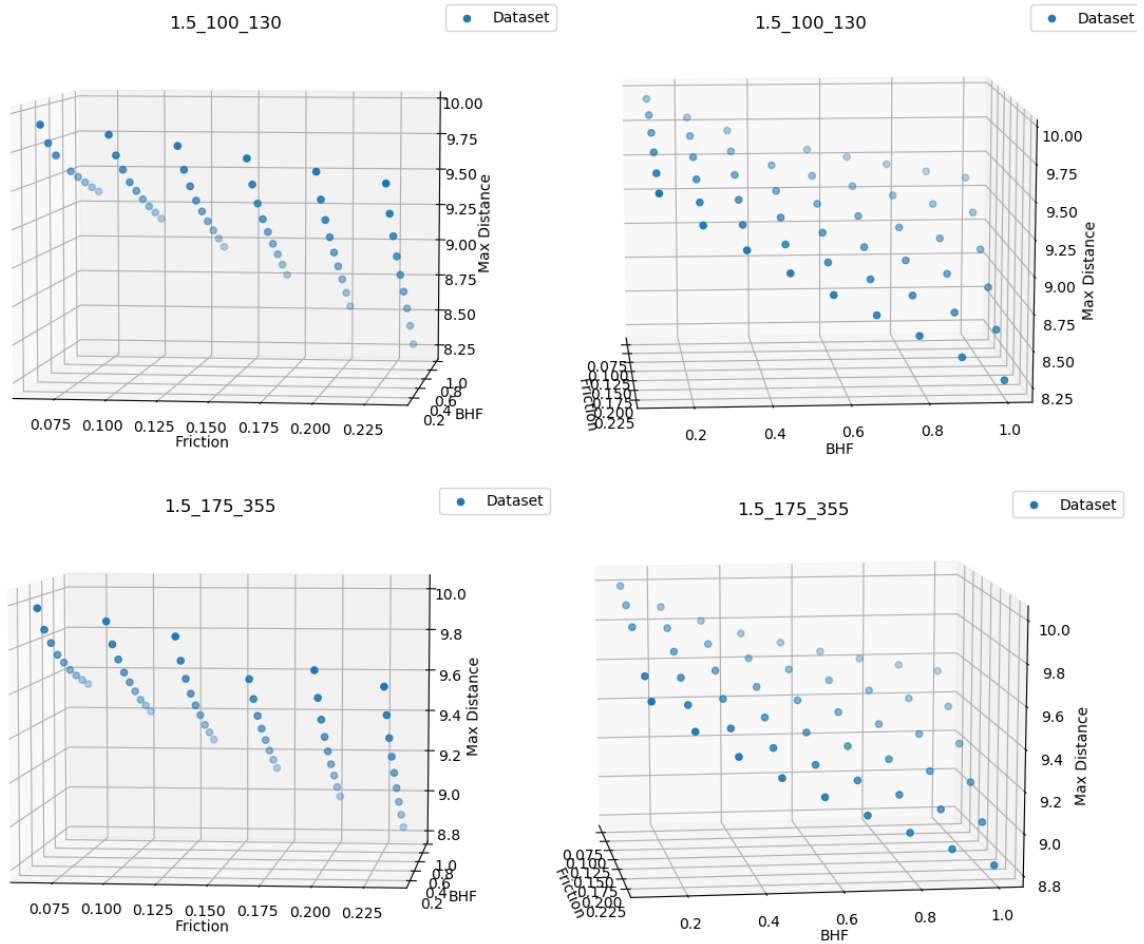


Figure 5.5: 3D Scatter plot for the process parameters BHF and F, and the process quality distance for the following process conditions: Metal sheet thickness 1.5, initial stress of 100 and saturation of 130 (above) and metal sheet thickness 1.5, initial stress of 175 and saturation of 355 (below).

parameter optimization. For this particular case, a fixed topology was used for all the 18 processes with 3 hidden layers with 10 neurons each, along with the a learning rate starting at 1 with the Adaptive Subgradient Methods for weight optimization (Duchi et al., 2011) and decreased once two consecutive epochs fail to decrease the training loss by $1e-8$, and 150000 epochs were used for the training process. This optimization method was mainly used to avoid overfitting to data and ensure convergence for the parameter optimization. In this particular case, no k-fold validation was performed due to the reduced number of datapoints and the complexity of the data to model. Instead, a visual inspection was performed in order to ensure that the model corresponds to a good generalization of the system dynamics. Table 5.2 presents all the MSE obtained after training all the process models. As can be seen, the error obtained in each of the models is very low meaning that the model could in fact fit the presented data. However, as previously said, a visual inspection is required to ensure no overfitting occurs in this situation and a good generalization is achieved.

Table 5.2: Mean Squared Error for all the 18 Deep Drawing processes using ANN.

	Mean Squared Error
Process 1	6.96e-4
Process 2	5.87e-5
Process 3	1.21e-4
Process 4	3.47e-5
Process 5	2.83e-5
Process 6	1.5e-4
Process 7	3.05e-5
Process 8	3.7e-4
Process 9	1.8e-4
Process 10	1.07e-4
Process 11	4.5e-5
Process 12	1.23e-4
Process 13	1.25e-4
Process 14	3.58e-5
Process 15	8.46e-5
Process 16	1.04e-4
Process 17	1.69e-4
Process 18	3.66e-4

Hence, Figure 5.6 depicts the datapoints of the corresponding dataset for the process conditions of $T = 1.5$, $IS = 100$ and $S = 130$ that was used for training (blue dots), and the resulting model with a set of new predicted values for a different test set (orange dots). As can be seen, the predicted model is a good generalization for both dependent variables of maximum stress and distance, meaning that they are suitable to generate the required shapes for the Hyper-Model training process. The main idea is to use the 18 trained process models to generate one shape each to be applied in the SSM.

5.2.3 Process Model Generation

In order to generate the process models based on new process conditions and existing process models, the proposed approach of HPM should be applied. As detailed in the previous sections, the proposed algorithm is composed by a technique from computer graphics called Statistical Shape Model, where based on a set of shapes, a mean shape and a set of deformable parameters are calculated in order to predict new related shapes. It is based on the differences for the set of shapes that these deformable parameters are calculated and can generate back an approximation of the real shape. Additionally, the great advantage of this method is that we can vary the deformable parameters within a certain range and create new shapes that are different, but related to those used in the SSM.

In order to create the shapes for the SSM, we need to use the process models to sample a new dataset with the same number of datapoints and all samples matching with each other. This way,

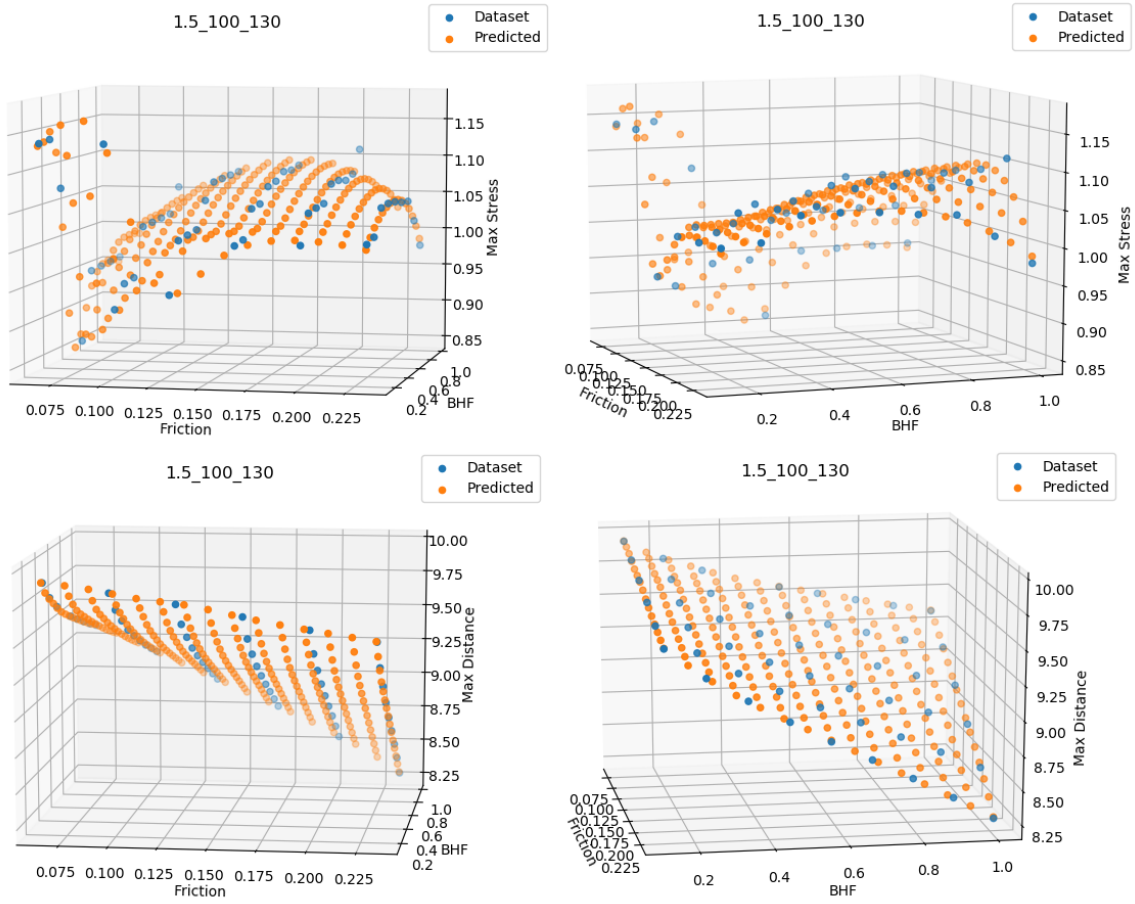


Figure 5.6: 3D Scatter plot for the Process Parameters Blank Holder Force and Friction, and the Process Quality Maximum Stress (above) and Distance (below) for the following Process Conditions: Metal Sheet Thickness 1.5, Initial Stress of 100 and Saturation of 130.

an input dataset (BHF and F) was created to be used in all the process models in order to obtain the corresponding output (MS and D). As previously explained, there must be differences in shapes in order to calculate the corresponding deformable parameters and generate back an approximation of these shapes. Hence, for this very specific problem, it is defined that the inputs provided to the models to generate the shapes are all the same. In a practical way, this means that all 18 models correspond to 18 different product variants that are yielded in the same machine. For this case, different product variants are described using the process conditions defined earlier as thickness, initial stress and saturation. As far as the machine is the same, so are the machine parameters and its range used to explore its dynamics. This way, for different product variants only the output will be different, and hence, a shape is defined only with the output of the process models.

As for the defined inputs of the process models, a dataset was built using a Full Factorial Design approach. This method allows to grasp the overall distribution of response variables according to a full variation of process parameters. The design is dependent on levels and factors. Factors

are simply the number of process parameters that one wants to vary, and the levels are the number of equally distributed points for each process parameter to perform an experiment. Assuming a full combination of all the process parameter value, the number of datapoints is determined using l^n , where l is the number of levels and n is the number of factors. For this case, since the inputs are friction and blank holder force, the number of factors is 2 and we have defined a number of levels as 15 to have a large and complete representative shape. This means that the number of datapoints per shape is $15^2 = 225$, which is more than 4 times larger than the initial datasets. The orange dots on the scatter plot from Figure 5.6 also represent the shape composed from 225 points.

Before moving forward, for this particular case we should clarify an important point about the generation of shapes. One might question why we need to train the process models at this stage to generate a set of shapes and not simply use the 54 datapoint datasets sampled from the Abacus simulator directly in the SSM as shapes. Despite being a reasonable thing to do in this very specific context, this is not usually the case at shop-floor. Normally, different people perform a different amount of experiments with different input values for machine parameters, which invalidates using these experiments as shapes due to 1) different amount of datapoints and 2) non-matching datapoints throughout the datasets. As previously defined, shapes need to have the same amount of landmarks (datapoints) and should exist a correspondence between these datapoints, hence the name landmarks. From this point of view, the trained process models can be seen as a transformation from the initial datasets into suitable shapes. This additional step of training the process models was taken because we wanted to replicate the normal conditions if such a system would be used in real life, and therefore test the proposed algorithm as fair as possible.

This means for the SSM we will have a set of shapes based on 225 datapoints sampled from the process models, where the statistical model and the corresponding deformable parameters b need to be calculated. Since we have two output variables per process model, we can consider these as 2D shapes $x = [S_1, \dots, S_M, D_1, \dots, D_M]^T$ being M the number of datapoints sampled, which for this case is 225. By using Equations 2.8 and 2.13 the mean shape $\bar{x}$ can be calculated, along with the eigenvectors ϕ and eigenvalues λ . As for the SSM, the number of features for each shape is the number of datapoints multiplied by the number of dimensions used for the shapes, so we get $225 \times 2 = 450$ features, which means 450 eigenvectors and eigenvalues. To avoid handling such number of eigenvectors, only the first eigenvectors (ordered by decreasing order of corresponding eigenvalues) that are significant to the SSM are chosen. Hence, we get the eigenvector ϕ and mean shape $\bar{x}$. Afterwards, the deformable parameters b can be calculated using Equation 2.15. As detailed in Section 4, these b deformable parameters together with the process conditions ζ are the essential parts to build up the Hyper-Model used in the HPM.

In order to determine the significant number of components / eigenvectors to use, a preliminary assessment was made. A rule of thumb used when performing PCA or other decomposition method is to use a number of components that describe at least 95% of all data variance, assuming these components are sorted in decreasing order of variance. Despite this kind of approach is applied in cases of feature selection, in this case our goal is to perform dimensionality reduction and not so much the case of selecting the main features to understand which attributes are more rele-

Table 5.3: Assessment of the number of components used in PCA for SSM to generate back the initial shape.

Process	Process Conditions	1 Comp	2 Comps	3 Comps	4 Comps	5 Comps	6 Comps
1	[1.5 100. 130.]	0.067319	0.001447	0.001265	0.001209	0.001098	0.000534
2	[1.5 100. 165.]	0.097688	0.001448	0.000333	0.000328	0.000296	0.000153
3	[1.5 100. 200.]	0.103319	0.004047	0.001230	0.001197	0.000938	0.000799
4	[1.5 175. 227.5]	0.095521	0.004595	0.000795	0.000357	0.000300	0.000296
5	[1.5 175. 288.75]	0.091586	0.010356	0.001032	0.000907	0.000722	0.000572
6	[1.5 175. 355.]	0.072669	0.017411	0.002619	0.002231	0.001028	0.000306
7	[1.5 250. 325.]	0.068921	0.012403	0.002196	0.001007	0.000444	0.000393
8	[1.5 250. 412.5]	0.050095	0.019486	0.000765	0.000748	0.000571	0.000477
9	[1.5 250. 500.]	0.038059	0.029813	0.004104	0.000955	0.000334	0.000242
10	[2. 100. 130.]	0.840504	0.149459	0.000010	0.000003	0.000002	0.000001
11	[2. 100. 165.]	0.065349	0.048184	0.001317	0.000184	0.000159	0.000143
12	[2. 100. 200.]	0.047413	0.035017	0.000990	0.000335	0.000289	0.000234
13	[2. 175. 227.5]	0.056604	0.036109	0.000397	0.000350	0.000335	0.000302
14	[2. 175. 288.75]	0.039476	0.022818	0.001807	0.001692	0.001107	0.000832
15	[2. 175. 355.]	0.040388	0.014643	0.001805	0.001545	0.001315	0.001203
16	[2. 250. 325.]	0.058006	0.023535	0.002163	0.001330	0.001279	0.001178
17	[2. 250. 412.5]	0.062091	0.011179	0.001258	0.000723	0.000648	0.000575
18	[2. 250. 500.]	0.095793	0.006742	0.002077	0.001939	0.000816	0.000409
	Average	0.110600	0.024927	0.001454	0.000947	0.000649	0.000480

vant for the problem, as usually performed in feature selection. The main idea is to use only a few components that allow to minimize the error when generating back the shapes in SSM, regardless of what they are.

Hence, the performed assessment compares an increasing number of components used in the SSM with the MSE of real and generated shapes from a specific process. Additionally, the average of all the processes per number of components tested was also calculated. Therefore, Table 5.3 displays the MSE for all tests performed ranging from 1 component generating back the shapes, to the 6 components that describe the highest variance from data. As expected, as the number of components increase, the MSE of the generated shapes decreases. This is valid for each shape and also for the average of all shapes per number of components. It can be seen that a drastic reduction of the MSE occurs until the use of 3 components, and from that point the MSE decrease is less significant. This trend can be better seen in Figure 5.7 where from 3 components the increase of component does not decrease significantly the MSE.

Now, the next step is to calculate all the deformable parameters regarding each of the shapes used in the SSM. For that, Equation 2.15 should be used together with the eigenvectors, as well as the real shape and the mean model. This way, we have a set of parameter values where we can generate back the initial shape by just using the deformable model using Equation 2.14.

Since the main idea is to build a hyper-model for HPM based on the process conditions and deformable parameters, in order to better understand the relation among the shapes in the deformable space, a 3D scatter plot is built. Figure 5.8 depicts all the 3 deformable parameters. Is

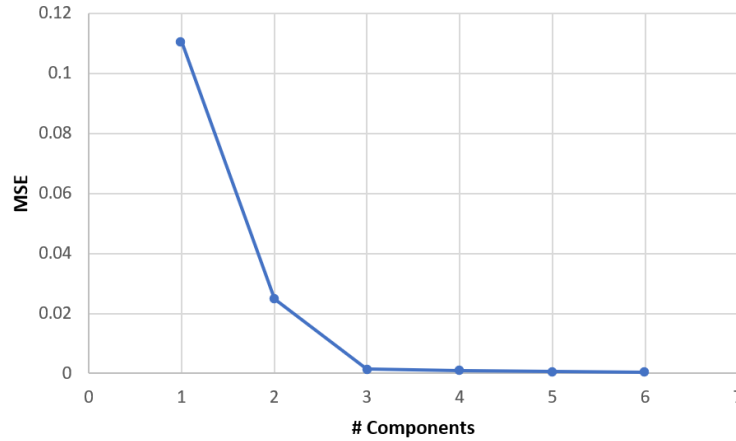


Figure 5.7: Graphical display of number of components and the average MSE for all the processes per number of components.

can be clearly seen that there are two groups of points referring to the two different existing thicknesses, being clear that this process condition is greatly correlated with deformable parameter 2. Moreover, it can also be seen that the remaining process conditions of IS and S are more correlated with deformable parameters 1 and 3. Ultimately, it can be seen an outlier in the top right corner, which is also one of the main reasons to visually inspect the dataset. This point is visibly far from the two observed clusters because, in order to generate back the respective shape, the parameters should be considerably different. This means that the shape is too different from the mean calculated shape for the deformable model, and hence too different from the remaining ones. This shape refers to the process condition where $T = 2$, $IS = 100$ and $S = 130$. Thus, due to this fact, this model was not considered for the next steps of the HPM approach. So, instead of using all 18 models only 17 were used. Hence, the final set of 17 deep drawing processes used for the further tests is presented in Table 5.4.

Based on the definition of the hyper-model, a mapping of process model parameters into process conditions should be made using a machine learning algorithm as depicted in Equation 2.4. This means that in order to predict the process model parameters for the SSM based on new process conditions, a minimization problem needs to be formulated. The objective function for this problem would be simply the difference between the model prediction and the target process conditions, where a L1, L2-norm or Quadratic Loss function could be used. However, since in this case we have 3 process conditions and 3 components from the SSM, which are considered the process models parameters, we could simply train a machine learning model inversely. Hence, instead of mapping process models parameters into conditions, we can map process conditions directly in the process models parameters and avoid the minimization problem formulation. This means that we can build a predictor based on all the training set from all the process conditions and parameters collected previously to estimate the process parameters. Considering the process conditions as ζ_n and the process models parameters, which are the deformable parameters from

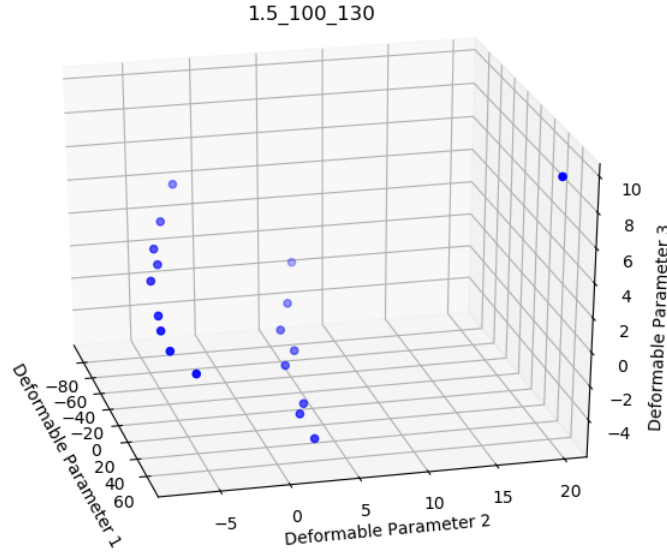


Figure 5.8: 3D scatter plot for 3 deformable parameter regarding all 18 shapes from the deep drawing processes.

the SSM, as λ_n , the new equation used for the hyper-model is as follows:

$$\lambda = g_{\beta}(\zeta) \quad (5.2)$$

As for the next step of the proposed approach, the hyper-model should be trained. This means that a function $H : R^3 \rightarrow R^3$ needs to be modeled in order to map process conditions into deformable parameters. For this case, three different statistical methods were used, namely a Linear Regression Model (Lai et al., 1979), a Least Absolute Shrinkage and Selection Operator (also known as LASSO regression) (Tibshirani, 1996) and Ridge Regression (Hoerl and Kennard, 1970) - which is based on Tikhonov regularization and in machine learning mostly known as weight decay for ANNs. For the Linear Regression, a polynomial of degree 1, 2 and 3 were used, and for both LASSO and Ridge only polynomials of degree 2 and 3 were used. The reason behind the chose of these models lie in the existence of a small amount of datapoints to train and the expected shape to model in the deformable parameter space.

The Linear Regression method is a widely used approach to model a relation between certain inputs (independent variables) and outputs (dependent variables) and is considered the first regression analysis method to be thoroughly studied, being used in vast number of practical real world applications. Normally, when there's more than one input variable, this method is often called Multiple Linear Regression. The linear regression takes the following form:

$$\hat{y} = X\hat{\beta} \quad (5.3)$$

where the input vector is $X^T = (1, X_1, X_2, \dots, X_p)$. The value 1 is added to input vector due to

Table 5.4: Process conditions for the deep-drawing process simulation in Abaqus.

	Metal Sheet Thickness (T)	Initial Stress (IS)	Saturation (S)
Process 1	1.5	100	130
Process 2	1.5	100	165
Process 3	1.5	100	200
Process 4	1.5	175	227.5
Process 5	1.5	175	288.75
Process 6	1.5	175	355
Process 7	1.5	250	325
Process 8	1.5	250	412.5
Process 9	1.5	250	500
Process 10	2	100	165
Process 11	2	100	200
Process 12	2	175	227.5
Process 13	2	175	288.75
Process 14	2	175	355
Process 15	2	250	325
Process 16	2	250	412.5
Process 17	2	250	500

the β_0 parameter. The process of parameter estimation takes into consideration the training data with N cases, where each case i is $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$. For this case, the popular ordinary least squares will be used where the β coefficients are found based on the minimization problem using the residual sum of squares

$$\hat{\beta} = \arg \min_{\beta} \|y - X\beta\|_2^2 \quad (5.4)$$

where a unique solution can be found using the following expression in matrix notation:

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (5.5)$$

For the polynomial degrees form, the input vector is represented as $X^T = (1, X_1^1, \dots, X_p^n)$ where p is the number of independent variables and n is the polynomial degree. This means that a 2 degree polynomial input vector takes the form of $X^T = (1, X, X^2)$ and 3 degree polynomial of $X^T = (1, X, X^2, X^3)$.

Since the dataset to train the hyper-model is composed of few cases, the use of 2 and 3 degree polynomial can lead to overfitting situations. Moreover, since the tests to be made consider an increasing number of available models from a minimum of 4 and a maximum of 17, the use of quadratic and cubic functions in the polynomial will almost certainly lead to overfitting situations. Therefore, regularization terms can be used in the polynomial form to improve the hyper-model performance when estimating new deformable parameters. Hence the use of LASSO and Ridge

regression for this case.

The LASSO and Ridge regression methods are framed into a category of methods often called Shrinkage methods mainly due to the regularization terms that are added to the linear regression model. These regularization terms penalize high values for the model parameters forcing and "shrinking" these to be near zero. The minimization of penalized residual sum of squares for LASSO is the following:

$$\hat{\beta}^{lasso} = \arg \min_{\beta} \frac{1}{2N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \quad (5.6)$$

where $\lambda \geq 0$ is a complexity parameter that controls the amount of parameter shrinkage by using a L1-norm penalty. The ridge regression minimization of penalized residual sum of squares is similar to LASSO, but instead of using a L1-norm, a L2-norm is used.

$$\hat{\beta}^{ridge} = \arg \min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \quad (5.7)$$

One of the main differences between L1 and L2-norm is that with small values of λ the L1-norm forces the parameters to be exactly 0, contrary to the L2-norm case. This means that LASSO makes some sort of subset selection of the model parameters. Due to this fact, LASSO is also used in the literature for the problem of dimensionality reduction.

For a better understanding of the tests to be performed, we should refer to the linear regression as LIN, degree 2 polynomial as POL2, degree 3 polynomial as POL3, LASSO using a degree 2 polynomial as LASSO2, LASSO using a degree 3 polynomial as LASSO3, and the same for Ridge, where RIDGE2 and RIDGE3 are the degree 2 and 3 polynomial, correspondingly. This leads to the test of 7 different approaches for the hyper-model in the HPM.

Before diving into the results of all the proposed tests, we should first wrap-up all the conditions for the tests performed. As for the number of components / deformable parameters, 3 of them will be used in decreasing order of variance because more components do not reduce the error once generating back the shapes in SSM. For the number of process models available to generate the shapes for SSM, 17 out of 18 will be used due to an outlier of one of the deformable parameters to train the hyper-model. Moreover, 7 different methods will be used, ranging from linear regression to Ridge regression with degree 3 polynomial. In the next section we will depict and detail all the results and all the tests performed.

5.2.4 Results

Complementary to all the conditions aforementioned about all the regression methods to be used, 2 different tests will be performed in order to better understand the properties of HPM in the context of deep drawing regarding different process conditions. In order to test the hyper-model performance in the HPM algorithm using an increased number of processes, one should define a

distance measure between the processes. The main idea is to chose the closest processes to the one being generated in order to assess the improvement of the proposed approach once new processes are contemplated in the hyper-model.

First, this distance between processes is calculated using the values of process conditions associated with each process model. This way, if we assume the process conditions take the form of a 3-d vector space due to thickness (T), initial stress (IS) and saturation (S), we can define, e.g. T=1.5, IS=100 and S=130 as $\vec{\theta} = (1.5, 100, 130)$. Based on this vector representation, two different distance functions are used: 1) Euclidean distance; 2) Euclidean distance using normalized process conditions. Equation 5.8 is used to calculate the Euclidean distance, where the main difference between 1) and 2) is that the latter uses the normalized data to calculate the distance between 0 and 1 eliminating conditions' degree of magnitude. These leads to two different scenarios where the process models chosen to incrementally add to the hyper-model change and the impact of different orders or sorting strategies can be assessed.

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2} \quad (5.8)$$

The equation to normalize all process condition values between 0 and 1 is as follows:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (5.9)$$

where x_i is the i^{th} case per process conditions feature (e.g. thickness), and $\min(x)$ and $\max(x)$ are the minimum and maximum values for the process conditions feature, correspondingly.

Table 5.5 presents the ranking of deep drawing processes according to the two approaches for distance calculation, considering the target process of T=1.5, IS=100 and S=130. As can be seen, the Euclidean distance sorts the processes first by initial stress, then by saturation and only then by thickness. As for the Euclidean distance of normalized process conditions, first the processes are sorted by thickness, then by initial stress and finally by saturation. Considering the present dataset, it was chosen not to use the same approach of first sort by saturation because there are more groups due to a higher number of possible values, resulting in a lower number of cases for the same value of saturation (a maximum of 2 cases). This is not the case for process conditions thickness and initial Stress, where for the same value of thickness we have up to 9 cases and for initial stress there are up to 6 cases. Thus, sorting by saturation is not relevant in this context. Hence, this kind of comparison will allow to assess which process condition is more meaningful / can decreases the MSE of shape generation, for the target process.

In the next sub-section an assessment of LIN, POL2 and POL3 for the two types of distance measures will be made, where further on the same assessment will be made but with LASSO2, LASSO3, RIDGE2 and RIDGE3.

Table 5.5: Ranking of deep drawing processes by shortest distance from process with $T=1.5$, $IS=100$ and $S=130$, using 1) Euclidean distance and 2) Euclidean distance with normalized process conditions.

Order	Euclidean Distance			Euclidean Distance using Normalized Process Conditions		
	Thickness	Initial Stress	Saturation	Thickness	Initial Stress	Saturation
1	1.5	100	165	1.5	100	165
2	2	100	165	1.5	100	200
3	1.5	100	200	1.5	175	227.5
4	2	100	200	1.5	175	288.75
5	1.5	175	227.5	1.5	175	355
6	2	175	227.5	1.5	250	325
7	1.5	175	288.75	1.5	250	412.5
8	2	175	288.75	1.5	250	500
9	1.5	175	355	2	100	165
10	2	175	355	2	100	200
11	1.5	250	325	2	175	227.5
12	2	250	325	2	175	288.75
13	1.5	250	412.5	2	175	355
14	2	250	412.5	2	250	325
15	1.5	250	500	2	250	412.5
16	2	250	500	2	250	500

5.2.4.1 Linear, Degree 2 Polynomial and Degree 3 Polynomial Regression

One of the main aspects to be assessed in the proposed HPM approach for shape generation, and consequently target process model training, is the performance of the hyper-model in terms of predicting the deformable parameters for the SSM. This will allow to compare how close the generated shapes are from its ground truth. First, as previously discussed, each of the 17 process model shapes will be generated according to an increasing number of different process models. This increasing number will start with the 4 most closest source models according to the distance measure to be used, and the MSE will be calculated for that case. Then, the 5 most closest source models will be used, and again, the MSE calculated. This process goes until all the 16 remaining process model shapes are used in the hyper-model for shape generation. For one modeling technique, e.g. LIN, this means that for each process model generation, 13 tests (from 4 shapes to 16) will be made, resulting in a total of $13 \times 17 \text{ processes} = 221$ tests in order to assess all the process models. Finally, 7 different modeling techniques were used, so $221 \times 7 = 1547$ tests were performed in total for the whole thesis.

The first analysis made was to assess how the error (MSE) between the generated shape and the ground truth evolves as soon as more process models are added to the hyper-model in HPM. For that case, the generation of all 17 process shapes was tested from a minimum value of the 4 closest processes to a maximum of 16 processes, as previously described. Therefore, the mean value of all MSE using a fixed number of source models was calculated. Figure 5.9 presents all

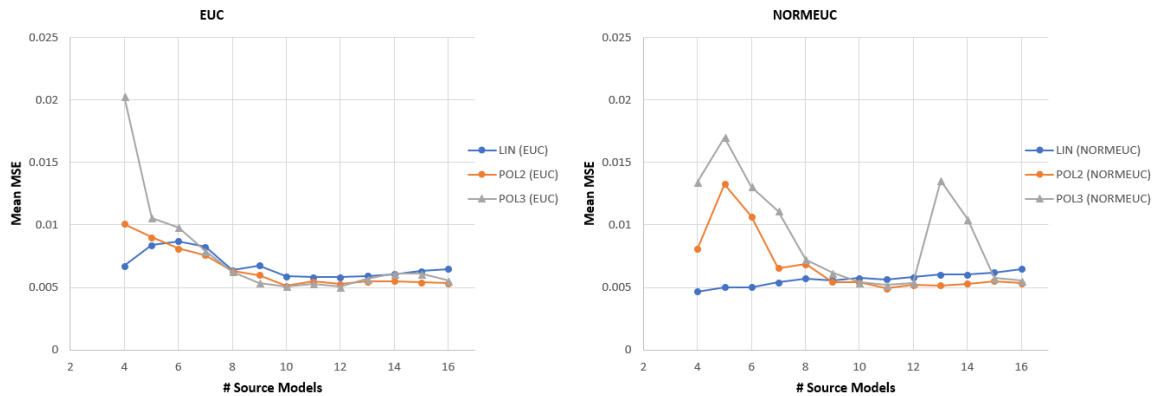


Figure 5.9: Mean values of MSE for all the processes per number of source models using LIN, POL2 and POL3 techniques. On the left, the Mean MSE for Euclidean distance (EUC) and on the right for the Euclidean distance with normalized process conditions (NORMEUC).

these mean values for each number of closest source models, from 4 to 16, for the LIN, POL2 and POL3 techniques using both distance measures EUC and NORMEUC.

The first major difference between NORMEUC and EUC approaches for the LIN technique is related with the error evolution behavior in regard to the number of source models, where the mean MSE for NORMEUC using 4 to 10 source models is lower when compared with the EUC approach. Once the number of source models increase, the performance of the both approaches tend to be the same, as expected. However, despite the low MSE for low amount of source models in the NORMEUC, the MSE increases as soon as new source models are added. Additionally, both POL2 and POL3 suffer from overfitting for a lower amount of source models used leading to high peaks of MSE in both EUC and NORMEUC. For the POL3 approach using NORMEUC, also overfitting effects can be seen for a higher amount of source models. The overfitting effects will be thoroughly explored later in this sub-section. Despite not being significantly lower, the POL2 and POL3 techniques can perform better than LIN in both EUC and NORMEUC for a high number of source models, where it seems to neither increase or decrease. This is contrary to the LIN approach where the MSE is slowly increasing. This slow increase might be due to the fact that a linear model is underfitting the data, revealing that the model complexity is not enough to mimic the system dynamics. Finally, for this first analysis, it can be seen that MSE varies according to the number of source models used, and also that different performances are observed when using different types of statistical techniques for modeling.

In order to better understand the presented results, more detail about how the MSE for each process evolves over the number of source models is required. Therefore, Figure 5.10 presents a set of 6 plots, where each plot contains the MSE for all 17 process models from 4 to 16 source models. The left column presents the results of LIN, POL2 and POL3 techniques, from top to bottom for EUC, and the right column uses the same organization but for NORMEUC. From these plots, there are two different aspects that catch the eye: 1) Peaks of high MSE in certain process

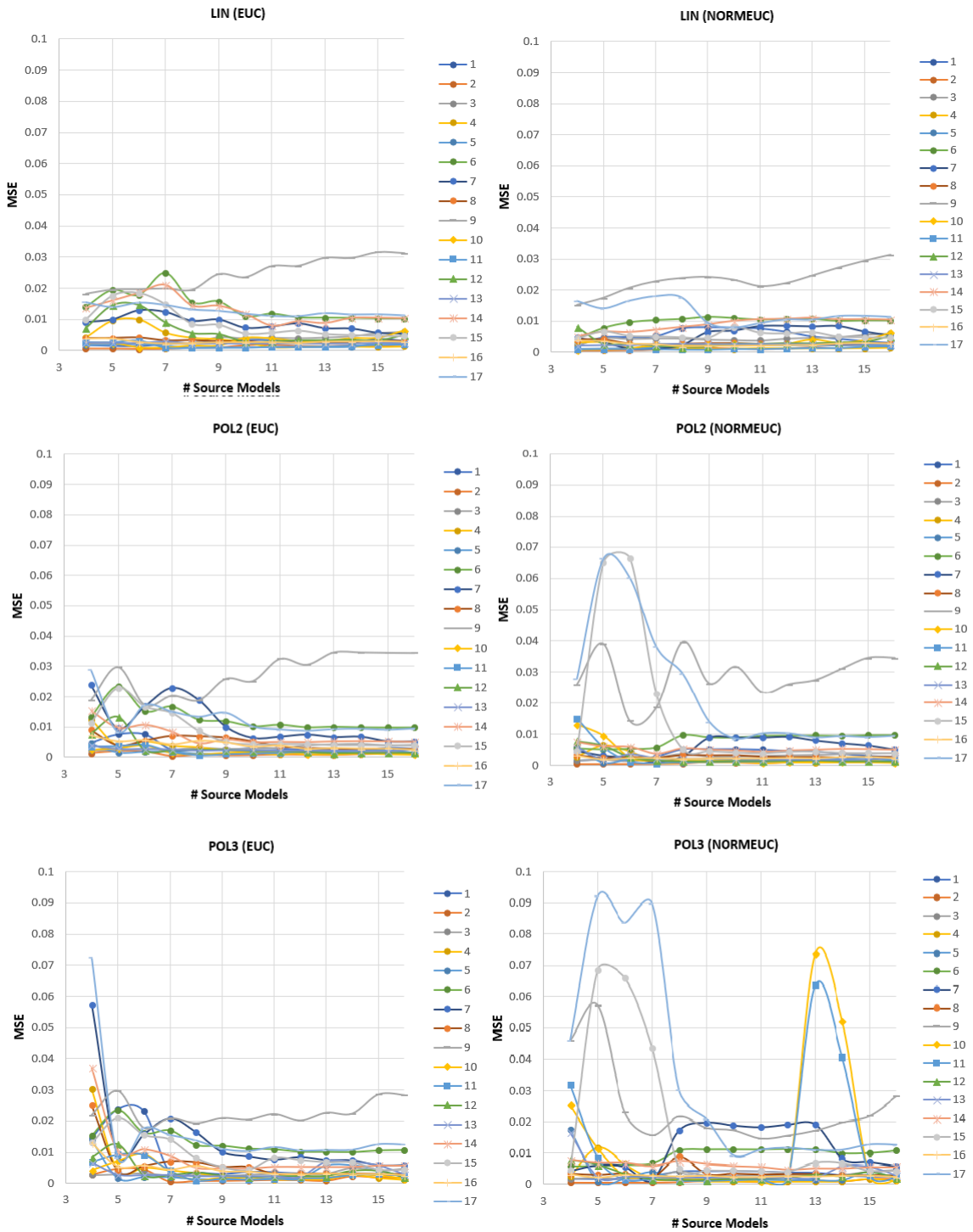


Figure 5.10: MSE values for all the 17 processes with an increasing number of source models. On the left, 3 graphs are displayed with LIN, POL2 and POL3 techniques from top to bottom for Euclidean distance (EUC). On the right, the same techniques are used for Euclidean distance with normalized process conditions (NORMEUC).

models, mainly in POL2 and POL3 for NORMEUC in some processes; 2) Some process models do not decrease the MSE over the increase of source models. As discussed previously, the main hypothesis for the peaks presented in Figure 5.10 is the overfitting effect due to low amount of data points and high model complexity. This is not observable in LIN technique because of the low model complexity. In order to prove that overfitting is occurring in these models, a visual inspection should be performed.

For this case, process 15 with process conditions $T = 2$, $IS = 250$ and $S = 325$ was chosen for this analysis using NORMEUC strategy. Hence, in order to assess if overfitting is occurring, the shape of predicted deformable parameters using POL2 as a technique to train a hyper-model should be plotted. Figure 5.11 presents three different plots, each using different amounts of source mode, namely 4 source models (top left plot), 5 source models (top right plot) and 9 source models (bottom plot). These values of source models were chosen because represent the state of the hyper-model before overfitting, when it overfits and after overfitting, correspondingly. As can be seen, the shape of the predicted values for 4 and 9 source models is simple and smooth, as the presented data points. The observed predicted shape for 5 source models is irregular and it predicts the target process condition far from the optimal value due to this fitting of data using a irregular shape. The blue dots are the source models, the green dots are all the predicted values in the range of process conditions (Thickness = [1.5, 2], Initial Stress = [100, 250], Saturation = [130, 500]) acting as a test set, and orange dot is the predicted value for the target process conditions (process 15) that will be used to generate the corresponding shape. For comparison reasons, Figure 5.12 presents the plot for 4, 5 and 9 source models for the same target model but now for EUC strategy. Contrary to the NORMEUC strategy, now a regular shape is predicted and no overfitting effect occurs.

Additionally, and in order to better understand the impact of overfitting in the final generated shape, Figure 5.13 presents the maximum stress and distance for all generated shapes using 4, 5 and 9 source models. The blue dots represent the group truth of maximum stress and distance, and the orange dots represent the generated shapes from the HPM. It can be seen that the generated shape for 5 source models is far more disperse with high values in its periphery. This difference can be greatly noticed if we look at the value range using 5 source models, where the maximum stress varies between 4.5 and 7, and using 4 source models with a shorter range between 4.9 and 5.8. This range is even shorter with 9 source models values between 5.1 and 5.5. As for the process quality Distance, it can be seen that for 5 source models the prediction surface is quite irregular and bumpy compared with the smooth surface from the ground truth. For 4 data points this prediction is better with a more regular surface and for 9 data points it is near perfect fit.

Despite the presented results of the overfitting effect, one might wonder why this also didn't occur when using 4 process models, or even why it also didn't occur in other processes if it occurs for process 15? For all the experience dealing with such process models and data from the Abaqus simulator, the relation between processes is not that linear as encoded by the process conditions. One might realize that the three process conditions are a rough approximation of the process itself. The true distance between the process models cannot be summed up to an Euclidean

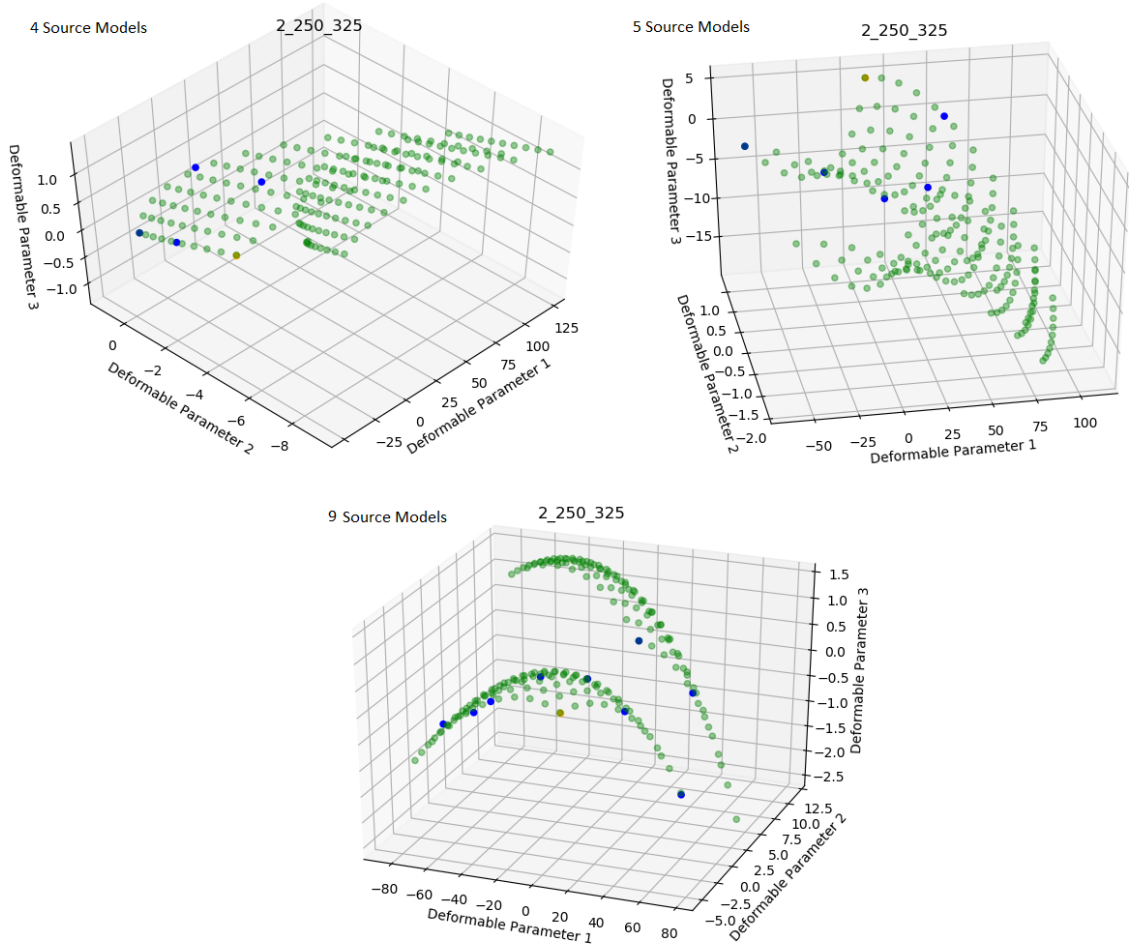


Figure 5.11: Scatter Plot for the deformable parameters using NORMEUC strategy. The blue points are the source models, the green points are all the predicted values in the range of process conditions (Thickness = [1.5, 2], Initial Stress = [100, 250], Saturation = [130, 500]) and orange point is the predicted value for the target process conditions (process 15) that will be used to generate the corresponding shape.

distance among process conditions as the dynamics are much more complex than the ones grasped by the dataset. To this intent, it is normal that in some cases the inclusion of certain process models in the hyper-model for shape generation bring more entropy to the whole solution rather than an improvement.

This is tightly related with the way these process models are incrementally added to the hyper-model, which makes a considerable difference. In Figure 5.9 it can be seen that no peaks occur during the inclusion of source models into the proposed approach when using the EUC strategy. However, it can be seen in more detail using Figure 5.10 that even for EUC there are small peaks for certain processes, although not as high as in the case of NORMEUC. This difference is mostly due to how the source models are selected for inclusion. As for the NORMEUC, all the processes with the same Thickness are presented first to train the hyper-model, as seen in Table 5.5. This

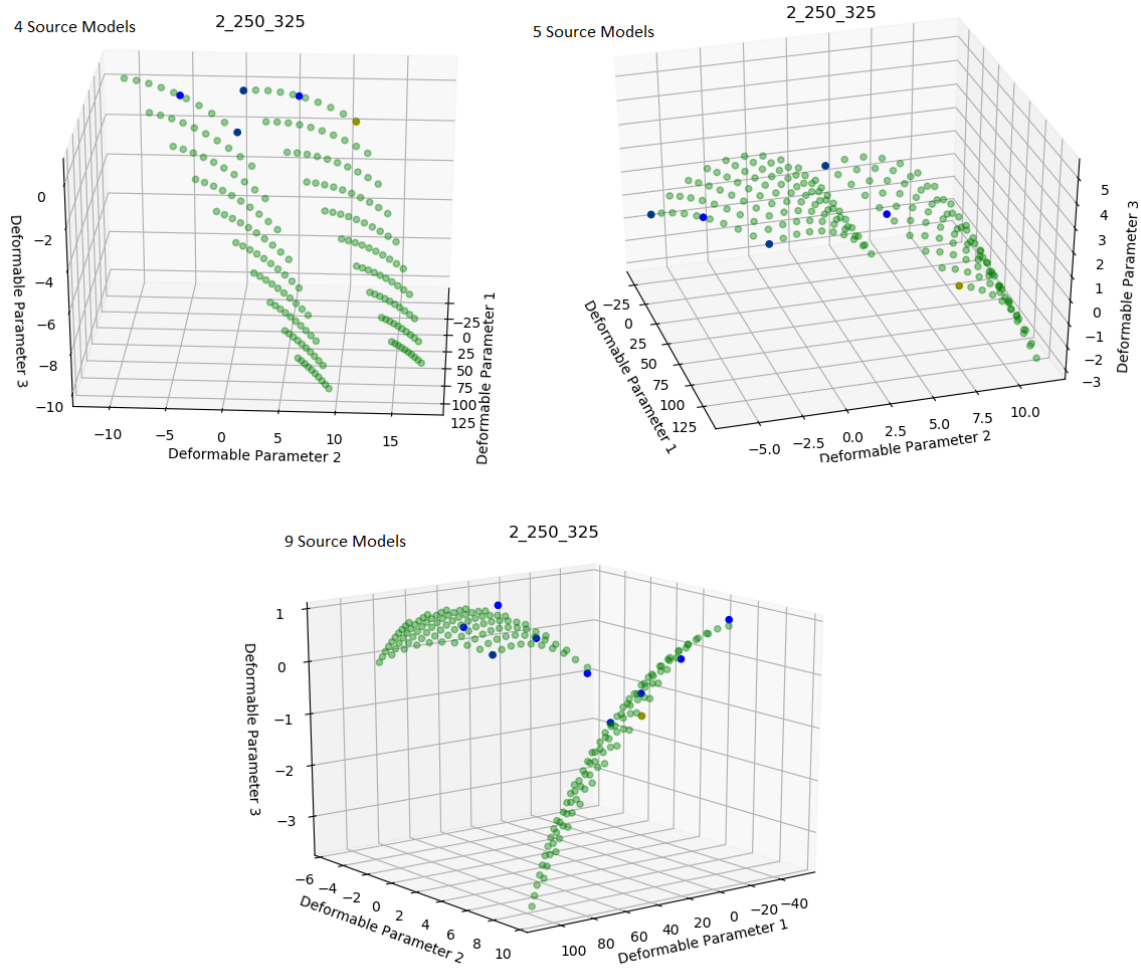


Figure 5.12: Scatter Plot for the Deformable Parameters in the SSM using EUC strategy. The blue points are the source models, the green points are all the predicted values in the range of process conditions (Thickness = [1.5, 2], Initial Stress = [100, 250], Saturation = [130, 500]) and orange point is the predicted value for the target process conditions (process 15) that will be used to generate the corresponding shape.

means that more heterogeneity can be grasped from initial stress and saturation, which is beneficial only for a specific thickness in the target process condition. In the EUC case, the same initial stress and saturation is presented first, meaning that initially a high heterogeneity of thicknesses is presented to the hyper-model. Assuming that the deep drawing deformable parameters are grouped by thickness, as observed in Figure 5.8, the EUC can be seen as a strategy that initially looks for the same examples in different groups to get a holistic picture of the whole deep drawing. The NORMEUC first looks to all possible examples in the same group to rapidly grasp a local perspective of the process for that specific thickness. This justifies why MSE for the EUC strategy starts with high values and quickly decreases as soon as new source models are included, but no

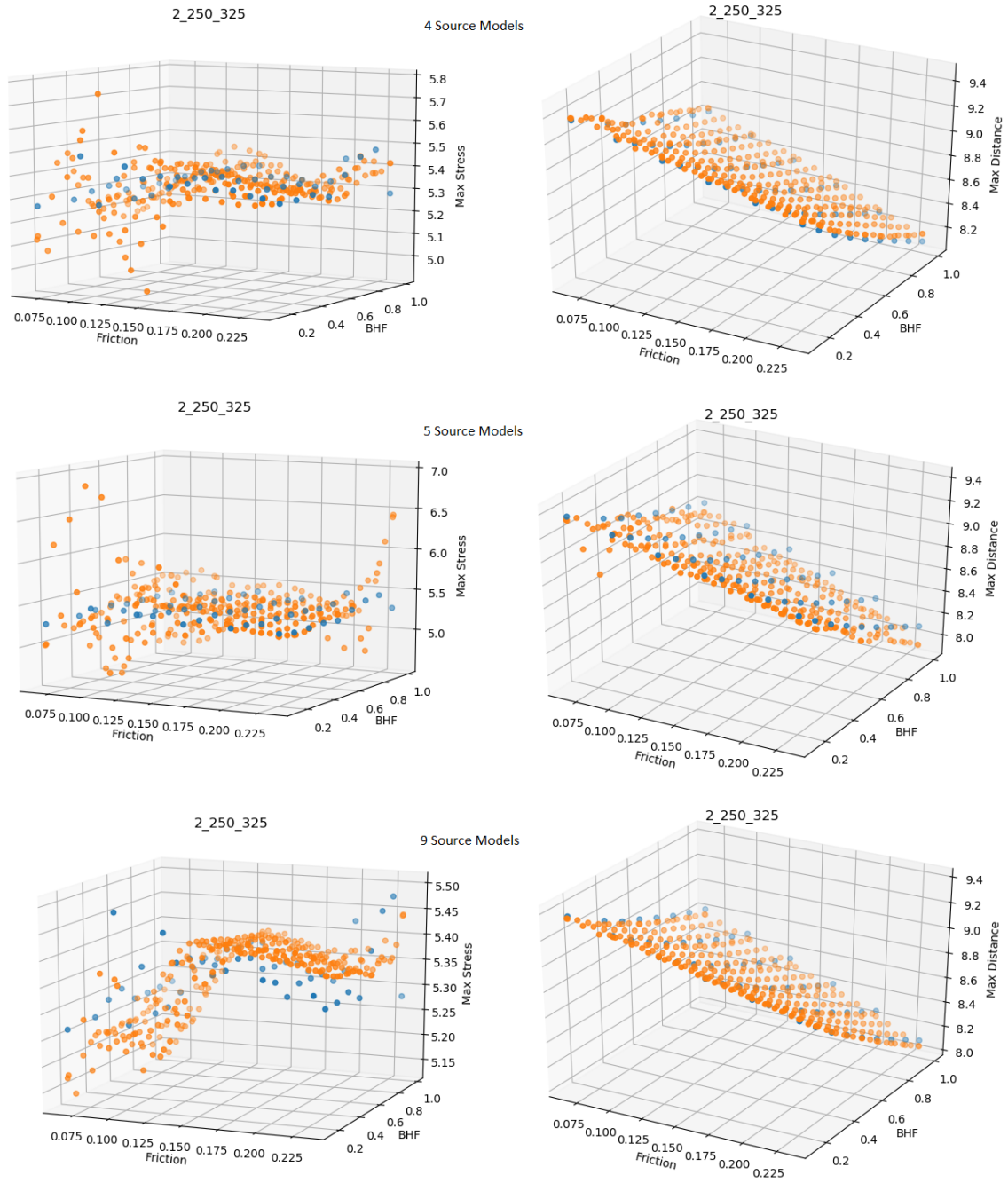


Figure 5.13: 6 Scatter Plots of maximum stress and distance using the for degree 2 Polynomial with NORMEUC strategy for shape generation of process 15. On the left there are the maximum stress plots with 4, 5 and 9 source models from top to bottom. On the right, the same organization is used but for process quality distance. The blue points represent the group truth of maximum stress and distance, and the orange points represent the generated shapes from HPM.

overfitting effect occurs due to this more complete understanding of the deep drawing process. Contrary to this, since in NORMEUC only process models with the same thickness are included

first, a more local perspective and less holistic overview is obtained. This imply that for low amount of source models the MSE can be low, but due to low amount of data, when using more complex techniques, overfitting might occur. Nevertheless, since a local perspective is first built for NORMEUC, there's a chance that overfitting can also occur for high number of source models. This is seen in processes 10 and 11 for NORMEUC, where it peaks with high MSE at 13 source models.

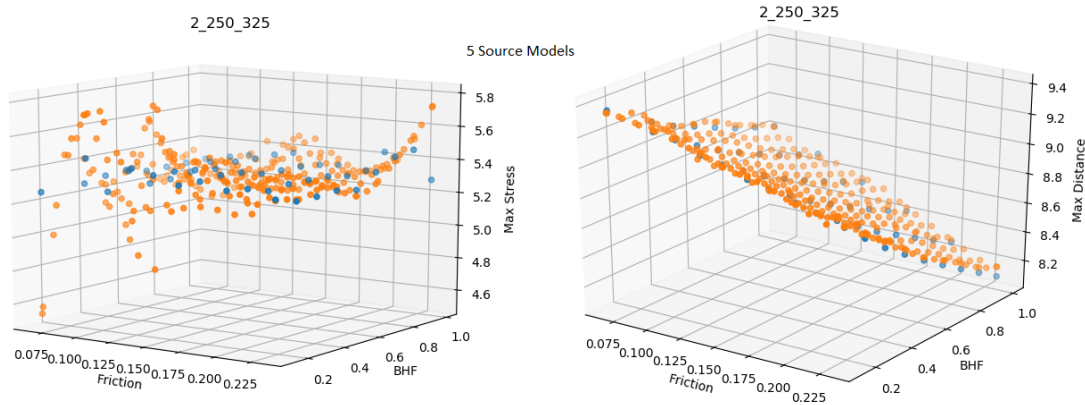


Figure 5.14: 2 Scatter Plots of Maximum Stress and Distance using the for Degree 2 Polynomial with EUC strategy for shape generation of process 15. On the left there are the Maximum Stress plots with 5 source models from top to bottom. On the right, the same organization is used but for process quality Distance. The blue points represent the group truth of Maximum Stress and Distance, and the orange points represent the generated shapes from the Hyper-Process Model.

In order to better compare the two strategies of EUC and NORMEUC, Figure 5.14 presents the same results as Figure 5.13 for process 15, but now using EUC strategy using 5 source models (when it overfits using NORMEUC). The first difference to notice is that for EUC in 5 source models, contrary to the NORMEUC, process models with both 1.5 and 2 thicknesses are used to train the hyper-model. This makes a significant difference avoiding overfitting because examples of both groups (thickness 1.5 and 2) are shown and a more complete understanding of the deep drawing dynamics is grasped. This can be proved by a more smooth and regular shape presented in EUC for 5 source models when compared with NORMEUC. Finally, we can see from Figure 5.14 that the generated shape for the target process is better in both Maximum Stress and Distance comparing with NORMEUC.

5.2.4.2 LASSO Degree 2 Polynomial, LASSO Degree 3 Polynomial, Ridge Degree 2 Polynomial, Ridge Degree 3 Polynomial

Due to the overfitting effect observed in the previous section, there are some ways to tackle such an issue. As described in the beginning of the section, regularization terms are often used in the problem of parameter optimization for certain techniques to alleviate such an effect. In this particular case, the LASSO and Ridge regression techniques use the regularization approach to avoid high

values for model coefficients. The described regularization terms, on one hand, reduce the overfitting effects by shrinking the parameter values to zero. On the other hand, better performances due to parameter selection can also be achieved. As for the polynomial degree 2 technique, in this context it presents low bias error but in some cases a high variance, which can lead to a low performance of the model. By shrinking parameters to zero, the bias is sacrificed towards a lower variance error, and therefore a better performance can be achieved. Additionally, by selecting a subset of the parameters, small details are lost but an holistic overview can be better achieved. Sometimes, large values for the parameters are required to grasp these details, which can hide the effects of other parameters with small values. Hence, regularization impose some size constraints to parameters, allowing for better interpolation among all the parameters. In some cases, details do not bring any additional information to improve the final solution (Friedman et al., 2009).

Based on this, the present sub-section is dedicated to the use of LASSO and Ridge regression techniques to assess if the overfitting effects can be attenuated and a better performance achieved. As described, both LASSO and Ridge are similar to Linear Regression, but a regularization term is added to the parameter optimization function. LASSO uses a L1-norm regularization term, as for the Ridge regression, a L2-norm is used. In Equations 5.6 and 5.7, the λ parameter controls the amount of shrinkage applied. For high values of λ , the shrinkage is also higher, as if the λ decreases to 0, less penalization is applied. Figure 5.15 presents a graphical overview of the effect both regularization terms have in the parameter space. As can be seen, LASSO has a squared constraint region, whereas the Ridge as a circular region. The main difference between these techniques is that Ridge tends to minimize the value of model parameters to be near zero, while LASSO tends to shrink the parameters to zero, discarding irrelevant parameters in the regression problem. The corners of L1-norm of LASSO forces the parameter values to be zero, while the circular shape of L2-norm tends the parameter values to be near zero, but not completely zero.

Hence, these two techniques were used in the deep drawing scenario to overcome some overfitting issues. Therefore, 4 different approaches were applied to NORMEUC and EUC. These are LASSO for Degree 2 and 3 Polynomial, as well as Ridge with the same polynomial degrees. Hence, several values of λ were tested in order to find the best hyper-parameter quantity that minimized the MSE for an increasing number of source models used to generate the target shapes. The strategy used in order to find the best λ was to start with a high penalization, in this case 0.1, and then decreasing it by a factor of 10 until the MSE starts to increase.

Figure 5.16 presents the graphs for LASSO 2 with λ values of 0.1, 0.01, 0.001 and 0.0001 using both EUC and NORMEUC strategies, as well as POL2 for comparison reasons. For the EUC, the best graph is for $\lambda = 0.001$ where lower MSE is observed across all number of source models. In this case for POL2 no overfitting was observed, but even though the LASSO can achieve better performance. As for NORMEUC, the best λ value is 0.01 and the results are very similar to the Linear regression presented in Figure 5.9.

As for the degree 3 polynomial using EUC, as can be seen in Figure 5.17, for low amount of source models is hard to select the best parameter for λ , where both 0.01 and 0.001 oscillate in terms of mean MSE. For a larger amount of source models, the value 0.001 results in the lowest

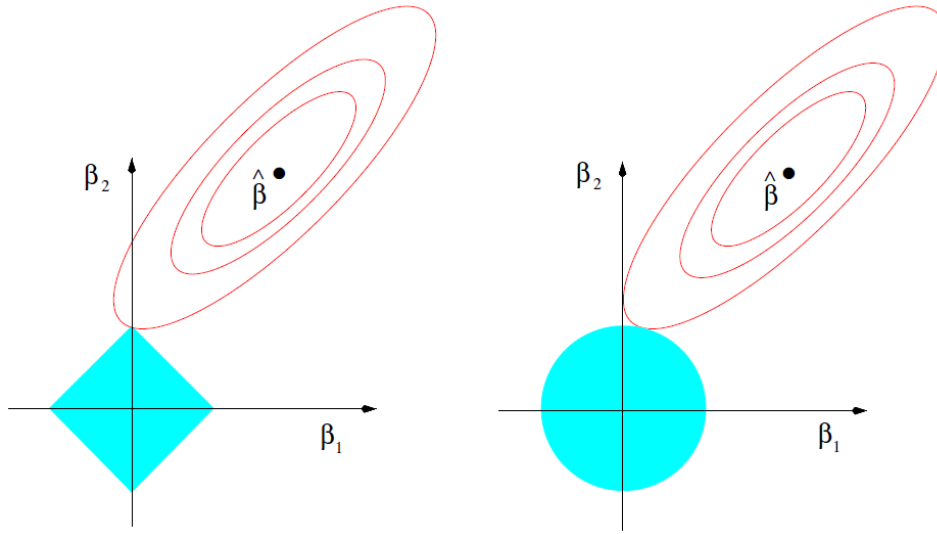


Figure 5.15: Plot of two graphs presenting the parameter space for LASSO (left) and Ridge (right) regression. The blue areas represent the constraints in the optimization function imposed by the regularizers and the red lines are the contours for the least squares error function (Friedman et al., 2009).

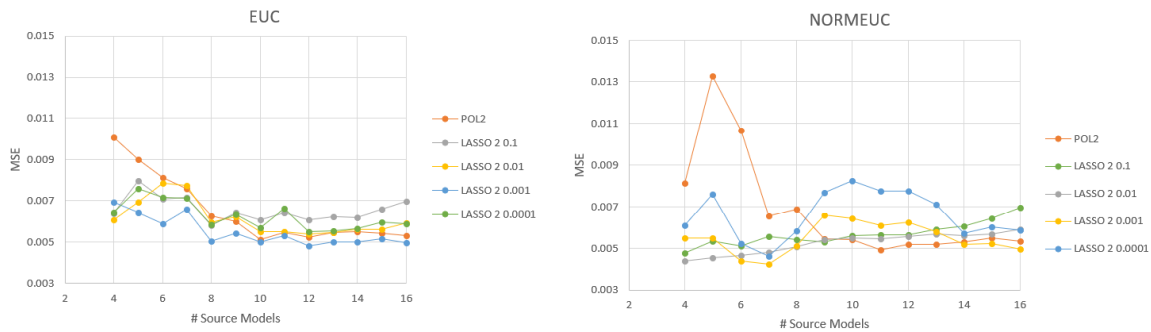


Figure 5.16: Plots of the EUC (left) and NORMEUC (right) strategies for LASSO 2 where multiple values for λ were tested in order to minimize the MSE.

mean MSE, so this value is considered the best for EUC. As for NORMEUC, the best value is 0.01, resembling again with linear regression. Nevertheless, it can be seen that the overfitting effect can be alleviated by the regularization term for low amount of source models, and completely removed for larger values of source models.

For the case of Ridge regression with degree 2 polynomial, for the EUC no improvement was achieved. As can be seen in Figure 5.18, as soon as λ value drops below 0.0001 no regularization is applied and Ridge is the same as POL2. For all the tested values above 0.0001, the results are worse than using POL2. As for the NORMEUC, we can see that 0.00001 is very close to the POL2, and with larger values of λ a lower MSE is obtained but only for a small amount of

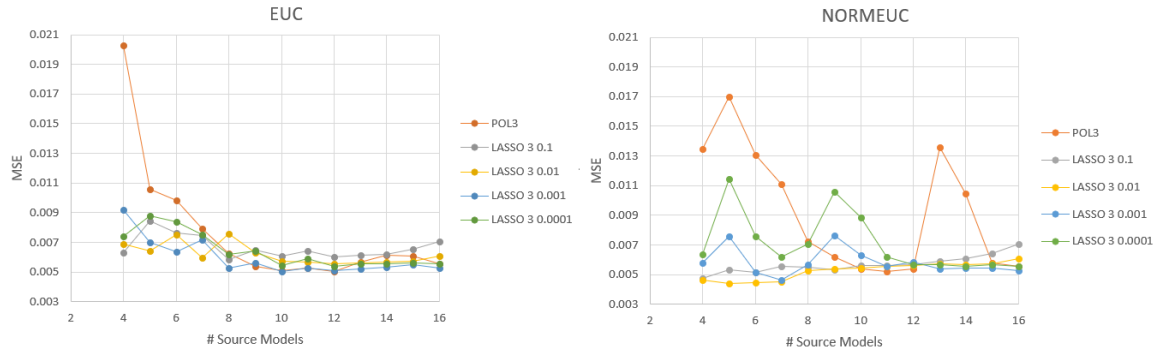


Figure 5.17: Plots of the EUC (left) and NORMEUC (right) strategies for LASSO 3 where multiple values for λ were tested in order to minimize the MSE.

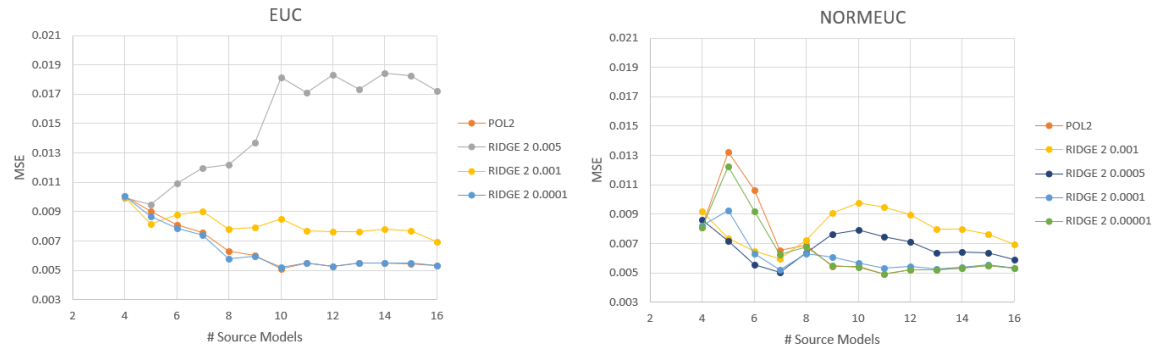


Figure 5.18: Plots of the EUC (left) and NORMEUC (right) strategies for Ridge 2 where multiple values for λ were tested in order to minimize the MSE.

source models, ranging from 4 to 8 source models. From this point, the MSE gotten worse. This behavior occurs because NORMEUC first presents a group of source models with the same value of thickness as the target model, and from 8 (for target thickness of 2) and 9 (for target thickness of 1.5) source models to the whole 16, the other group of source models with different thickness is presented. This means that from a certain amount of source models the same cases but from a different group of thicknesses are presented, turning the problem more complex to model and thus the increase in error around 8 to 9 source models. Hence, the main hypothesis for the presented behavior is that the model complexity of POL2 can support well these new cases from a different group of thicknesses from the target conditions, contrary to the regularized approach of Ridge, where constraints in the parameter space are applied. This is somehow an expected behavior because due to regularization, the model complexity of Ridge is lower than POL2. Hence, for a small amount of models, the simpler is the problem and a simpler model is more suitable. On the other hand, for a large amount of source models, more difficult is the problem and more complex models are required. Therefore, the high MSE in POL2 for a small amount of source models is overfitting and high MSE in Ridge for a high amounts of source models is underfitting, revealing

that the model is not capable to grasp the overall important details of the system. In conclusion, this means that different values of λ should be used with different amount of source models to minimize the MSE.

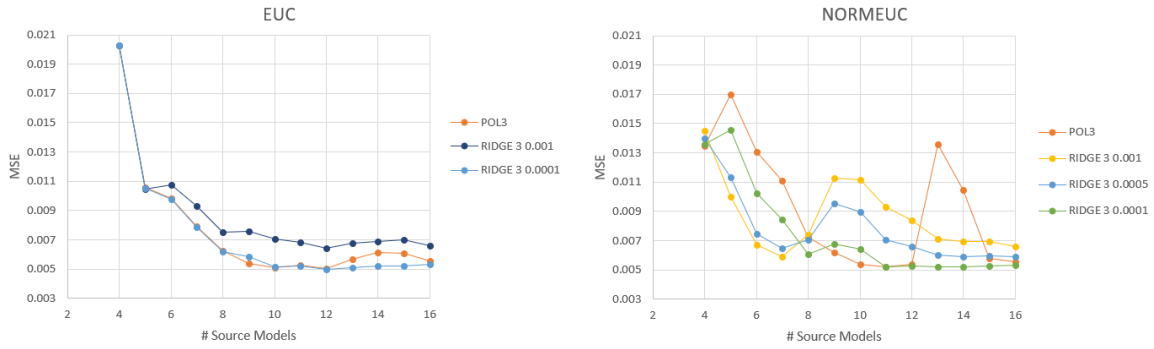


Figure 5.19: Plots of the EUC (left) and NORMEUC (right) strategies for Ridge 3 where multiple values for λ were tested in order to minimize the MSE.

Finally, as for Ridge degree 3 polynomial using EUC strategy, a small benefit exists for λ value of 0.0001 for large values of source models as depicted in Figure 5.19. However, for higher values of λ the performance is worse than POL3. In this case, the graphs for 0.1 and 0.01 are not depicted because the performance is far too worse to be shown in a presentable and understandable way. For NORMEUC, the same behavior as POL2 in the previous case is observed. For small values of source models, the MSE is reduced, but for larger values the MSE increases. Additionally, it can be seen that even for the lower value of λ , the overfitting effect is alleviated and better performances are achieved. Hence, for the best case, the value 0.001 should be used until the same cases of a different thickness group are included in the hyper-model, around 7 and 8 source models, and then 0.0001 is the best choice. Once again, a value very close to 0 would lead to results very similar to POL3, but the Ridge can deal with the existing overfitting, being a good candidate for this situation.

Figure 5.20

In order to prove that LASSO 3 and RIDGE 3 can alleviate the overfitting effect by penalizing high values of the hyper-model parameters, a visual inspection should be made to the deformable parameter space. For this case, process 10 with 13 source models is used with NORMEUC strategy because, as already presented in Figure 5.10, it suffers from overfitting effect when using POL3 where high peaks of MSE occur. Figure 5.21 depicts on the left the deformable parameter values using POL3 technique, on the right the same plot is shown but for RIDGE 3 with $\lambda = 0.0001$ and LASSO 3 on the bottom with $\lambda = 0.01$. Hence, each plot has the corresponding deformable parameters used to train the hyper-model (blue dots), together with the predicted values for all the range of process conditions (green) and the prediction of deformable parameter for the target process condition (orange). As observed, the produced deformable parameter surface for POL3 is very complex and it is clearly seen that is not a good fit and generalization. On the contrary,

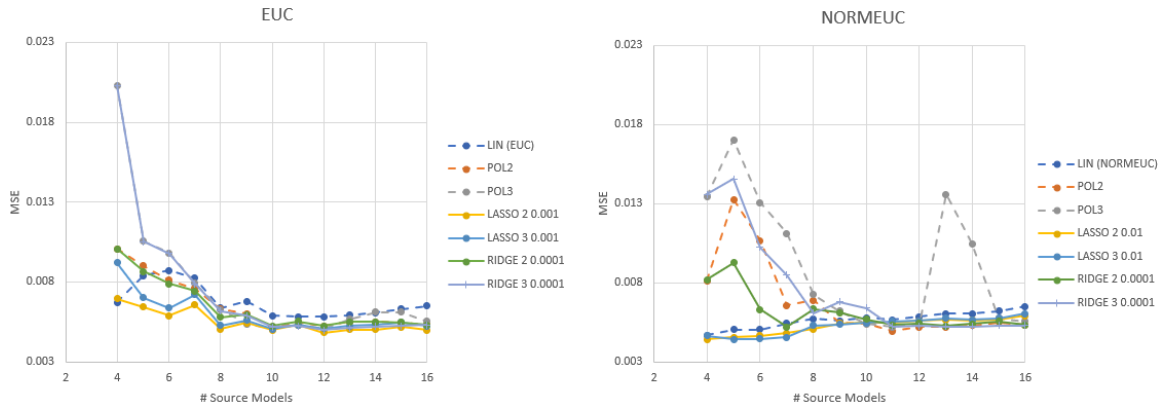


Figure 5.20: Plot of all techniques tested for each of the strategies used to measure distance among process models: EUC on the left and NORMEUC on right. All the penalization λ used for LASSO2, LASSO3, Ridge2 and Ridge3 minimized the overall MSE.

for the RIDGE 3 and LASSO 3, the produced surface is very smooth, simple and it represents and good generalization, clearly grasping the shape of data. For POL3, the MSE obtained for training the hyper-model mapping process conditions into deformable parameters is $1.610e-10$ but the MSE for target shape generation is 0.0738. In this case, the low MSE for training the hyper-model clearly reveals an overfitting effect when a high MSE is observed when generating the target shape. As for Ridge 3, the MSE for training the hyper-model is higher than in POL3 with 0.00893 and the MSE for the generated target shape is 0.00151. In this case, although the MSE is higher for training the hyper-model, it benefits generalization by compromising bias to improve variance error, and the final result is a better shape generation. Finally, for LASSO 3, the training MSE for the hyper-model is 0.0328 and the MSE for the generated target shape is 0.00126, in which the same explanation as Ridge 3 is applicable.

This supports the point explained in the beginning of the section, where it was claimed that sacrificing bias to lower the variance by shrinking the model parameters can, in fact, greatly alleviate overfitting effects. Moreover, it can also be claimed that in the cases where overfitting does not occur, better performances can be achieved by losing some of the details of the data and better grasping the big picture of the problem.

5.2.5 Discussion

From the previously presented results, we would like to highlight two different aspects that significantly influence the system performance. The first aspect is related with the types of source models used to build the hyper-model and generate a new target model. As it was observable from the two strategies that incrementally define which source models should be added to the hyper-model, namely EUC and NORMEUC, the selected models can benefit or hamper the final results. As expected, as the number of models increase, the tendency is for both strategies to converge for the same performance since almost the same models are in both training sets. Although, this is not

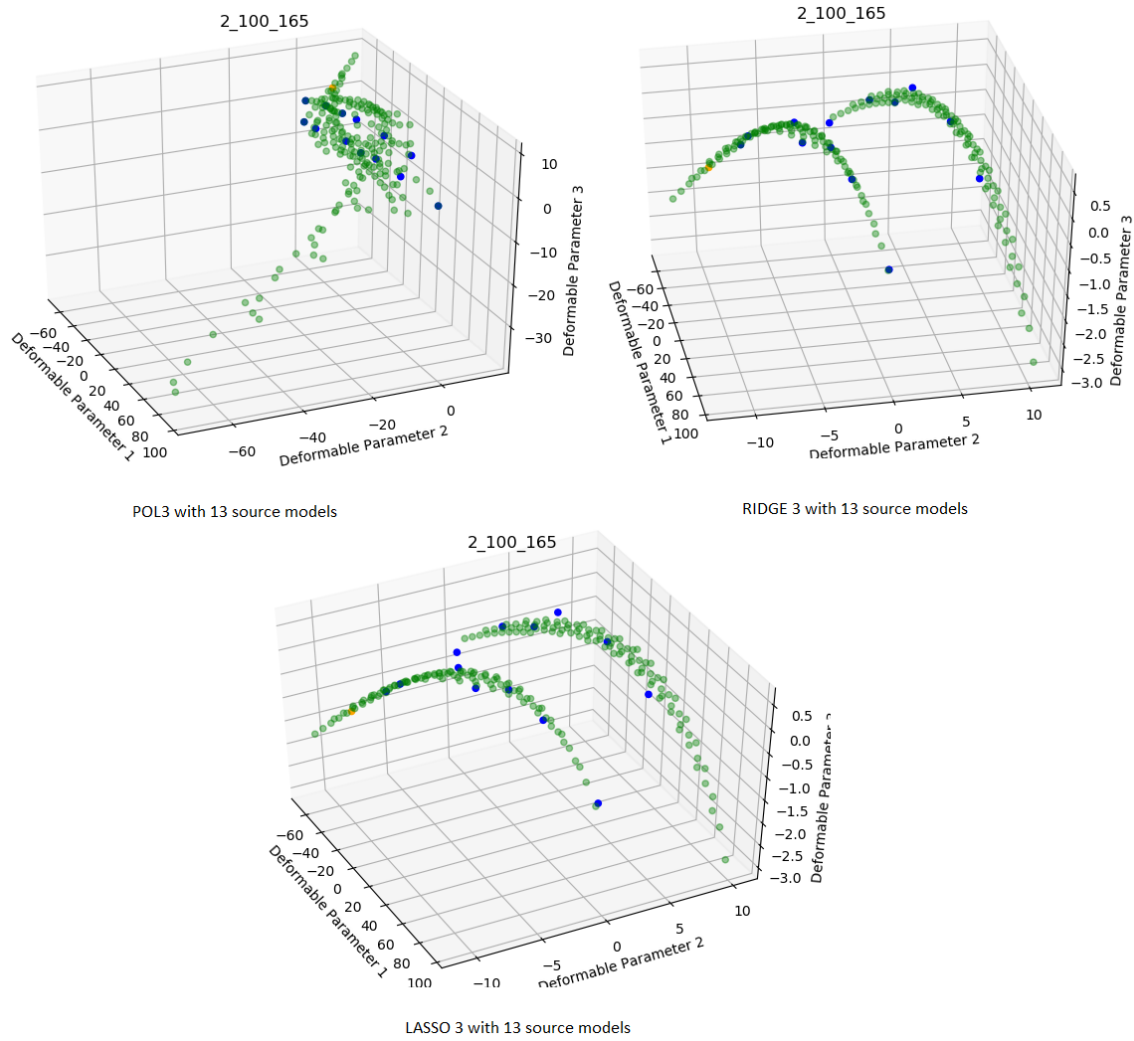


Figure 5.21: Deformable parameter space for both POL3 (left), Ridge 3 (right) and LASSO 3 (bottom) using NORMEUC strategy.

true by using a smaller amount of models. In this case, we observe that in order to maximize the performance of the system, the most suitable strategy for process model selection will benefit the overall system performance. In the present work, only a distance based approach using the process conditions (task descriptions in ZSL) was used, but others might be more suitable depending on the application case. Since one of the advantages of the HPM is to learn a set of unsupervised features based on data shape and modes of variation, the distance among the feature vectors would be a good similarity indication. As can be seen from Figure 5.8, there are some correlations and clusters between the learned deformable parameters. One can take advantage on these to estimate a set of source models to use in the HPM algorithm.

Despite the already described usefulness of HPM in estimating new process models, also the reverse logic can be applied in this case. Instead of predicting the best models to integrate a new

product variant, one could estimate which product should be integrated first from all the available models in HPM and a set of products required to be integrated. In this case, all the process conditions from the not yet modeled product variants can be provided to the HPM algorithm, and a set of deformable parameters predicted. It is based on these predicted parameters that the analysis of the most suitable product to integrate, and not using the process conditions directly. The intuition being such an approach is that a process condition referring to a process model that is close to the already existing process models might be distant in the deformable parameter space. Take as an example the influence of thickness in the previous presented scenarios. As explained, each thickness of 1.5 and 2 yield two different groups in the parameter space, although the thicknesses are not that different. Hence, this approach proposes to measure the amount of influence new conditions might have in the deformable parameter space, instead of considering conditions themselves.

The second aspect that greatly influences the system performance is the possibility of, what we call, *domain drift*. This drift can occur due to the two-stage learning approach for the HPM. In the first stage, the source models should be trained with the available datasets and in the second one, an hyper-model should be trained to map conditions into deformable parameters. From the proposed algorithm, it is not guaranteed that both stages are optimal, leading to domain drift. This effect was one of the limitations from DAP (one of the firsts ZSL algorithms) that was tackled by the ALE algorithm proposed by Akata et al. (2013) by performing a joint learning of input and output embeddings. As referred in the Section 3.2, this joint learning is performed by using bi-linear compatibility functions where the a compatibility among these embeddings is maximized. In the case of deep drawing, this domain drift effect was observed when the trained hyper-model overfitted the data when using both degree 2 and 3 polynomials. This difference in performances from the first to the second stage was addressed mainly with LASSO and Ridge, where the regularized attenuated overfitting and minimizing the domain drift effect.

Meanwhile, if one compares the state of the art methods of ZSL for regression, such as the hyper-model (HM) or TaDeLL, one can easily conclude that these are much more susceptible to domain drift than HPM. Since in TaDeLL or HM only one technique can be used to model all the source tasks to keep the parameters in the same feature space, the probability of getting a sub-optimal model in the first stage is higher than in HPM. As different datasets can have different properties, being ones more complex than others, the same technique can be either a good of bad candidate for training, according to those properties. By taking advantage on HPM being independent from the techniques used to train source models in the first stage, at principle optimal or near-optimal models can be achieved. This eases the problem of domain drift by only addressing overfitting effects on the second stage, which can be tackled with regularized techniques, as previously demonstrated.

5.3 Laser Seam Welding Process

In the previous section the main goal was to assess how the the HPM algorithm performed with a varying amount of source models, and observe if its performance improved as soon as more source models were used. Contrary to this, the main goal of the present section is to assess the performance of HPM in a situation where the best machine parameters should be provided to an operator for the target task, by defining a target process quality. For this case, a real dataset will be used, replicating all the conditions that would be available in the company the provided the dataset. Additional to the proposed algorithm, an optimization problem was formulated to address the process parameter estimation part, where the generated model from HPM was used.

These parts represent the minimal implementation of a system that could really assist an operator in case of integrating a new product variant into the production line. On one hand, it is necessary to have the trained models in order to predict the process quality according to specific parameters. However, this is not sufficient because it is not possible to estimate process parameters according to a desired quality. On the other hand, once the target process model is generated from the HPM, a search algorithm can be used, together with an objective function that minimizes the distance between predicted and desired quality, to find the optimal (or near-optimal) solution to deliver. The accuracy of the models used for this optimization steps is key to find good estimates of parameters. Thus, the best the HPM algorithm can generate target models, the more we are closing the gap between learning the relation among these models and the real dynamics of the process.

For the laser seam welding scenario, the same structure as the deep drawing will be used. First, a set of process models will be trained according to the existing product variants and will act as source models. Additionally, a process parameter estimation approach will be used to derive the best process parameters when specifying the process quality. Afterwards, the HPM algorithm is applied to the current scenario where each of process models will be generated based on the remaining process models. Ultimately, a discussion is made to depict the benefits and restrictions of the proposed algorithm in the context of laser seam welding.

Based on this, the next section will depict all the details about the data used for training and the adopted experimental setup will be given.

5.3.1 Scenario Description

The real manufacturing process in which the HPM algorithm will be applied is called Laser Seam Welding, and was provided by AWL-Techniek company placed in The Netherlands. This process can be described as a laser head mounted in a robotic arm with the objective of welding two metal sheets by issuing radiation from a laser head to a local area where the materials need to be joined. Thus, it creates a melting zone around the laser focus in both sheets, which solidifies when the focus is moved and the material cools down. This produces a continuous welding seam, when the beam is moved along the overlapping sheets at controlled speed. Currently, this process is parameterized using 3 independent variables: Laser power (P) up to 6kW; Focal distance (F) from

the surface in a range of $\pm 10\text{mm}$ and; Robotic arm velocity (V) up to 200mm/s . The quality variables of the process are the weld width (W) and penetration depth (D), both in mm , and can be seen in Figure 5.22. Additionally, and similar to the deep drawing process, also these different product variants are described based on process conditions. In this particular case, there are two process conditions, namely the upper and lower sheet thicknesses. Once again, this implies that if a process model is trained mapping process parameters into quality, this model is only valid for a specific product variant. If new products are required to be integrated, new process models need to be trained due to different process dynamics influenced by the material thicknesses.

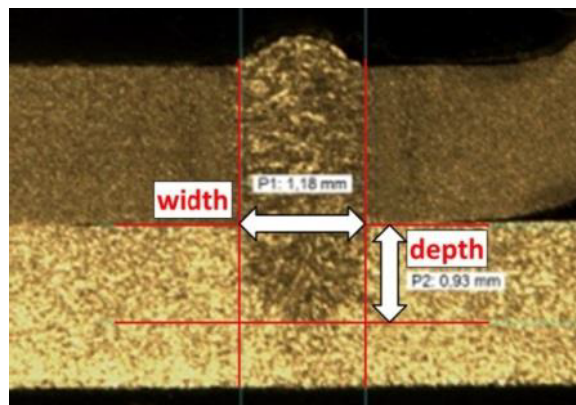


Figure 5.22: Representation of weld width and penetration depth of two metal sheets by laser seam welding of AWL-Techniek.

For this particular process, there are three different product variations and, for each, a dataset was created based on a DoE performed by the company. All the process conditions (upper and lower sheet thicknesses), process parameters (P , F and V), process quality (D and W) and number of experiments are depicted in Table 5.6.

For Process 1, the upper sheet is DC04 steel with 1.5mm thickness and is used together with a lower sheet of HC380LA steel with the same thickness of 1.5 . Process 2 is composed by a 0.6mm thickness of the HC260LA upper metal sheet and a lower sheet of 1.2mm from HC420LA. Finally, Process 3 have a upper sheet of HC420LA with 1.2mm thickness and the lower sheet of HC380LA with a thickness of 1.5mm . For each of the processes there is a high number of samples to use, which is a key requirement to guarantee that the trained model can be a good generalization of the process. Concretely for the ZSL setting, the main idea is to use two of the existing process models to generate the remaining process model, and use the same approach to generate all the processes. The performance of the system will be evaluated by comparing the generated model with the ground truth from the provided datasets, like it was done in the deep drawing process.

However, the HPM algorithm requires that first, process models should exist as source models, and only then the proposed approach can be applied. For this case, the process models will be created using Artificial Neural Networks (ANNs) that will be trained using labeled data from the laser seam welding process. As previously described, the training process is required to use the

Table 5.6: Process Conditions, Process Parameters, Process Quality and Number of Experiments

	PROCESS 1	PROCESS 2	PROCESS 3
UPPER THICKNESS	1.5	0.6	1.2
LOWER THICKNESS	1.5	1.2	1.5
P (KW)	4676.2±666.8	4408.3±742.1	4594.6±702.6
F (MM)	-0.6±12.6	-0.3±14.0	-0.1±12.4
V (MM/S)	104.8±26.5	154.1±32.0	120.9±28.8
D (MM)	0.6±0.3	0.6±0.2	0.6±0.3
W (MM)	0.9±0.1	1.0±0.1	0.9±0.1
EXPERIMENTS	188	260	220

models as data generators and create the corresponding shapes. Since the shapes need to have the same number of datapoints and each datapoint from one shape should match a datapoint from other shape (landmark), these should be sampled from the trained models. If the datasets are all built using the same number of experiments and using the same DoE strategy (same parameters are tested for different tasks), the training step can be avoided. However, this is not the case for the current scenario. As presented in Table 5.6, one can see that different experiments were performed for different processes, meaning that it is not possible to build a deformable model from SSM. While in the last section the data was drawn from the Abacus resulted in datasets with the same sizes and samples for each process, and datasets could be directly used in SSM, in this case this is not guaranteed. Thus, we reinforce the importance of the trained process models acting as generative systems where data can be drawn. This way, if the same inputs are provided to the process models, the outputs can serve as shapes for the HPM.

In the next two sections, the training process of all source models and generating the target models will be detailed, correspondingly. Additionally, the results of applying the HPM algorithm together with the parameter optimization to this real industrial scenario will be discussed.

5.3.2 Process Modeling

For process modeling, first the well known Multi-Layer Perceptron (MLP) was used and tested for a set of network topologies. For that purpose, we have trained an Artificial Neural Network (ANN) using a Multilayer Perceptron where the Adaptive Subgradient Methods for weight optimization (Duchi et al., 2011) was used. P, F and V specify the inputs X_i and D and W define the outputs Y_i of the network, for $i = 1, \dots, N$ where N is the number of experiments per process. All the neurons from both input and output layers have a linear activation function, while in the hidden layers the sigmoid activation function was used. The number of hidden layers and neurons was obtained experimentally through a trial and error process of all combination of number of layers $L = \{2, 3\}$ and number of neurons per layer $M = \{4, 6, 8, 10\}$. An adaptive learning rate was used starting at 0.5 and decreased once two consecutive epochs fail to decrease the training loss by at least $1e-8$, or fail to increase validation score by the same value. For the purposes of training, the input values were normalized between 0 and 1. As a training procedure, a 5-fold Cross-Validation

Table 5.7: ANN Topology assessment in order to find the most suitable model for each process.

Process	ANN Topology	MSE
DC04 - HC380LA	[4,4]	0.012
	[6,6]	0.0091
	[8,8]	0.01
	[10,10]	0.011
	[4,4,4]	0.01
	[6,6,6]	0.0086
	[8,8,8]	0.0097
	[10,10,10]	0.0093
HC260LA - HC420LA	[4,4]	0.0065
	[6,6]	0.0065
	[8,8]	0.0063
	[10,10]	0.0063
	[4,4,4]	0.007
	[6,6,6]	0.007
	[8,8,8]	0.0065
	[10,10,10]	0.0065
HC420LA - HC380LA	[4,4]	0.0087
	[6,6]	0.0088
	[8,8]	0.0088
	[10,10]	0.0092
	[4,4,4]	0.0085
	[6,6,6]	0.0085
	[8,8,8]	0.0086
	[10,10,10]	0.0084

strategy was used where the dataset is divided in training and test sets, with 80% for training and 20% is for further performance testing, where 10% of the 80% from training data is for validation set. This means that the model is trained using the training set, a first assessment is made on the validation set, and a final confirmation is made using the test set. Additionally, as previously explained, in order to stop training the ANN the Model Checkpoint strategy was used to ensure the best convergence and results during training.

All the network topologies assessed are depicted in Table 5.7 together with the MSE to evaluate which one should be used in order to minimize the overfitting effect. As can be seen, the topology for the DC04-HC380LA metal sheet pair is [6,6,6], meaning that the ANN structure as an input layer with the 3 inputs from dataset, 3 hidden layers with 6 neurons each, and one output layer with two outputs. The best performance achieved an MSE of 0.0086. Moreover, for the HC260LA-HC420LA pair the best topology achieved the lowest MSE of 0.0063 with a topology of 2 hidden layers with 10 neurons each. Finally, the last pair of HC420LA-HC380LA required a topology of 3 hidden layer of 10 neurons each to achieve the best performance with MSE 0.0084.

In order to assess the performance of each trained network in a practical way, the best and worse predictions per process are presented in Table 5.8. As can be seen, the best predictions

Table 5.8: Best and Worse predictions for the best ANNs trained for the three presented processes

Process	Case	Variable	Real Value	Predicted Value	MSE
DC04 - HC380LA	Best Case	Penetration Depth	0.22	0.217	1.13e-5
		Weld Width	0.77	0.774	
	Worse Case	Penetration Depth	0.63	0.982	0.0659
		Weld Width	1.18	1.091	
HC260LA - HC420LA	Best Case	Penetration Depth	0.3	0.298	8.55e-6
		Weld Width	0.89	0.893	
	Worse	Penetration Depth	0.88	1.101	0.0315
		Weld Width	1.01	1.127	
HC420LA - HC380LA	Best Case	Penetration Depth	1.15	1.149	1.97e-6
		Weld Width	1.26	1.258	
	Worse Case	Penetration Depth	0.7	0.464	0.0279
		Weld Width	1.08	1.064	

are very close to the real values, yielding a MSE of 1.13e-5, 8.55e-6 and 1.97e-6 for each of the processes. However, for the worse predictions there are some significant differences that require a more closer look in order to assess if the number of errors with high value is frequent or if it only applicable to a small set of cases. For this case, Figure 5.23 presents the MSE prediction histograms for all the presented welding processes. As can be seen, most of the samples are between the range of 0 and 0.02, being the most of them around 0. Thus, we can consider the worse cases presented in Table 5.8 way off the normal pattern observed. Moreover, this supports the presented results in Table 5.7 where a good performance was achieved with the ANN training using the real datasets provided.

As the main purpose of training such models in a real application scenario is to perform process parameter estimation, we will now assess the performance of the model by estimating machine parameters based on specified target quality. Despite now being part of HPM, this parameter estimation is an approach that can be applicable after the generation of models from HPM. By providing a set of quality measures representing customer requirements, through the formulated minimization problem and search algorithm, the best process parameters can be found. In this context, the process qualities defined by the customer are the weld width and depth yielding more robust or fragile welds in the final product. Different customers might have different requirements depending on the product application. One might only want to join metal sheets for aesthetics, where not a strong joining is required when compared with a car chassis that should be as strong and robust as possible in the automotive industry.

Hence, the process models provide a prediction of a certain X (process parameters) finding the most suitable Y (process quality). Contrary to this, in the process parameter estimation one should provide the desired process quality Y in order to find the best process parameterization X . This means that we can specify the customer requirements and obtain the optimal or near optimal machine parameterization.

A set of optimization processes were made in the trained models to assess how close the parameterization found is from the ground truth. As defined in the experimental setup for the

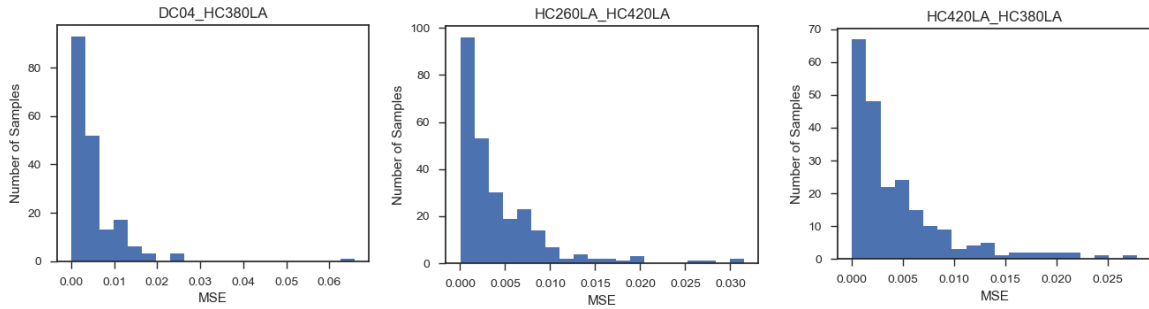


Figure 5.23: MSE Prediction histograms for the three ANNs trained for each of the processes.

training process where a test set was used to evaluate the models' performance, also the parameter optimization will be performed using the test sets for each welding process. This assures that the fairest tests are made and validates if these models are suitable for a practical application.

For the process parameter estimation the Basin-Hopping algorithm (Wales and Doye, 1997) was used to find the most suitable machine parameters by minimizing the difference between the target and optimized process quality. The Basin-Hopping (BH) algorithm was first introduced by Wales and Doye in 1997 (Wales and Doye, 1997) to study the lowest-energy structures of Lennard–Jones clusters consisting of up to 110 atoms, and is based on the Monte-Carlo algorithm and gradient-based local search. It is therefore a stochastic algorithm aiming to find the global minimum of a certain function (in this case a loss function), and is mainly based on the following steps: 1) Random perturbation of the coordinates to be tested in the provided function; 2) Step towards the local minimization of the solution; 3) Reject or Accept the proposed coordinates based on the minimization step. As for the acceptance test, the Metropolis criterion is used from the Monte-Carlo application.

As for the optimization process, the process models are used to iteratively assess a set of process quality values according to a certain process parameters produced by the optimization algorithm. Since these process models will be used and the normalized dataset between 0 and 1 was used in the training set, we have constrained the solutions generated by the algorithm also between 0 and 1. As an initial guess of a solution, we have set the value to 0 for each of the parameter. Regarding the problem formulation, we aim to minimize the difference between the target of process quality and the solution generated, and for that purpose the MSE (which is similar to the sum of squared error because is used only on one sample) was used to assess these differences. Therefore, Equation 5.10 defines the minimization problem:

$$\begin{aligned}
 x &= \arg \min_x L(\hat{y}, y) \\
 &= \arg \min_x \sum_{i=1}^n (y_i - \hat{y}_i)^2
 \end{aligned} \tag{5.10}$$

Table 5.9: Process Parameter Optimization for the best 3 solutions in the test set that minimizes the distance between optimized and real quality.

Process	Case	Parameterization			Quality		MSE	MSE
		P	F	V	Depth	Width	Parameterization	Quality
DC04 - HC380LA	Real	3500	-10	100	0.14	0.67	0.362	1.828e-7
	Optimized	5500	-20	127.18	0.14	0.67		
	Real	5000	-10	80	1.1	1.05	0.079	5.99e-7
	Optimized	5100.02	8.97	61.72	1.09	1.048		
	Real	5000	5	160	0.15	0.72	0.021	6.003e-7
	Optimized	5385.53	11.45	153.46	0.149	0.718		
HC260LA - HC420LA	Real	3500	-20	80	0.67	1.25	0.0711	3.848e-10
	Optimized	4264.57	-19.96	124.13	0.67	1.25		
	Real	4500	-20	130	0.65	1.2	0.016	3.911e-10
	Optimized	4934.01	-20	138.86	0.65	1.2		
	Real	4000	15	170	0.21	0.71	0.414	1.495e-9
	Optimized	5333.16	-18.79	220	0.21	0.709		
HC420LA - HC380LA	Real	4000	-15	90	0.87	1.07	0.023	2.11e-9
	Optimized	3500	-15.41	75.85	0.869	1.069		
	Real	4500	15	90	0.97	1.2	0.345	1.058e-7
	Optimized	3500	-20	65.13	0.97	1.199		
	Real	4500	15	120	0.47	1.12	0.281	1.968e-7
	Optimized	5500	-15.38	142.98	0.469	1.12		

where x is the machine parameterization, y is the target process quality to be achieved and $\hat{y}$ is the process quality predicted by the process model trained with experimental data.

To that intent, and similarly to the tests performed to the models, the best optimizations will be depicted. However, we must clarify what is a good or bad optimization process in this context. Intuitively, one might think that a good optimization process is just to find a certain process parameterization that yields the closest process quality considering a defined target. This way, the objective is to minimize a loss function that calculates the distance between what the model produces and the provided target. Intuitively, as this distance is close to 0, that best the process parameterization found is. However, this is not very useful if the solution minimizes the distance from the target if the real parameterization is not close to the real application, or if it is even out of the parameterization bounds. This means that, in order to correctly evaluate the process parameter optimization process using the trained process models, not only this distance from the target should be considered, but also the process parameters from the ground truth, in this case, the test set.

Hence, first Table 5.9 depicts the best solutions that minimize the distance between the target quality and the optimized one for all three processes, along with the process parameters to be suggested to the operator in a real application, and both MSE for process parameters and process quality. As can be seen, the obtained MSE for the process quality is very low, meaning that the algorithm used for the optimization process is very effective in finding the global optimum solution. Complementarily, Figure 5.24 presents the histogram for each process with number of samples in relation to the MSE between target and optimized quality where can be seen that most of the MSE samples are near the value 0, which represents more than 70% of the total test samples in the 3 processes.

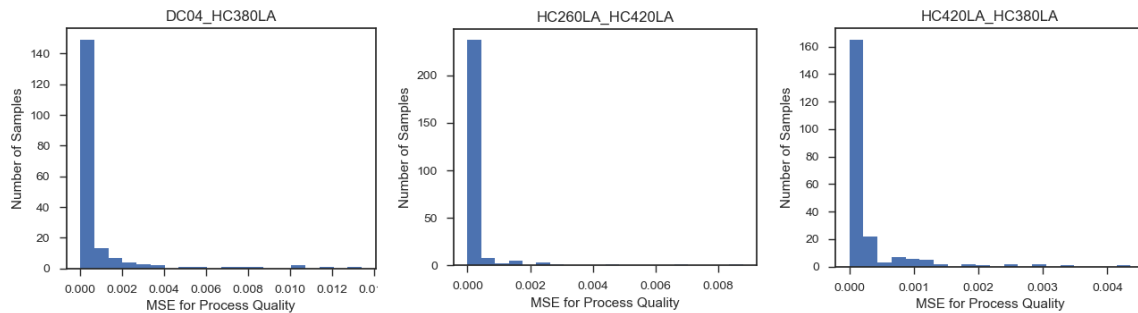


Figure 5.24: Histogram depicting the number of samples in relation to the MSE for the target and optimized quality.

Moreover, Table 5.10 presents the 3 best solutions considering the MSE for process parameterization, where a more balanced trade-off between MSEs is achieved. We can see that the presented solutions are near in both process parameters and process quality, being the ideal case in a practical application where a shop-floor operator can truly rely on what the system advises him to do. Hence, in order to understand if these results are frequent in the test set, Figure 5.25 depicts the histogram for each process with the MSE between desired and optimized process parameters. It can be seen that again that the majority of the samples are around 0, meaning that the process model, together with the optimization technique are capable of indicating a suitable machine parameterization according given a certain process quality. However, there are some samples with higher errors, also revealing that the process model, for a small amount of data points, is not capable to provide a good indication of machine parameters.

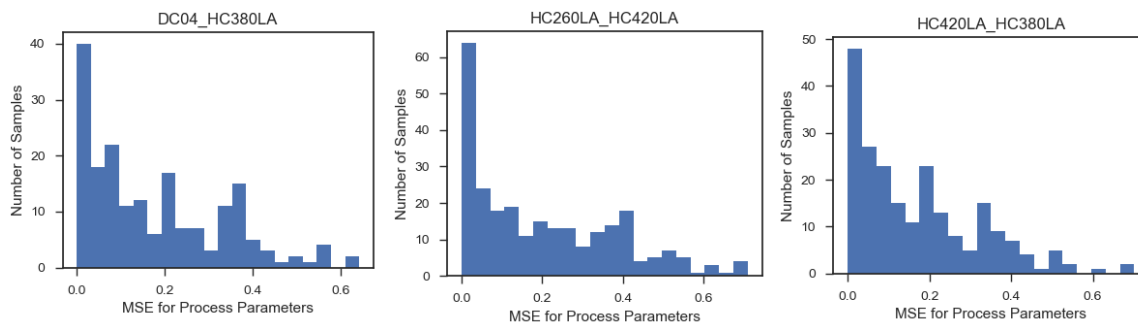


Figure 5.25: Histogram depicting the number of samples in relation to the MSE for the desired and optimized parameters.

The process parameter estimation is considered a very useful process to assist shop-floor operators when calibrating a certain machine towards certain quality goals defined by the customer. However, these process models require a certain amount of experiments that should be collected before the system can be used in a truly useful way. What happens if a new product variant needs to be produced in the same machine, but experiments are not available to train a process model?

Table 5.10: Process Parameter Optimization for the best 3 solutions in the test set that minimizes the distance between optimized and real parameterization.

Process	Case	Parameterization			Quality		MSE Parameterization	MSE Quality
		P	F	V	Depth	Width		
DC04 - HC380LA	Real	3500	-10	80	0.69	1.01	9.688e-5	1.349e-5
	Optimized	3500	-9.77	82.73	0.689	1.004		
	Real	5500	20	120	0.12	0.73	1.655e-4	2.405e-4
	Optimized	5500	20	123.78	0.138	0.718		
	Real	4500	5	120	0.44	0.94	4.657e-4	1.613e-4
	Optimized	4437.9	4.36	122.28	0.422	0.937		
HC260LA - HC420LA	Real	3500	-20	100	0.57	1.2	3.285e-5	1.743e-4
	Optimized	3500	-19.99	101.68	0.573	1.181		
	Real	5500	10	220	0.48	0.86	4.501e-5	4.66e-4
	Optimized	5500	10.46	220	0.46	0.883		
	Real	3500	-20	120	0.41	1.07	8.348e-5	3.267e-4
	Optimized	3500	-20	122.69	0.43	1.05		
HC420LA - HC380LA	Real	3500	20	60	0.93	1.21	1.55e-6	2.095e-5
	Optimized	3500.55	20	59.63	0.934	1.215		
	Real	3500	-20	70	0.76	1.15	3.778e-6	3.86e-6
	Optimized	3500.76	-20	70.56	0.762	1.148		
	Real	3500	20	90	0.23	0.64	2.075e-5	3.883e-5
	Optimized	3500	20	91.34	0.222	0.645		

Can we take advantage on the existing process models to generate a process model that corresponds to that product variant and afterwards perform parameter estimation? In the next section, we will be applying the HPM algorithm to address these questions.

5.3.3 Process Model Generation

This section presents what is called as process model generation, where we will be applying the HPM algorithm to the presented case of laser seam welding. Thus, we are addressing the case where a new product variant needs to be introduced but no process model exists in order to perform process parameter optimization. In the current HPM application on the laser seam welding scenario, the main objective will be to solely use the information about two of the processes, and generate a third process model based only upon specific new process conditions. For example, one of the experiments will use Process 1 and 2, together with the corresponding process conditions to estimate Process 3. In this context, only using two models represents a great limitation because no nonlinear relation between the models can be grasped if only two samples exist, being necessary at least 3 samples to model these nonlinearities. This means that it does not matter how good the machine learning algorithm used is to train the hyper-model, because the data itself does not reveal such nonlinearities. Based on these limitations, and once the hyper-model is trained, the evaluation is performed by comparing the process model trained with experimental data with the generated one from the Hyper-Model itself. This means that the generated model will try to predict the Depth and Width of the weld according with the true measurements.

As for the first step for the HPM algorithm, similar to the deep drawing case, we need to sample the same number of data for all the process models as a way to create a set of suitable

shapes for the deformable model. For that purpose, we performed a full factorial design on all the 3 process models available to generate shapes with 800 samples S for all the input space P , F and V between 0 and 1 (due to normalization), and all the predictions of each process models were registered resulting in new 3 datasets, each one constituting a shape for the SSM. Based on this, we have only considered the output attributes of D and W as shapes because all the input values are the same in the datasets generated and bring no additional statistical information. All the generated shapes are necessary to create the deformable model and calculate the corresponding b parameters. Since we have two output attributes, we can consider these as 2-d shapes $x = [D_1, \dots, D_S, P_1, \dots, P_S]^T$. By using Equations 2.8 and 2.13 the mean shape $\bar{x}$ can be calculated, along with the eigenvectors ϕ and eigenvalues λ . As for the SSM, the number of features for each shape is the number of samples multiplied by the number of dimensions used for the shapes, so we get $800 \times 2 = 1600$ features, which means 1600 eigenvectors and eigenvalues. To avoid handling such number of eigenvectors, only the first eigenvectors (ordered by decreasing order of corresponding eigenvalues) that represent more than 95% of the variance are chosen. Due to the limitations of using only two processes to generate a new one, only one eigenvector maximizes the variable in the data. Hence, we get the eigenvector ϕ and mean shape $\bar{x}$. Afterwards, the deformable parameters b can be calculated using Equation 2.15 which are the process model parameters to be used in the hyper-model training. As detailed in Chapter 4, these process model parameters b together with the process conditions ζ are the essential parts to build up the hyper-model. Basically, in this case, the hyper-model should map a 1-dimensional feature vector (process model parameters) into a 2-dimensional feature space (process conditions). For this case, two independent models were trained, mapping the 1-d process model parameter with each process condition. As presented in Equation 2.4, the idea is to train a model that is able to map the process model parameters into process conditions. To that intent, and in order to create this hyper-model, a Linear Regression Model was trained. This maps the deformable parameters of the SSM into a pair of process conditions.

Once the hyper-model is created, the main focus now is to be able to create a new process model based on new available process conditions. As already discussed, all the processes will be generated based on the remaining two. According to the proposed algorithm, we need now to infer the process model parameters based on new process conditions ζ' . For that purpose, the Nelder-Mead optimization algorithm was used to minimize an objective function, which in this case is the distance between the new process conditions and the process conditions generated by the hyper-model, by using the Mean Squared Error (MSE). In sum, the optimization algorithm uses the downhill simplex method to exploit the search space trying out multiple process model parameters that near-optimally can generate the process conditions that minimizes the objective function. The search for these process model parameters is the crucial step of the whole methodology.

In Table 5.11, the deformable parameters calculated by the SSM using only one eigenvector for all the processes are detailed. As observed, the Nelder-Mead algorithm can find the corresponding deformable parameter for each of the processes. There are *Optimized* and *Training* values for each of the scenarios, representing the values from the optimization process to find the

Table 5.11: Process parameters and process conditions used for hyper-model training, and optimized process parameters according to the defined process conditions.

Process	Process Model Parameters (Deformable Parameter)	Process Condition	
		Upper Sheet	Lower Sheet
DC04 - HC380LA (Optimized)	-14.058	1.5	1.5
HC260LA - HC420LA (Training)	-7.810	0.6	1.2
HC420LA - HC380LA (Training)	7.810	1.2	1.5
DC04 - HC380LA (Training)	4.151	1.5	1.5
HC260LA - HC420LA (Optimized)	-20.759	0.6	1.2
HC420LA - HC380LA (Training)	-4.151	1.2	1.5
DC04 - HC380LA (Training)	-9.644	1.5	1.5
HC260LA - HC420LA (Training)	9.644	0.6	1.2
HC420LA - HC380LA (Optimized)	-3.857	1.2	1.5

deformable parameters for new conditions, and the deformable parameters from existing process models, correspondingly. Since there are only two source process models, the resulting deformable parameters have 0 mean, meaning that the values are the same but one is negative and the other is positive. The farthest process from the others is the HC260LA-HC420LA with upper and lower sheet thicknesses of 0.6mm and 1.2mm, resulting also in the farthest predicted deformable parameter value of -20.759 when compared with 4.151 and -4.151. In the prediction of processes DC04-HC380LA and HC420LA-HC380LA, it can be seen that the predictions are most closer to the source deformable parameters due to the similarity in process conditions. As farthest is the process in terms of process conditions, as farthest if the deformable parameter prediction. Since the SSM is only based on shape analysis, this correlation means that the chosen process conditions in the first place are good meta-descriptors of the laser seam welding process. For the process DC04-HC380LA, the Nelder-Mead algorithm took 68 iterations with an error of $9e-3$, for the process HC260LA-HC420LA the algorithm took also 68 iterations with an error of $4.5e-2$, and finally for process HC420LA-HC380LA it took 65 iterations until it found the minima, and the error value for the solution is $4.5e-3$.

In order to obtain a new process model that suits the new process conditions, the hyper-model needs to be representative enough and a good generalization of the machine process domain, otherwise the process model parameters obtained from the optimization process will lead to a bad new process model with no practical use at the shop-floor level. As the hyper-model was trained only using two previous existing processes, which means the only one parameter needs to be optimized, the task of finding the most optimal value is simple. Since the model is linear, the resulting optimization curves are convex and the local minima corresponds to the global minima as can be seen in Figure 5.26. This matches with the calculated values presented in Table 5.11 resulting from the optimization technique and can also be confirmed in the referred Figure.

After finding the most suitable b values for the new process conditions, the new shape needs to be generated using Equation 2.14 that defines the deformable model. This way, the new shape x' can be created, and a new process model can be trained using a machine learning technique.

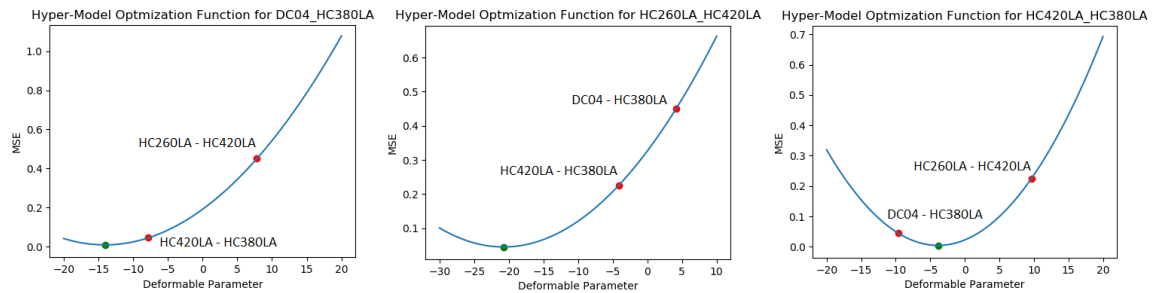


Figure 5.26: Optimization function for the hyper-model in the laser seam welding case for all the processes.

This way, a new process model can be obtained by solely specifying the process conditions and used by the operators that intend to calibrate a specific machine by performing process parameter estimation according to well-defined process goals from customers.

The main outcome of this experiment is to evaluate how good a process model is when generated using the HPM algorithm compared with the process model trained using the real data from the laser seam welding process. As referred in the beginning of Section 5.3.2, the metric used to compare the generated process model with the one trained from real experimental data is the Mean Squared Error (MSE). To make a valid comparison between the models, and as described in the previous section, a new process model was trained using the generated data from the hyper-model. The resulting model was then used to predict the process quality values using the same process machine parameters from the experimental dataset. This way, a fair and valid comparison between the models can be performed.

Table 5.12 presents the MSE for both models, one trained using data from the dataset, and the other with generated data from HPM algorithm. As expected, the generated process models are not as good as the process model trained with real data. For process DC04-HC380LA the MSE achieved was 0.048 when compared with 0.0086 from the process model trained with real data. For process HC260LA-HC420LA, which was by far the worse model, the MSE achieved was 0.22 compared with 0.0063. This is in line with the previous analysis that this process is the farthest from the remaining ones, and thus, the difficulty in predicting a reliable shape in SSM. Finally for process HC420LA-HC380LA, which was the best generated model, the MSE achieved was 0.02 compared with 0.0088. The high error obtained in the first two processes are mainly due to the complexity of relation among the models that is not possible to be grasped by only using two source processes.

This kind of approach allows for a first glance of how the dynamics for the new process looks like, and a better understanding of how distinct it is from the other processes that yield different product variations. Figures 5.27 and 5.28 plots a scatter matrix for the process models trained with real data and the generated model, correspondingly, for process HC420LA-HC380LA. At this stage, the results obtained for the other processes were not encouraging, so all the further

Table 5.12: Comparison between the generated model and the process model trained with real experimental data

Process	Model	MSE
DC04 - HC380LA	Real Data	0.0086
	Generated	0.048
HC260LA - HC420LA	Real Data	0.0063
	Generated	0.227
HC420LA - HC380LA	Real Data	0.0084
	Generated	0.0188

analysis will be performed for the HC420LA-HC380LA process . As can be seen, the shape of the data from the generated model is fairly maintained when compared to the real model, capturing the dynamics of the real process with a certain degree of confidence.

From the all experiments performed to achieve these results, we truly believe that the accuracy of the generated process model can be higher if more process models are used to yield the hyper-model, as demonstrated previously in the deep drawing scenario. For this particular case, two process models were used to generate a third one, which we consider the minimum number of models that should exist for this purpose. This way, we can consider the present case as the worse case scenario of accuracy that a generated model can achieve, and still, a fair representation of the new process is generated. From this perspective, we believe that the obtained results are encouraging and promising to pursue a more thorough research related with the proposed algorithm with real experiments and data, not only in the industrial panorama, but also in other application areas.

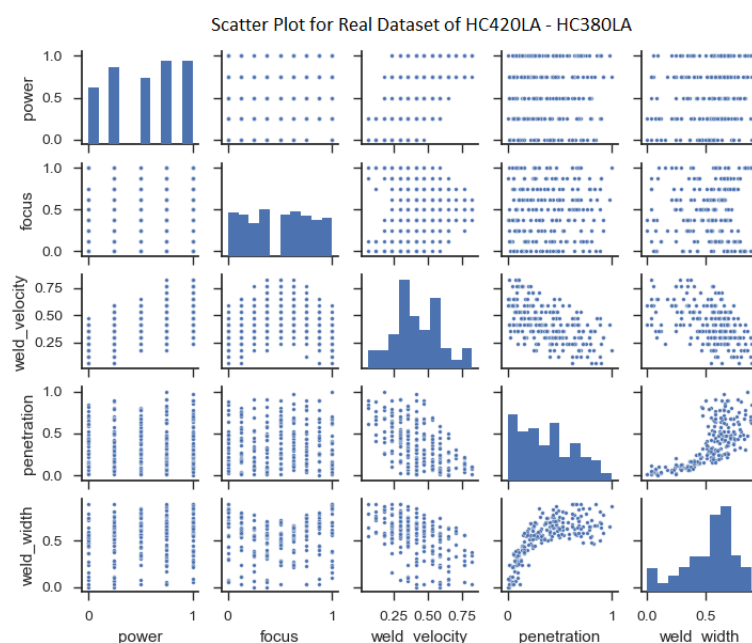


Figure 5.27: Scatter Matrix Plot of the real experimental data for process HC420LA - HC380LA

Table 5.13: Process Parameter Optimization in the generated process for the best 3 solutions in the whole dataset that minimizes the distance between optimized and real parameterization.

Process	Case	Parameterization			Quality		MSE Parameterization	MSE Quality
		P	F	V	Depth	Width		
HC420LA - HC380LA	Real	5500	-20	140	0.27	0.77	2.01e-4	2.024e-5
	Optimized	5458.87	-20	137.71	0.274	0.765		
	Real	5500	20	150	0.16	0.69	3.344e-4	3.021e-4
	Optimized	5500	20	144.61	0.17	0.667		
	Real	3500	-10	100	0.57	0.97	9.6e-4	2.04e-4
	Optimized	3500	-9.66	90.98	0.588	0.962		

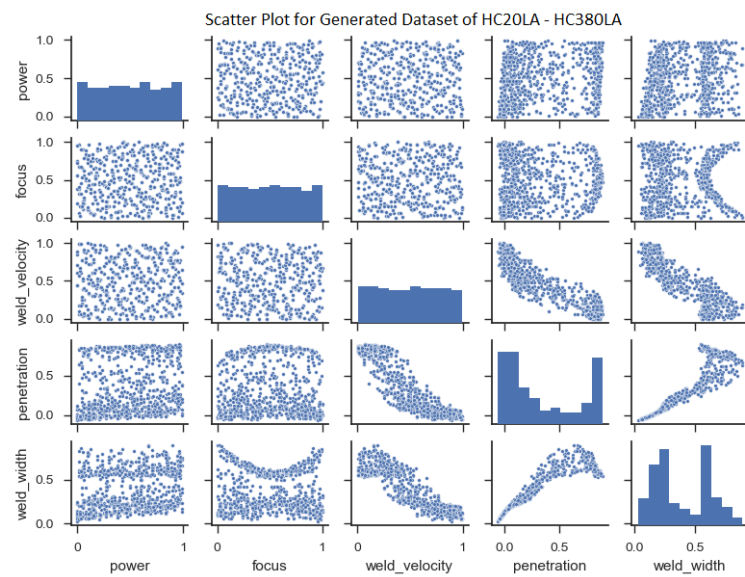


Figure 5.28: Scatter Matrix Plot for the generated process model for process HC420LA - HC380LA

The last step of the process generation section is to assess how the generated model would perform in a real industrial environment. For that case, we will perform the process parameter estimation, similarly to Reis and Gonçalves (2018b,a), only for the generated process that yielded the best results of MSE, namely the HC420LA - HC380LA. Then, we will compare how good the estimation of process parameters are in regard to the real parameters that the operator should use. Hence, all the samples from the available dataset will be used, instead of only the test set. Since the process model was trained only using generated data, no overfitting is possible to occur, making the estimation procedure valid for the entire real dataset.

In Table 5.13, the best three estimations of machine parameters to use for process HC420LA-HC380LA were compared with the ground truth. The presented table is similar to Table 5.10, where the *Real* cases represent the actual parameters used at the shop-floor, and the *Optimized* cases are the ones estimated from the generated model. As can be seen, the MSE for machine

parameterization is low indicating that the generated model in fact grasped some of the target task dynamics from the two source tasks. In this case, the closest estimation from process parameters is presented, and not the closest process quality. As previously explained, analyzing the distance between predicted and true process quality only guarantees that the search algorithm can converge towards a solution that minimizes this distance, and that the found parameterization is in fact viable. Opposite to this, by analyzing the distance between estimated and true process parameters a more reliable assessment is made because it would be in fact what the operator would use at the shop-floor for machine calibration, if such a system is deployed. From this, the depicted results show a great potential for this algorithm even with a small number of source process models used. However, the only way to truly verify these results is by the use of such a system in a real industrial environment.

5.3.4 Discussion

As described throughout the present work, the real usefulness of the proposed approach is to estimate machine parameters upon specified quality targets. Hence, the best the generated model is, the best the optimization procedure will find suitable and reliable parameters for the operator calibrating the machine. This very same strategy to estimate parameters is being applied to models trained with a complete datasets (Reis and Gonçalves, 2018b), it should also be applicable to generated models. However, as it could be observed, if the number of models are not sufficient to grasp the dynamics from the relation among the tasks, it is difficult to generate a model with high accuracy for the target task.

In such cases, by performing a small amount of experiments in the target process might, significantly improve the system accuracy through the use of other Transfer Learning Techniques. As explored and detailed by Reis and Gonçalves (2018a), the authors use an Inductive Transfer Learning approach to improve the system accuracy based on these small experiments performed in the target process to be integrated at the shop-floor. First the authors trained the closest process model to the target process using an ANN from a complete dataset of experiments, and then transfer the acquired information to a new model where it was fine-tuned by using this small amount of data. The transfer process was achieved by creating an ANN with the same topology as the source model, and initializing the target model with the same weights as the source. Hence, in order to learn the specific details about the new task, the last layer was initialized randomly. For the training procedure, a low learning rate was used, avoiding catastrophic forgetting, to train the new network with different amounts of experiments. The amounts used were 8, 24, 67 and 125 in order to assess the benefit of transfer as soon as new experiments are introduced. In this work, with the small amount of 8 and 24 experiments, already significant performance was achieved for the target process model. However, this is all dependent on the complexity of processes and the which experiments are performed for the target process. In this work, the authors have used a full factorial design approach to perform the 8, 24, 67 and 125 experiments to transfer. If the experiments performed do not follow a well established approach from DoE, most probably the transfer will not yield good results.

For the case of the HPM, the generated model could act as source, instead of already existing sources, and by providing this small number of experiments the performance might be higher. If so, ZSL could then be a complementary step to improve the performance of systems with a small amount of data available, by training a model that might yield reliable predictions.

Chapter 6

Discussion

In the last decade, the interest for ZSL area has been growing due to its potential in solving a specific target task where no data about its dynamics is available. As discussed, this is normally achieved with a high-level description of the task to solve together with data about other related tasks. This area is particularly interesting because it allows to explore a set of new tasks that are outside the domain it was trained to operate. Specifically, for image classification, it is particularly useful to classify images where the corresponding class was not used in the training process. This kind of system closely relates with the capacity for humans to identify a certain object while only given a high level description, such as a person by its height, hair color and age, or a house by the numbers of floors, its color, if it has a garden or a pool. In order to achieve more human-like behavior, a relation between images and classes needs to be decomposed for further estimation of new, unseen classes according to provided images. Despite the image classification being a suitable application of ZSL methods, there are a multitude of applications where this approach can be applied. From chemistry to industrial scenarios, the problem of predicting and estimating outside what one once learned is a situation that often challenges humans due to the uncertainty about future tasks and inexperience of those tasks to come. To address this challenge, one relies on past experience and intuition to make the best decision possible. Developing a system that can address such a problem and assist humans in taking the next step, is the main goal of ZSL.

It is based on this goal that the present thesis lies its foundations. As a main motivation for assisting the integration of new product parts in an industrial scenario, a ZSL algorithm was proposed. From the review of current state of the art techniques, to the proposal of HPM algorithm and all the experimental setups defined for its assessment, the main goal is always assisting humans to perform a more effective and efficient job, or even replace them in case of high-risk situations. Hence, three different scenarios were defined to validate the effectiveness of HPM. First, a more intuitive and simpler scenario was set by making use of Beta Distribution to generate a set of curves with different shapes. Associated with each curve, there are two parameters α and β that dictate these shapes. The main idea behind such a scenario is to, first, make a simple introduction to the proposed algorithm, and second, highlight the main differences from state of the art ZSL methods applicable to regression problems. From this, the limitations of using model coefficients

as meta-features to build a hyper-model (Pollak and Link, 2016) are depicted, together with the advantage of using an unsupervised approach based on shape analysis. Ultimately, a visual and quantitative inspection about the hyper-model and HPM was made, supporting the hypothesis that using model coefficients limits the potentiality of ZSL for regression.

Afterwards, a scenario based on the deep drawing industrial process was defined, making use of 2-d simulation models of real mechanical dynamics from Abacus Simulator. The main idea of this scenario was to assess the effectiveness of HPM with a varying number of source tasks. In the scenario, two different aspects were observed and discussed. The first one was the observation that the HPM performance increases according to the increase number of source models used, as expected. However, it is highly dependent on which models are used. For that, two strategies for adding models to the HPM were used, which ultimately led to different performances over the increased number of source tasks used. This confirms that the learning process in fact occurs and the use of different source tasks that can be closer or farther way from the target one influences the system performance. The second aspect observed was that overfitting can occur once training the hyper-model, leading to domain drift due to the two-stage learning process. This effect was mainly due to the use of a training model with high complexity and small number of samples (source tasks) to train the hyper-model, where normally in complex problems, to reliably train a model, a significantly higher number of samples needs to be provided. Nevertheless, the overfitting effect was alleviated by the use of regularizers that avoided large numbers on the polynomial coefficients, minimizing the model variance by compromising the bias.

Finally, the last scenario is based on a laser seam welding case where real experiments were made in three different product variants for the same machine. In this case, not only the HPM algorithm was used but a complete system was developed that can assist the operator in a real scenario. For that, an additional optimization problem was defined to estimate machine parameters according to a desired quality. The main goal of this scenario was to test if a complete system could be built and used to assist the integration of new product variants into the production system based on a real dataset. This way, since there were only 3 datasets from different processes, two of them were used to generate the third. As expected, with only two source models, it was observed that the generated models for the target task were not as good as the real ones trained with the full dataset. However, these could be used to provide an initial guess to the shop-floor operator from where to start performing machine experiments, instead of starting without an initial clue or from a parameterization of a different product variant. Additionally, from the results obtained in the deep drawing scenario, it is foreseen that a better performance can be achieved if more source models were available.

As already described in Section 4 when presenting the HPM, one of the advantages of the proposed approach is the shape analysis it performs to the data itself using a deformable model. By analyzing how data varies from model to model, an unsupervised feature space that represents the main modes of deformation was built and used to generate back new data shapes based on new process conditions. Comparing to the hyper-model (HM) or the TaDeLL approaches that use the model parameters directly, using data instead allows for a more detailed analysis of system dy-

namics. Despite the model coefficients are also related with the shape of data, in certain situations the model can overfit or underfit the data. Since the main restriction of the HM is the assumption that a "model-fits-all" datasets, the same technique will have different performances according to the dataset in hands. However, for fairness of comparison, we should highlight that the effect of overfitting can be managed by the use of regularized methods, such as LASSO or Ridge regression. Since regularizers allow to shrink the coefficient values to decrease model variance, by using the well-known cross-validation strategy, the penalization parameters could be optimized in order to find the best model parameters. Regarding the case of underfitting, this can also be addressed by avoiding the use of a too simple model, e.g. such as increasing the polynomial degree. Since regularizers can be used to tackle model complexity, a model can be trained that does not suffer too much from both overfitting and underfitting effects.

Although this seems as a rule of thumb that can be applied to the HM, one should be aware that a compromise between underfitting and complexity should be made. Let's imagine that in an ideal situation a relatively high polynomial degree is used to avoid underfitting, and the use of regularizers will address the overfitting effect. In this case, since the polynomial degree is high, the number of coefficients that the HM should handle will also be high, and more complex is the problem. In this case, domain drift can occur where the models trained per task are near optimal, but the HM can have serious difficulties due to high complexity problem of high number of coefficients. Hence, the solution to this problem is to lower the number of coefficients in the polynomial. Now the complexity that the HM should handle is more acceptable and we can state that an optimal parameter set is found. However, since the polynomial degree is decreased to train the source models, in certain situations, the model can be too simple to grasp the system details and generality is lost. This is the case where domain drift occurs again, but now with the task models being sub-optimal. Hence, a good trade-off between task model complexity and HM complexity should be achieved.

As an example of the previously described situation, consider that 4 different source models A, B, C and D will be used to generate a fifth one, model E, considering two variables as task descriptors. For the source models, polynomial regression will be used, and the most suitable polynomial degrees for the models are 3, 4, 5 and 6, correspondingly to A, B, C and D. Since in the HM the same machine learning technique should be used for all source models, one can consider using the polynomial degree 6 to grasp all the details of all models. From this decision, two different situations can occur: 1) If the amount of datapoints is low, overfitting can occur in models A, B and C because a higher degree is being used for modeling; 2) Since degree of source models is 6, the number of coefficients to train a HM is 7 (6 plus the bias), meaning that a function $\mathcal{H} : b \rightarrow \zeta$ should be derived, where $X \in \mathbb{R}^7$ and $\zeta \in \mathbb{R}^2$. This leads to a hard problem to solve due to high dimensionality of inputs and low amount of datapoints to train the HM. For the problem of overfitting (1), regularizers can be used to decrease the possible high values of the polynomial coefficients, however, the high dimensionality of the HM is still an issue and only a sub-optimal performance could be achieved, and domain drift occurs. Therefore, the only option is to decrease the number of coefficients used, meaning that the technique used to model the source

models should decrease the polynomial degree from 6 to 4. Inevitably, this decrease could lead to underfitting in some situations, like in models C and D. By not fully grasping the system dynamics of some models, generality is lost and only a sub-optimal performance is achieved. Now the HM can be trained more easily due to a lower dimension, deriving a function $\mathcal{H} : b \in \mathbb{R}^5 \rightarrow \zeta \mathbb{R}^2$, but domain drift already occurred. This is the trade-off between source model complexity and HM complexity, and the compromise between underfitting (in the first stage of source modeling) and complexity (in the second stage of hyper-modeling).

For HPM, since a shape analysis is made using a decomposition method, the number of eigenvectors that map the shape back to its original feature space from a set of deformable parameters should also be chosen. Hence, this trade-off should also be well balanced for HPM. At this point, both approaches are side-by-side. The main difference starts when the pool of all available tasks is decomposed assessing its correlations in the HPM algorithm, and for the HM this is made independently. For HPM the learning of these parameters is made jointly with all tasks available, while for HM each set of parameters should be learned for each task separately. Therefore, a holistic perspective of all tasks is taken for HPM and a more local one for HM. On top of all that, one should not forget that the most suitable technique for each dataset can be chosen. This is a key advantage because if the model is a good generalization of data, the predictions made by the model can be considered reliable to be used for the shape analysis of HPM. If these models are not good generalizations, valuable information is lost and the hyper-model will be trained with a limited view of the system dynamics.

In terms of time efficiency, it should be said that HPM is in fact slower than the HM. This is due to two additional steps from the HPM: 1) Decomposition of all task shapes; 2) Training time when a shape is generated by HPM. As described previously, the deformable model from SSM is based on the decomposition of all shapes and this is an additional step that should be made in order to find the deformable parameters. As for the latter case, once the HPM generates a shape and does not generate coefficients for the task model directly, an additional step of training is required to estimate a predictive model. This is the step that might take some time, because, like when training the source task models the best technique should be chosen, for this case the process is the same and involves training a set of models with different machine learning techniques.

Although being out of scope of the present work, we would like to introduce a new version of the HPM algorithm where $P(X_i) \neq P(X_j)$, which is detailed in Algorithm 2. From this assumption, we could not only learn new information about the various output feature spaces from different tasks, but also learn about the input feature spaces. The only restriction about this approach is that the input feature space among different tasks should be the same $X_i = X_j$, but different distributions can be assumed. We consider this algorithm an expansion to the one proposed in the present thesis to a more general and broad application. Hence, we call this algorithm HPM2, not only for being the second version of the algorithm but also because it contemplates both input and output feature spaces in the context of ZSL.

Starting from the algorithm arguments, the first difference between HPM and HPM2 is related with the *min* and *max* where in HPM2 these represent matrices of size m by r , where m is the

Algorithm 2 Hyper-Process Modeling - Extension with input feature space

```

1: procedure HPM2( $F, \zeta, \zeta', n, min, max$ ) ( $F$  is a set of source models,  $\zeta$  is a set of conditions
   associated with each source model,  $\zeta'$  is the target condition to be used for model generation,
    $n$  is the of data points per shape,  $min$  and  $max$  are  $m$  by  $r$  matrices (assuming  $X_i \in \mathbb{R}^r$  and  $m$ 
   source models) with all minimum and maximum values, correspondingly.)
2: Statistical Shape Model:
3:   for  $i = 1 \rightarrow m$  do
4:     Define the input to sample from existing models:  $X_i \leftarrow GenerateInput(min_i, max_i, n)$ 
5:     Get shape by merging inputs and output vectors:  $S_i = [X_i, f_i(X_i)]$ 
6:     Get the mean shape:  $\bar{S} = \frac{1}{N} \sum_{i=1}^N S_i$ 
7:     Get eigenvectors from PCA decomposition:  $\phi \leftarrow PCA(S)$ 
8:     Get deformable parameters from PDM:  $b = \phi^T (S - \bar{S})$ 
9: Hyper-Model:
10:   Train the hyper-model:  $h : b \rightarrow \zeta$ 
11:   Get the deformable parameter for new shape:  $b' = h^{-1}(\zeta')$ 
12:   Get new shape:  $S' = \bar{S} + \phi b'$ 
13:   Get input and output vectors from generated shape:  $X', Y' \leftarrow getInputOutput(S')$ 
14:   Train a model for the new task.  $f' : X' \rightarrow Y'$ 
15: return  $f'$ .

```

number of source models and r is the number of input features. These two matrices are a set of minimum and maximum values for each input per source models, so all the shapes can be generated. As already explained, the main purpose of the algorithm is to include both input and output information for the ZSL problem. Therefore, a shape now is composed by both feature spaces (line 5). Furthermore, the algorithm remains the same until line 13, where a segregation of inputs and outputs should be made to train a new model in line 14.

In most of the present thesis, one of the main arguments to propose a new method for ZSL is the clear tendency for the scientific community to develop methods for classification, as observed in literature review. Despite the existence of one general framework applicable to regression that takes advantage on model coefficients, some limitations were addressed and additionally the spectrum of ZSL techniques applicable to regression was enlarged. However, a question that comes to mind is about the applicability of HPM not only to regression, but also to classification. Since the foundation for the proposed algorithm is the data shape analysis, mainly on the output feature space, it is not applicable for such class of problems. Contrary to the presented regression problem, a classifier normally implies different kinds of inputs, which breaks one of the assumptions of HPM where $P(X_s) = P(X_t)$, meaning that the marginal distribution of data from source and target tasks should be the same. For the case of image classification, is based on the differences between these distributions that the classifier is trained. An image from a house is considerably different from a dog, and is based on the different pixel distribution in terms of color and shape that these are discriminated. Nevertheless, in Section 4 we have proposed a second version of the algorithm where $P(X_s) = P(X_t)$ is not assumed, making the HPM2 applicable for classification. Despite no tests were performed on HPM2 due to be a different problem to solve, it would be interesting to

assess the end result. The key difference in HPM2 is that instead of solely considering the output feature space for shape analysis, both input and output are used. This means that if a new task description would be provided to generate a new shape, this would be composed by the output feature space with classifications (e.g. 0 or 1) and a set of images that correspond to true and negative samples. Since this shape analysis results in a deformable model that is linear, we believe the generated images would not be of great meaning for humans. Although, if a model is trained with this generated shape, perhaps it could be a good model initialization and be better than pure chance when classifying images from the target task. If such a result is observed, it means that transfer of knowledge occurs, and perhaps non-linear techniques could be worth to explore for deformable models. An additional consideration to make is that maybe by using the raw input image a good performance is not achieved, but some other features that can be learned from images might increase the performance of the system. Several examples of those features were already presented during literature review on ZSL methods for image classification.

As the application area of the present thesis is on industrial scenarios, we would like to explore a different area and bring to discussion an additional potential for HPM. This technique is valuable also to other kinds of applications where only task descriptions or attributes that describe a certain task are available, together with information about similar tasks. This can be used in a more exploratory way, so let's think about a scenario. Imagine that a pharmaceutical company wants to explore new processes, like the combination of different chemical agents that have different interactions among them to build new drugs. In a pool of chemical agents, the amount of possible combinations is very high, and it is not feasible to try all these combinations. This kind of approach would be similar to "exploration" in search algorithms that need to search for all solutions in a certain search space so the optimal solution can be found. Rather, a more exploitation-based approach can be taken. In this case, if information about the chemical agents (e.g. some chemical properties) is available, together with the end result of their reaction and parameters used for their combination (e.g. properties about the medium used for the reaction), models can be trained to predict the outcome of a reaction based on the experimental parameters, for a specific combination of agents. Hence, by applying the HPM algorithm, models can be generated by providing the new target combination of agents. From this point, a pool of new models can be generated and tested according to the requirements for this new drug. From this pool and based on these tests, a ranking of combinations can be made and provide a more oriented search for the end result. If a small amount of top models is chosen to test in the lab, all the remaining possible combinations are avoided to be tested in the lab. This process can be repeated until the optimal, or near-optimal, combination is found. Once new experiments are available, new models can be trained and the HPM algorithm updated to generate more reliable models. The intuition behind this process, is to basically create a gradient among models so that not all the combinations need to be tested.

Figure 6.1 presents a graphical representation about the two described processes of search. On the left handed side is presented the exploration approach where all the solutions need to be assessed in order to find the optimal solution, where the blue dashed circle is the whole search space and the green dashed circle is the optimal solution. In the example of the pharmaceutical

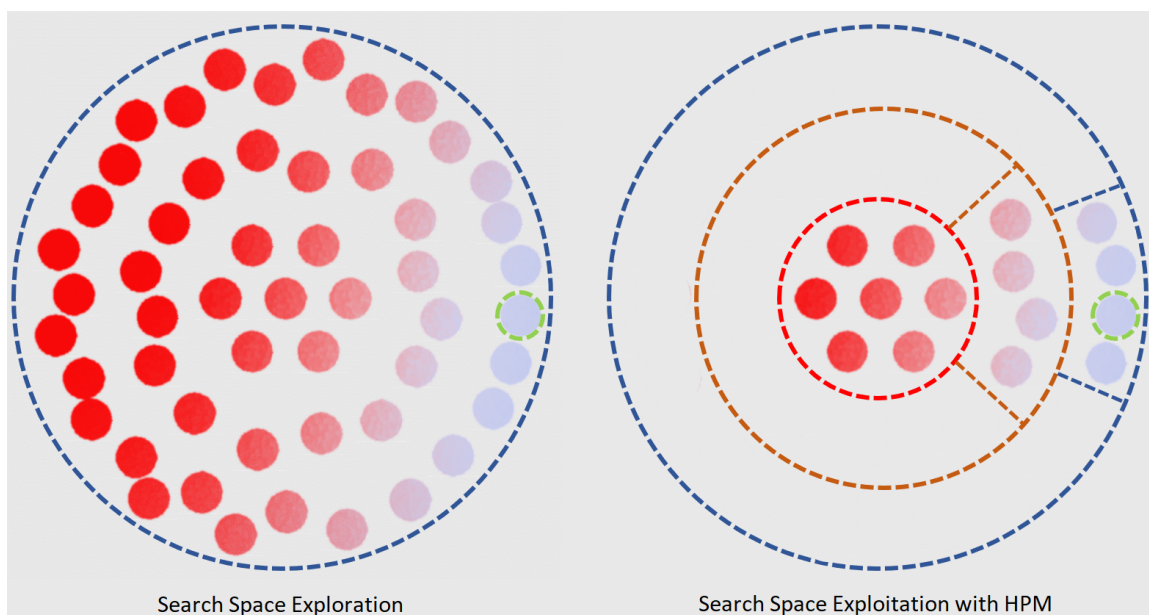


Figure 6.1: Exploration vs exploitation: On the left, a solution that needs to experiment all possible solutions in order to find the global optimal (green dashed circle). On the right, by using the HPM for model generations and an iterative process, the optimal, or near-optimal, predictive model can be found, avoiding to perform all possible experiments.

company, this means that a high amount of combinations needs to be tested in order to get the best combination of chemical agents to produce the desired drug. Contrary to this, the exploitation approach can be divided into iterations, and at each step only a few experiments are performed. The red and orange dashed circles represent two iterations of the process. In the first iteration, a pool of experiments is required to be made in order to build the hyper-model for HPM. Secondly, a set of candidate predictive models will be generated by the HPM and afterwards assessed. With this assessment, a ranking can be estimated and the best candidates of agents to combine can be selected to perform experiments. These selected experiments are the ones between the red and orange circle (first ring = first iteration). After experimentation, if the drug requirements are not yet met, a new iteration can be made, where the HPM can be retrained and generate a new set of candidate predictive models. This iterative approach can be performed the necessary number of times, and in principle, it should be more efficient than experimenting all possible combinations of chemical agents. Even if the experiments are very different from model generations from the HPM in the first iteration, since the HPM is updated every iteration, it becomes better and better at generating models and will eventually start converging.

Of course that humans do not perform these kind of experiments blindly, but rather based on knowledge about the field and previous experiments. This gradient based approach is performed by humans in almost every task performed that has a clear goal to fulfil. By measuring how far the result of performed task is from the intended goal, few changes are tried, and better results could be achieved. This is applicable as well in our example, where humans can acquire knowledge about

the chemical agents to experiment based on a set of previous experiments. However, this gets more challenging when the problem in hands is far too complex in order for humans to understand the gradient-progressing towards the final goal, and often the selection of new experiments is based on intuition. When there is a high number of variables describing the experiments and its end results, machine learning can help to make a more informative approach in such complex problems, avoiding using intuition that is hard to justify and no knowledge can be exploited from that. Hence, the main consideration to get is not to blindly rely on machine learning algorithms, mainly because it is not possible to quantify and translate all human knowledge to an algorithm, and neither to completely discard these algorithms. Opposite to this, these algorithms can be seen as complementary tools that humans can take advantage from and ease the burden of hard and heavy work that sometimes one needs to perform.

Chapter 7

Conclusions

The problem of process model generation to assist on the integration of new unseen product variants was addressed with the proposed HPM algorithm in the context of Reconfigurable Manufacturing Systems (RMS). This makes use of linear shape analysis on data itself to create a deformable model where new shapes can be generated, and the hyper-model to predict deformable parameters to generate new shapes. Here, shapes can be seen as an observable quality in a certain machine for a product variant, and by analyzing the shape variations from multiple product qualities, new ones can be predicted according to specific conditions. This constitutes a ZSL problem for regression where no information from the target product variant is available and a new predictor needs to be generated. So far, this kind of problem was explored mainly for image classification, where a new image from a different class than the ones used during the training phase needs to be correctly classified. As ZSL for classification solves a similar but different problem than regression, a new algorithm was proposed based on existing general frameworks already introduced for ZSL that are applicable to both classification and regression. These are mainly the works from Larochelle et al. (2008), Pollak and Link (2016) (Hyper-Model) and Isele et al. (2016) (TaDeLL). The main motivation to propose a new algorithm is basically to address some limitations from these general frameworks. We believe that by leveraging implicit properties from regression a better performance can be achieved when compared to general frameworks. This comparison was introduced in Section 4.3 where a scenario based on multiple types of curves generated from a Beta Distribution was defined. In this small example, the benefits of shape analysis on generated data from different machine learning techniques used to model different source tasks was explored. This shape analysis is particularly useful when a common representation for different machine learning techniques needs to be found. The innovative aspect of HPM is precisely such a shape analysis.

We have concluded from such an exploration for a new approach that the ZSL area has great opportunities for regression due to few works developed in this context. Regression problems present different kind of challenges for ZSL comparing with classification and new algorithms need to be proposed for further advancement and formal definition of the area. In the present work, a step forward in this direction is attempted by formally defining the ZSL problem for regression so new researchers can easily frame their challenges in this context, and ultimately foster

the development of better and more robust algorithms. With the proposed HPM algorithm, the intention is to introduce a first approach to solve the defined ZSL problem and therefore motivate new works to come.

Additionally, this new approach of ZSL problem for regression was explored in a more complex scenario by using simulated data from an industrial process of deep drawing. The idea behind such an experiment is to assess if the types and the number of source models used would greatly influence the performance of the system. One of the main conclusions is that the type of source models is key to take the most out of the HPM. The number of models is also important, but as soon as a certain amount of source models is available, the benefit of adding new models is less than adding the right one. As demonstrated in Section 5.2 the types of models used greatly influence the performance of the system. For this, two different strategies to incrementally add source models to HPM led to different system performances with a small amount of models. As soon as a significant amount of source models is used, the performance of the system tends to be the same because the two groups used become more similar.

Finally, with the last experimental setup from a real industrial scenario, despite the small amount of source models available, we conclude that it is possible to make use of the HPM to generate new models. Although the performance of the system is not near optimal, since only two source models were used, only a linear relation could be learned from these. We truly believe that if information from more tasks are available the performance would greatly improve. Additionally, the process parameter estimation approach was used to predict the best machine parameters in regard to quality requirements from customers. This was first performed in the available source models to make a first assessment of the system, and further on this parameter estimation was performed in a generated model from HPM. Although the performance of the generated model is not as good as a model trained with an available dataset, estimation could still be done to assist the shop-floor operator. Hence, from this experiment we can hypothesize that if the relation among tasks is linear at least two models are required, and if the complexity is higher, more models need to be used. The same principle is applied in traditional machine learning when curve fitting to a set of small data points. If the system is nonlinear and only a small number of points is available, some details will be lost and rough estimations will be made.

As final remarks about the presented work, the addressed problem of ZSL in regression is particularly useful in the context of RMS, where machine reconfiguration is often required to introduce new product variants. From the new paradigm of mass customization, the need to explore new types of products by increasing the flexibility of manufacturing systems is peremptory to increase the companies' responsiveness. With the proposed algorithm we intend to leverage already known and acquired data from previous yielded products to speed up the process of integrating new ones. This way, it is possible to explore a higher number of product variations while maintaining the best quality possible, ultimately increasing the offer to the final customer.

7.1 Contributions

The main contributions of the present thesis are described as following:

- State of the art regarding the problem of Zero-Shot Learning (ZSL), framed in the broader area of Transfer Learning. This literature review reveals a dominance of image classification application area in ZSL problems, with few works applicable in different areas, but still framed in a classification setting;
- Definition and formulation of ZSL problem on a regression setting and direct comparison with ZSL for classification. This new formulation is state to be particularly useful because it complements the problem of ZSL in order to contemplate both classification and regression;
- Proposal of an algorithm called Hyper-Process Model (HPM) that is a concrete realization of the ZSL problem in a regression setting. Despite the existence of general frameworks, non of them really takes advantage of the regression setting. This algorithm is mainly directed to problems where only the outputs of a task are able to change and the distribution of the inputs are equal $P(X_i) = P(X_j)$. Additionally, a second algorithm is proposed to address the limitation of equal input distributions among source tasks, allowing $P(X_i) \neq P(X_j)$, yet not explored in the current thesis. The main limitation of this second algorithm is that both task input and output features spaces should be the same $X_i = X_j$ and $Y_i = Y_j$;
- Frame the ZSL problem in a regression setting to industrial problems. Specifically, it was defined how ZSL could be used to ease the integration of new product variants in high-mix / low-volume paradigm, where a high number of products needs to yielded in a small-lot fashion, and constant machine reconfiguration and calibration is required;
- Exploration of HPM properties in a context of a varying number of source tasks, from a small number of 4 to a higher number of 16. With the definition of this scenario in a deep drawing process, a study about the relevance of source models to the overall ZSL system performance was presented, together with a strategy to address the domain drift problem due to HPM being a two-stage algorithm;
- Definition of the process parameter estimation problem in the context of industrial systems to estimate the best machine parameters according to customer specifications in a laser seam welding scenario. With this definition, the whole methodology to assist shop-floor operators to introduce new product variants, and already existing ones, is presented, closing the loop for the practical relevance of the proposed approach, which lies outside the theoretical domain.

7.2 Publications

- Reis, J., & Gonçalves, G. (2018, September). Laser Seam Welding optimization using Inductive Transfer Learning with Artificial Neural Networks. In 2018 IEEE 23rd International

Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 646-653). IEEE.

- Antao, L., Reis, J., & Gonçalves, G. (2018, September). Continuous Maintenance System for Optimal Scheduling Based on Real-Time Machine Monitoring. In 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 646-653). IEEE.
- Reis, J., & Gonçalves, G. (2018, June). Process Modeling and Parameter Optimization for Machine Calibration in Smart Manufacturing for Laser Seam Welding. In The International Symposium on Intelligent Manufacturing Environments (InManEnv) (Vol. 1, pp. 30-35). IARIA
- Neto, L., Madsen, A. L., Silva, R., Reis, J., McIntyre, P., & Gonçalves, G. (2018, May). A component framework as an enabler for industrial cyber physical systems. In 2018 IEEE Industrial Cyber-Physical Systems (ICPS) (pp. 339-344). IEEE.
- Reis, J., Gonçalves, G., & Link, N. (2017, October). Meta-process modeling methodology for process model generation in intelligent manufacturing. In Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE (pp. 3396-3402). IEEE.
- Reis, J., Pinto, R., & Gonçalves, G. (2017, October). Human-centered application using cyber-physical production system. In Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE (pp. 8634-8639). IEEE.
- Madsen, A. L., Jeppesen, N. S., Jensen, F., Sayed, M. S., Moser, U., Neto, L., ... & Lohse, N. (2017, July). Parameter learning algorithms for continuous model improvement using operational data. In European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (pp. 115-124). Springer, Cham.
- Pinto, R., Reis, J., Silva, R., Pesch, M., & Gonçalves, G. M. (2016, June). Smart sensing components in advanced manufacturing systems. In International Journal On Advances in Intelligent Systems (Vol. 9, pp. 181-198)

7.3 Research Projects

- **"INDTECH4.0 - Novas tecnologias para fabricação inteligente (PT2020 funded project)":** *INDTECH 4.0 project – New Technologies for Intelligent Manufacturing* has as main objective the development of innovative technologies in the nowadays context of industry 4.0 / Factories of the Future (FoF), with the collaboration of multidisciplinary companies and research centers, as well as the leadership of an vehicle assembly unit - Peugeot Citroën, S.A. (designated as PSA Mangualde) - as a reference company for technology experimentation, demonstration and validation. **Key Partners:** PSA Mangualde, Critical Manufacturing, Active Space Technologies, Motofil, UTAD, UBI

- **"PRODUTECH SIF - Soluções para a Indústria de Futuro (PT2020 funded project)":** *The PRODUTECH SIF programme – Solutions for the Industry of the Future* aims at yielding an integrated responsiveness for the specification and development of new production systems, lying at the foundation of advanced production technologies that will allow to equip manufacturing companies in the face of the fourth industrial revolution.
- **"SelSus - Health Monitoring and Life-Long Capability Management for Self-Sustaining Manufacturing Systems" (FP7 European funded project):** The vision of SelSus is to create a new paradigm for highly effective, self-healing production resources and systems to maximise their performance over longer life times through highly targeted and timely repair, renovation and up-grading. The SelSus vision will be achieved by the development of a new synergetic diagnostic and prognosis environment which is fully aware of the condition and history of all the machine components within a system or factory and is in constant knowledge enriched dialogue with their human personnel. **Key Partners:** Ford UK, Eletrolux, Fraunhofer IPA, MTC UK, University of Nottingham, Loughborough University
- **"ReBorn - Innovative Reuse of modular knowledge Based devices and technologies for Old, Renewed and New factories" (FP7 European funded project):** The vision of ReBorn was to demonstrate strategies and technologies that support a new paradigm for the re-use of production equipment in factories. This re-use will give new life to decommissioned production systems and equipment, helping them to be “reborn” in new production lines. Such new strategies will contribute to sustainable, resource-friendly and greener manufacturing and, at the same time, deliver economic and competitive advantages for the manufacturing sector. **Key Partners:** FIAT Research Center, Fagor Automation, Critical Manufacturing, Karlsruhe Institute of Technology (KIT), University of Oulu
- **"I-RAMP3 - Intelligent Reconfigurable Machines for Smart Plug & Produce Production" (FP7 European funded project):** The vision of I-RAMP3 is to enable the European manufacturing industry towards smart manufacturing systems in conventional production. The project aims at creating innovative solutions in order to improve the competitiveness for this important industry sector. This goal will be reached by a novel concept for fast, optimized ramp-up and operation of production lines. By this, significant reduction of time and efforts during the setup and re-configuration of production plants will be reached. At the same time, production costs will be reduced by increasing the efficiency of manufacturing. **Key Partners:** AWL Technik, Harms & Wende, IEF Werner, University of Applied Sciences Karlsruhe
- **"Co-factor - Cooperate, Communicate and Connect to boost smart Components for tomorrow's Industry" (H2020 European funded project):** Co-FACTOR was an action supported by the European Commission under its Research & Innovation Program Horizon2020 to bring together all those players who feel that they are part of a “smart components” community. Whatever role you have in the supply chain, whether you develop new

technologies or look for smarter solutions for your enterprise, smart components are the “magnetic” core for all. With Co-FACTOR you could meet the developers of latest technologies for a smart production, learn how you can benefit from recent “Factories of the Future” projects and their output, meet experts in the field, discuss and shape future needs and roadmaps. **Key Partners:** Steinbeis, Harms & Wende, Fraunhofer IPA, Loughborough University

7.4 Future Work

The first step in the future research related with HPM is to test such an algorithm for a large spectrum of datasets. This will allow for a more consistent evaluation of its properties and detail its potential in different application areas rather than industry. This is an important step because of the different complexities existing in all sorts of real world problems, and it would be useful to clearly specify in which conditions such an algorithm would perform good and bad, so a correct model selection can be made in relation to its application. Moreover, with the identification of its limitations, new improvements can be made and therefore evolve the proposed solution to become more robust and consistent for future use in intelligent systems.

As highlighted in the previous sections, one of the interesting changes in ZSL algorithms in the latest years that significantly improved the performance in image classification systems is related with going from a two-stage approach to the use of bi-linear and nonlinear compatibility functions, like in the ALE algorithm (Akata et al. (2013)). The main motivation to adopt the use of compatibility functions is to avoid the effect of, what we call, domain drift. This effect occurs when two independent models are trained but one requires predictions from another. For an optimal setting, this requires that both models have found optimal parameters for the problem in hands, which is something hard to achieve.

In particular, for the HPM algorithm this two-stage learning occurs. In the first stage, an unsupervised model needs to be trained to find the deformable parameters for each shape based on SSM, and in the second stage a hyper-model should be learned to map these parameters into task descriptions. It would be interesting to apply this concept of compatibility function to avoid the effect of domain drift and explore a joint learning of these two stages. Similarly to the ALE algorithm, the main idea is to maximize the compatibility between the unsupervised learning from SSM and the hyper-model, and achieve an overall good performance of the systems. This joint learning approach can be achieved by using an adaptation of compatibility functions for regression problems as presented by Akata et al. (2013).

Another aspect the worth be explored to improve the implementation of HPM is the use Kernel PCA as a nonlinear analysis of shapes. In the current proposal, linear PCA is used to perform a linear analysis on shapes from a set of source tasks. It would be interesting to use different kernels such as Polynomial or Radial Basis Function (RBF) to explore different relations among source tasks that would lead to better results. The idea behind Kernel PCA is to apply a nonlinear map in an input feature space to build a feature space F , and then use linear PCA in F . The use of

Kernel PCA was already explored in few works related with 2D and 3D image processing for reconstruction of pre-images, as presented by Rath et al. (2006). The authors perform a set of tests for image reconstruction and in their experiments Kernel PCA performed better comparing with linear PCA. A pre-image is a sample in the feature space produced by the Kernel PCA, and the main idea of the authors was to reconstruct an image from given pre-images. However, it is well reported in the literature the problem of pre-images in Kernel PCA where images need to be reconstructed based on this pre-image,

Despite the context of the present thesis is framed on a ZSL problem, where no data exists from the target task and a predictive model needs to be estimated by only using a target task description, the case where labeled target data becomes available is also interesting to be explored. Since in the HPM algorithm a new model can be estimated from generated shapes using SSM, Inductive Transfer Learning can be used to transfer knowledge from the estimated model to a new model where this new labeled data is used. Similar to the work presented by Reis and Gonçalves (2018a), if an Artificial Neural Network (ANN) is used to model the generated shape from the HPM, this network can be used to initialize a target ANN and labeled target data used to fine tune it. In this new case, the source and target tasks would be the same, where transfer would be made from the model generated for a target task t and data acquired on the same task t . Since the trained model HPM will naturally have a low performance than one trained with a complete dataset, Inductive Transfer Learning could increase the system's performance.

This case of transfer learning in ANN is a widely used approach in machine learning and is relatively easy to implement. Apart from this, we would like to explore a different approach to make use of labeled target data using the HPM. Since one of the steps from HPM is the generation of a shape representing the output feature space of a task (and both input and output feature space for HPM2), one could formulate an optimization problem that searches for the generated shape that maximizes the resemblance to the labeled target data. In this case, by using a search algorithm, one could set the initial guess as the target task description and then explore near by values that would minimize the distance between the generated shapes and target data. Hence, we can formulate the following minimization problem:

$$\begin{aligned}\hat{t} &= \arg \min_{\hat{t}} \|d(h^{-1}(\hat{t})) - y'\|_2^2 + \lambda \|h^{-1}(\hat{t})\|_2^2 \\ &= \arg \min_{\hat{t}} \|d(\hat{b}) - y'\|_2^2 + \lambda \|\hat{b}\|_2^2\end{aligned}\tag{7.1}$$

where t is the target task description, y' is the new labeled data from the target task, $h(\cdot)$ is the hyper-model mapping deformable parameters b into task descriptions (conditions) and $d(\cdot)$ is the deformable model that generates shapes using deformable parameters. In this case we assume the inverse of the hyper-model because, as already discussed, it is normal that the number of deformable parameters is higher than task descriptors. However, if this is not the case, the hyper-model could simply be the mapping from task descriptions to deformable parameters and $h(\cdot)$ can be used in Equation 7.1. Additionally, one should realize that the number of labeled target data

is far less than the generated shape. This way, the generated shape should also be filtered to only compare the samples that match with the labeled target data. This matching can be easily done using the input feature space.

A regularizer is used on the predicted deformable parameters used to generate a shape to avoid these to assume a high value. If a regularizer is not used an overfitting effect could occur, where the generated shape could be meaningless and too different from the ground truth due to high values of deformable parameters, but for the matching samples of the labeled target data the error can be very small. This is similar to the overfitting effect in normal machine learning settings for a low bias and high variance. For the matching points between the generated shapes and the target labeled data the error is very small (low bias), but for the remaining points the error is too high (high variance). This is in line with the theory behind the SSM, where the deformable parameters should be constrained between $\pm 3\sqrt{\lambda_i}$, being λ the eigenvalues from the decomposition process. However, this is only possible for the HPM because of the assumption of $P(X_s) = P(X_t)$. For HPM2 assuming $P(X_s) \neq P(X_t)$ this is not possible because different deformable parameters lead to different marginal distributions of input feature space and therefore, the matching between target labeled data and generated shape would not be possible. For this setting, Inductive Transfer Learning is recommended.

Appendices

Appendix A

Deep Drawing - Source Process Models

The main goal of the present appendix is to graphically present all the process model predictions making use of 3D scatter plot. In all the following figures, the blue dots correspond to the trained data set and the orange dots to the predicted values from the trained model. Regarding the model itself, as previously described, it was used an Artificial Neural Network to model the system dynamics with 2 inputs and 2 outputs (process parameters and final quality) and topology composed by 3 hidden layers with 10 neurons each. For the training procedure, no train-test validation was used due to the low amount of datapoints to train (54 in total) so these figures allow for a visual inspection that no overfitting or underfitting occurs.

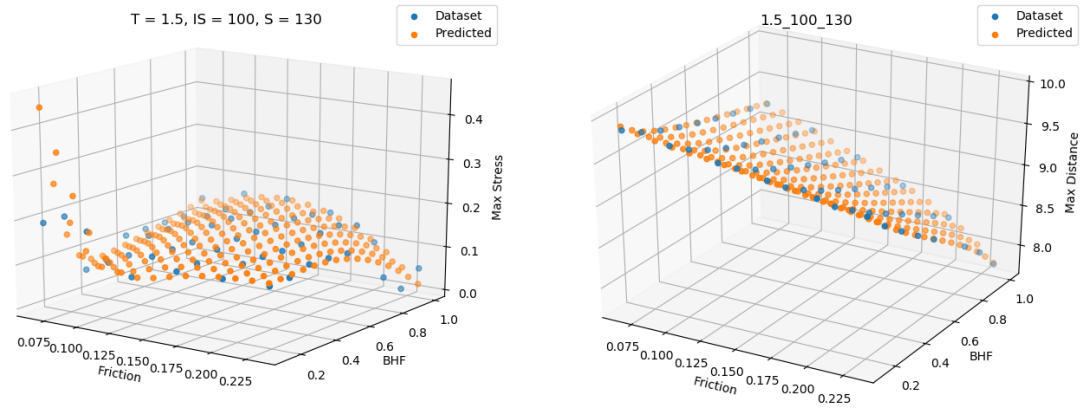


Figure A.1: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=130. For the process models, blue dots are the ground truth and in orange are the predicted values.

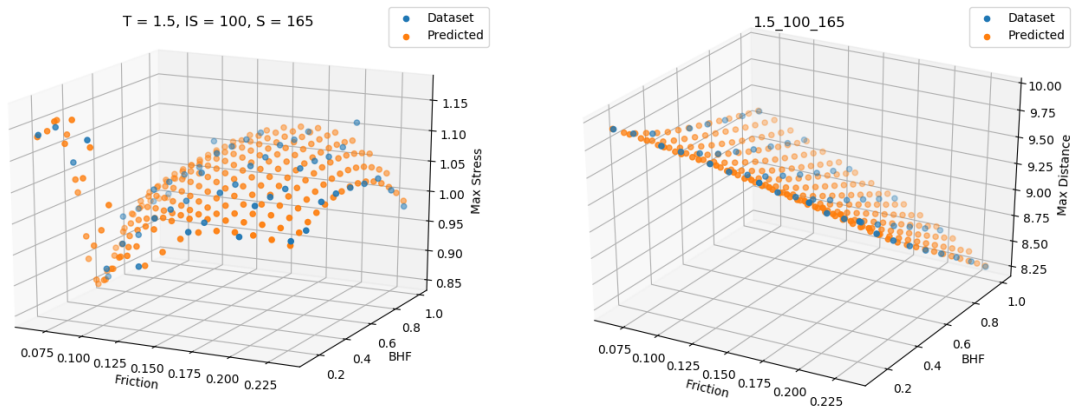


Figure A.2: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and in orange are the predicted values.

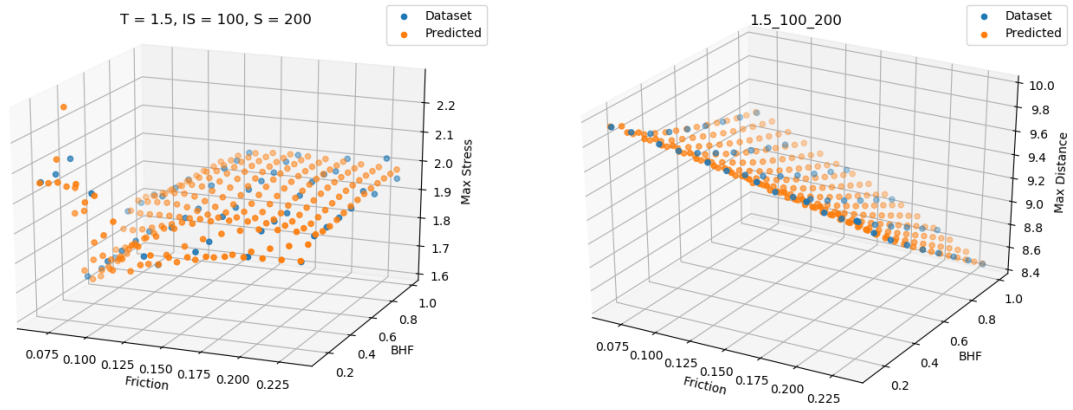


Figure A.3: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and in orange are the predicted values.

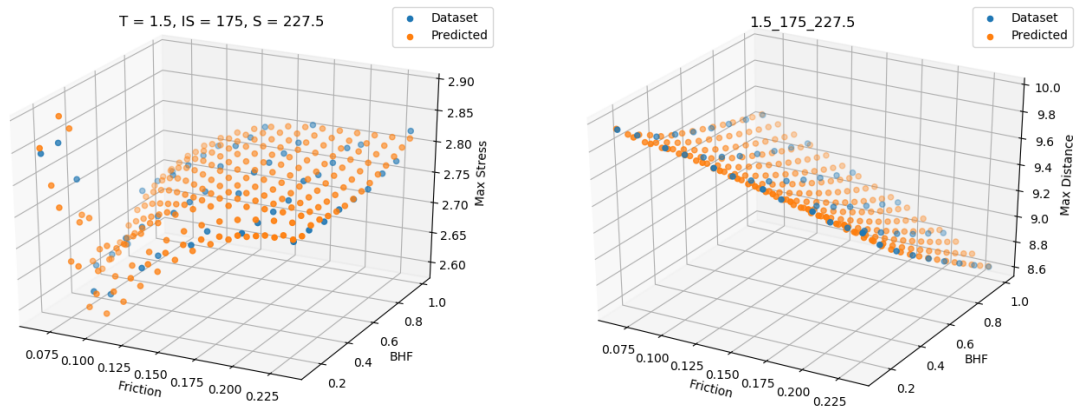


Figure A.4: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and in orange are the predicted values.

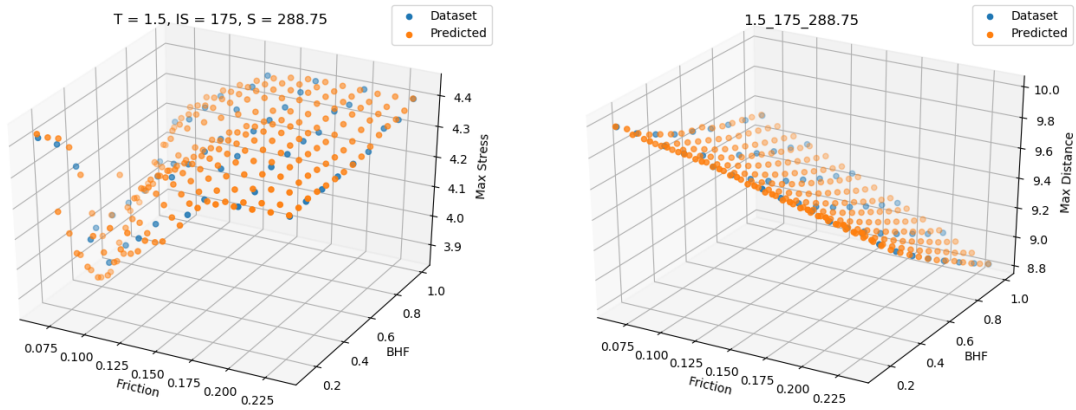


Figure A.5: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and in orange are the predicted values.

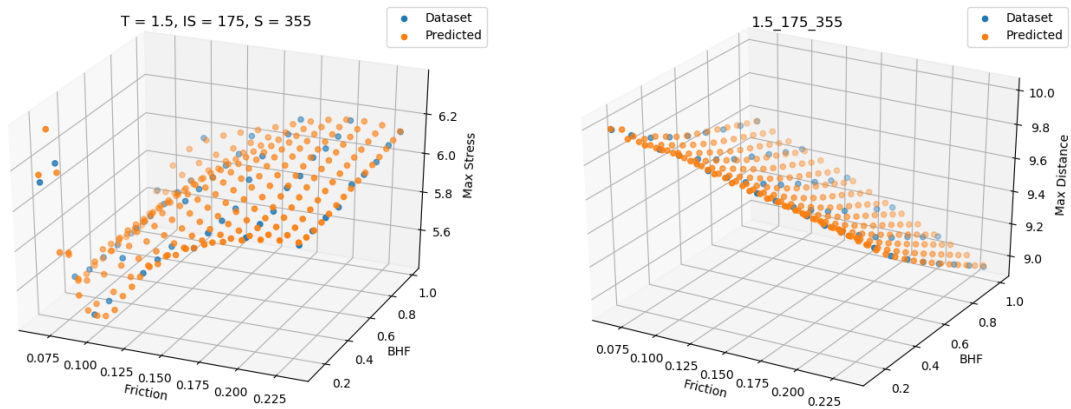


Figure A.6: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and in orange are the predicted values.

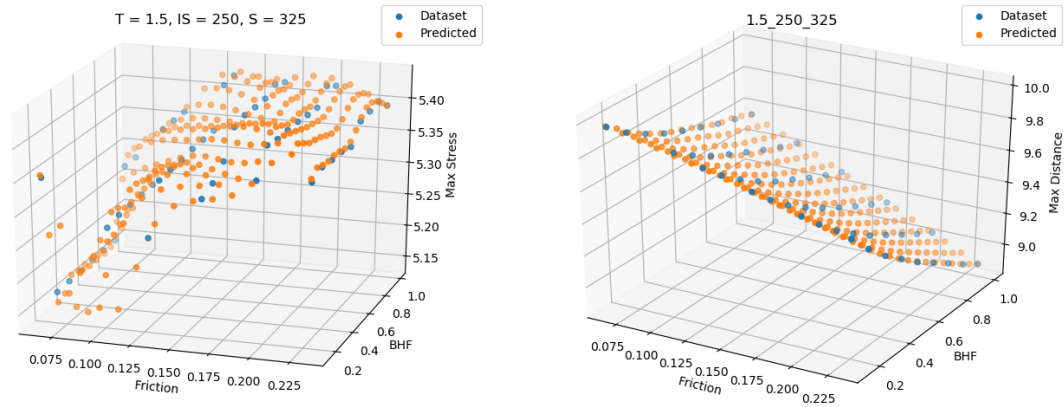


Figure A.7: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and in orange are the predicted values.

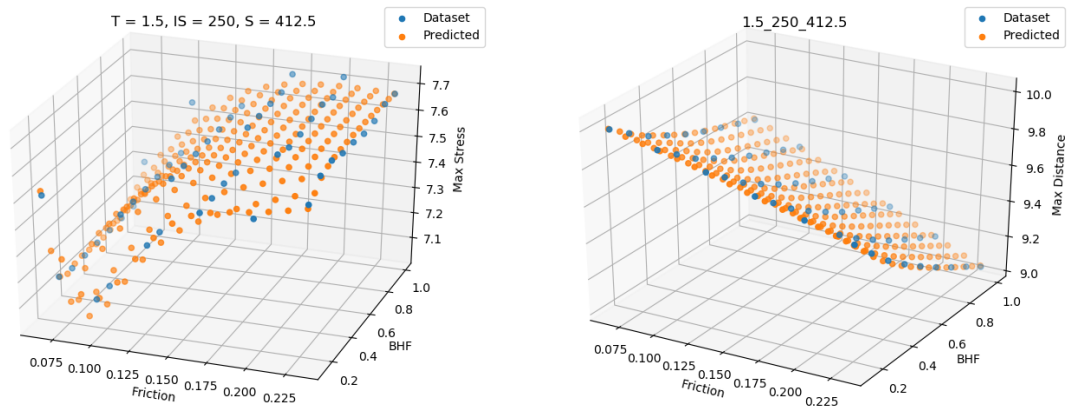


Figure A.8: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and in orange are the predicted values.

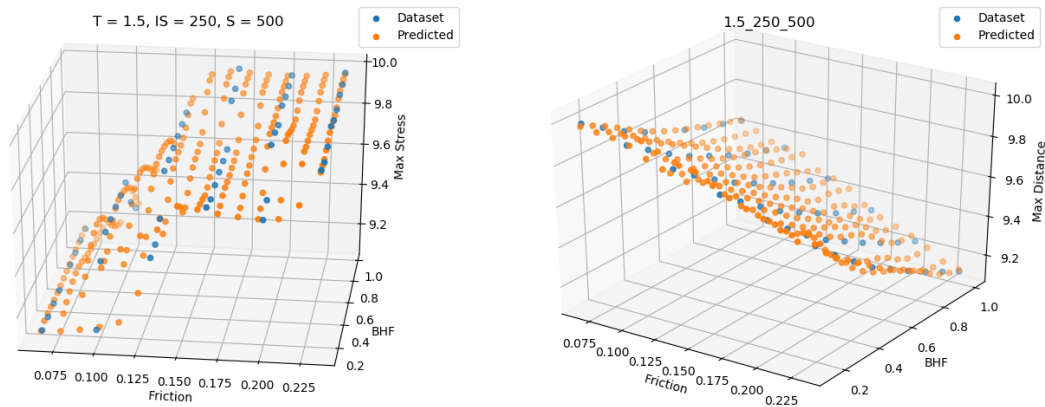


Figure A.9: Scatter 3D plot of source process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and in orange are the predicted values.

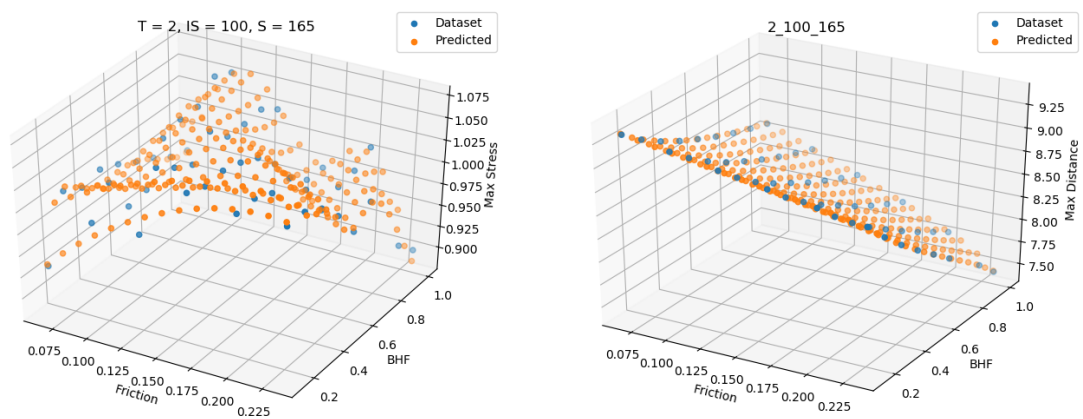


Figure A.10: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and in orange are the predicted values.

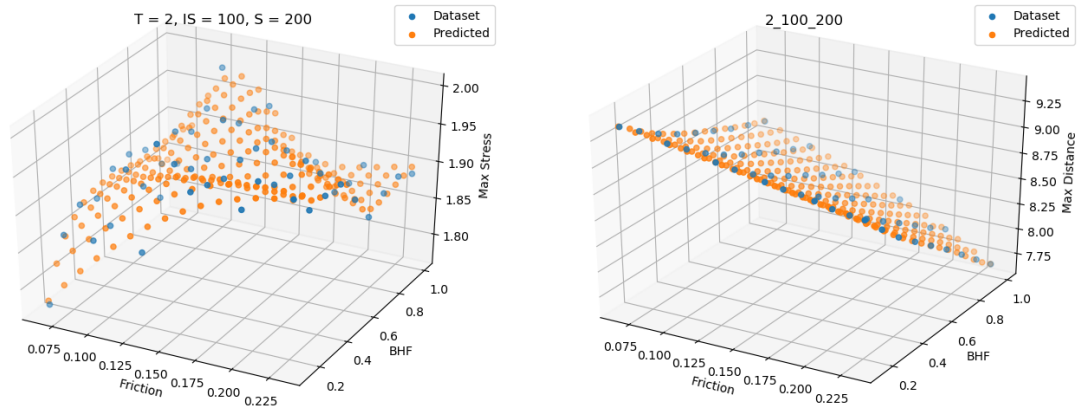


Figure A.11: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and in orange are the predicted values.

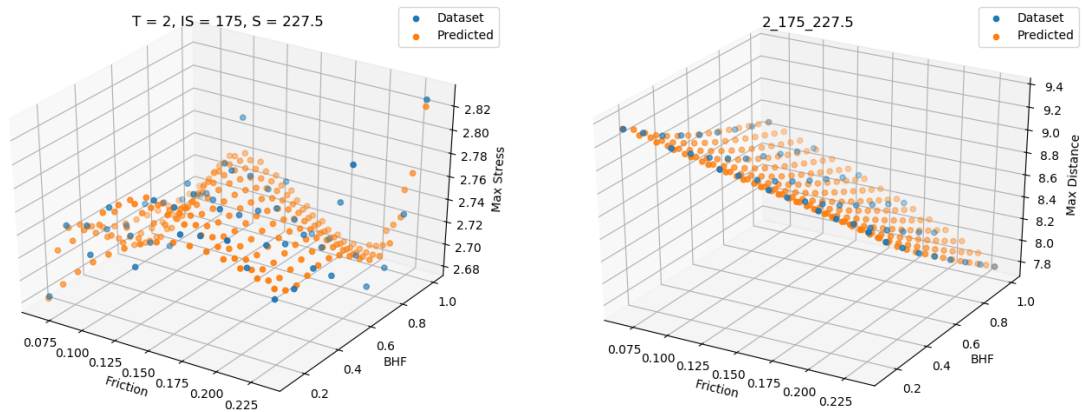


Figure A.12: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and in orange are the predicted values.

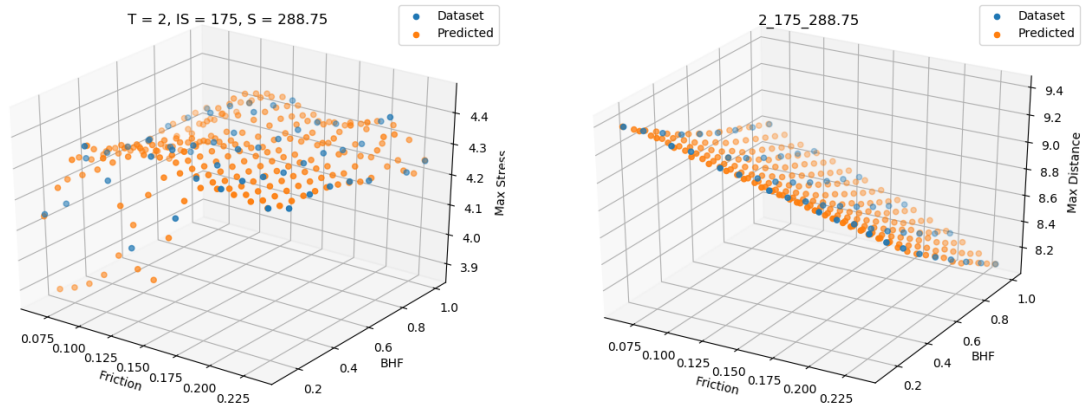


Figure A.13: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and in orange are the predicted values.

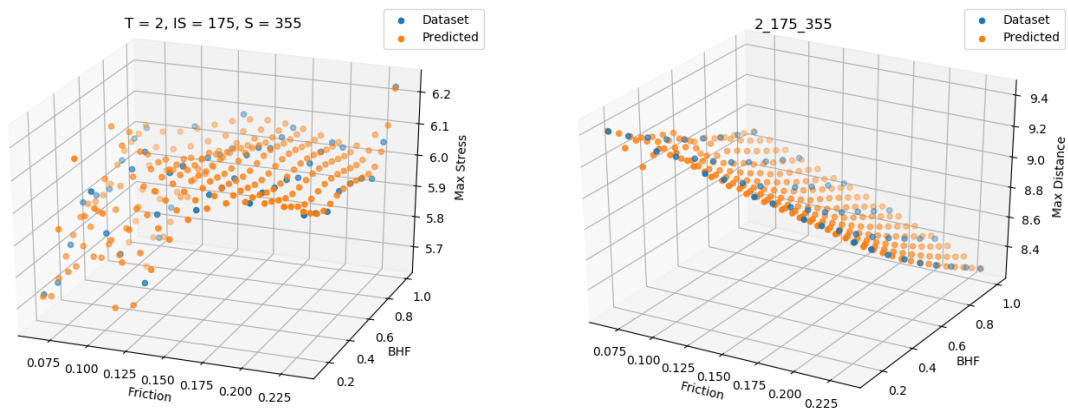


Figure A.14: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and in orange are the predicted values.

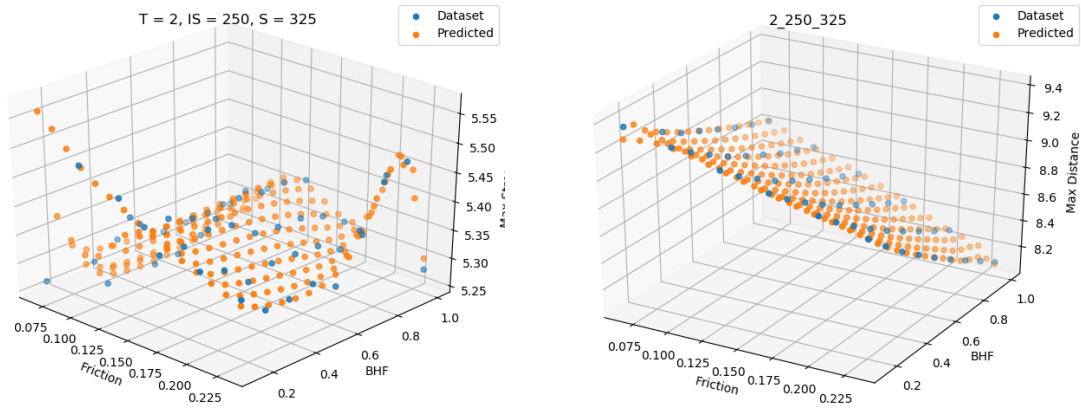


Figure A.15: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and in orange are the predicted values.

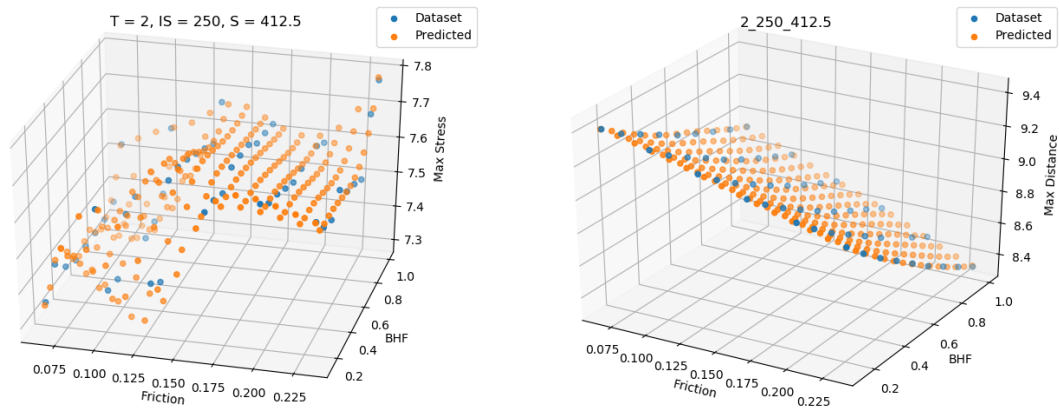


Figure A.16: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and in orange are the predicted values.

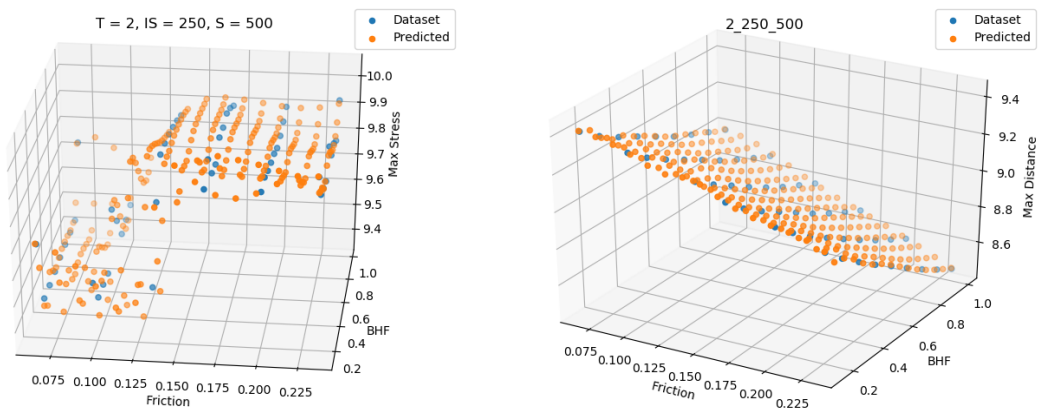


Figure A.17: Scatter 3D plot of source process model predictions for Thickness=2, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and in orange are the predicted values.

Appendix B

Deep Drawing - Zero-Shot Learning Results

The present appendix aims at depicting the best result for the HPM algorithm in predicting all the processes, based on a set of source models, in a Zero-Shot Learning fashion. This means that to predict target process models, no data was used from the target task to train the process model, and only the process conditions were used, together with the existing process models. Additionally, the deformable parameters used to generate the target shape are depicted. In the 3D scatter plot of the deformable parameters, the blue dots are the deformable parameters from the source processes, and the green dots are the predictions in a larger set of values. This larger set of values allows to visually assess if overfitting or underfitting is occurring. Regarding the 3D scatter plot of the generated shapes, the blue dots are the ground truth dataset and the orange dots are the generated shape itself.

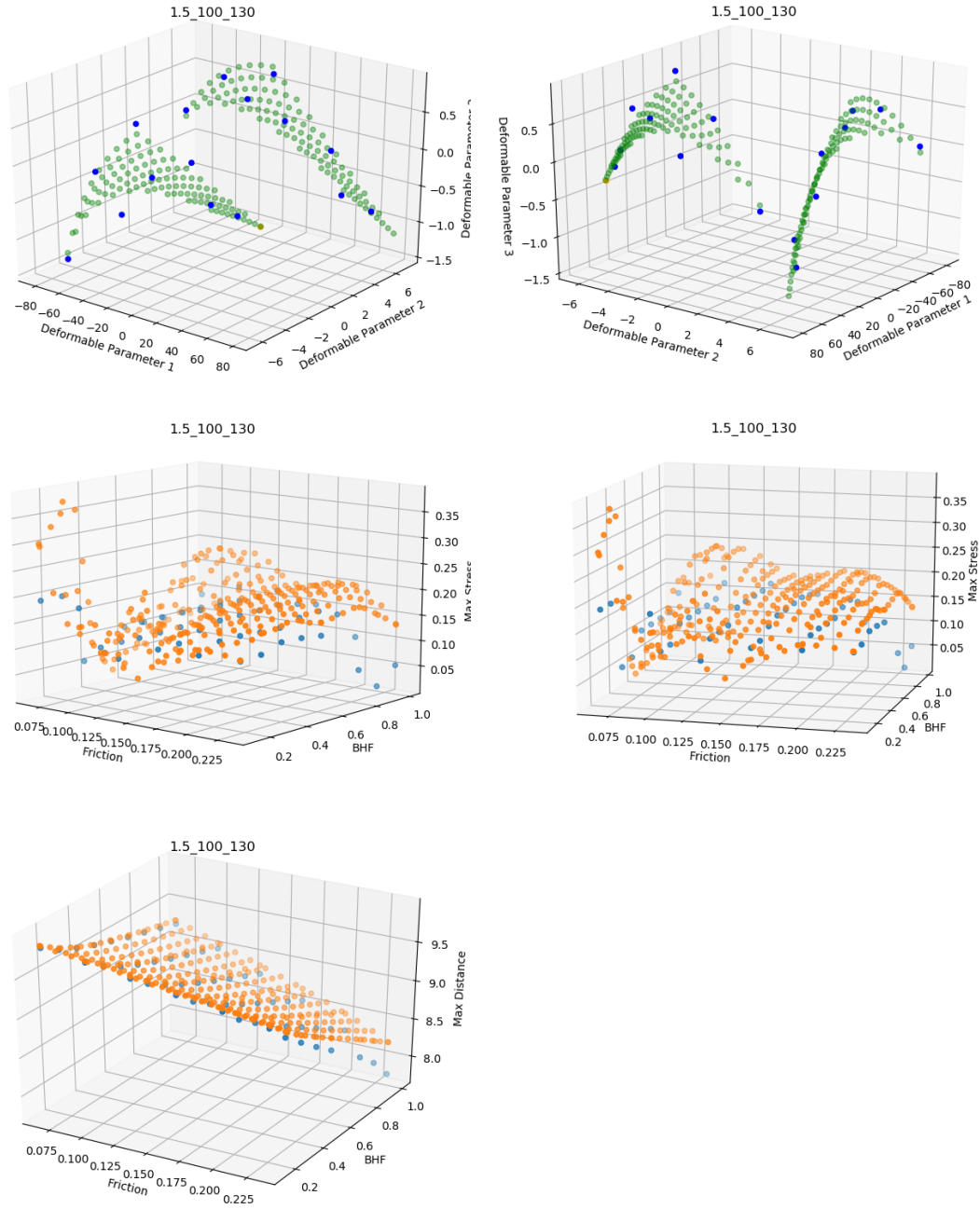


Figure B.1: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=130. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

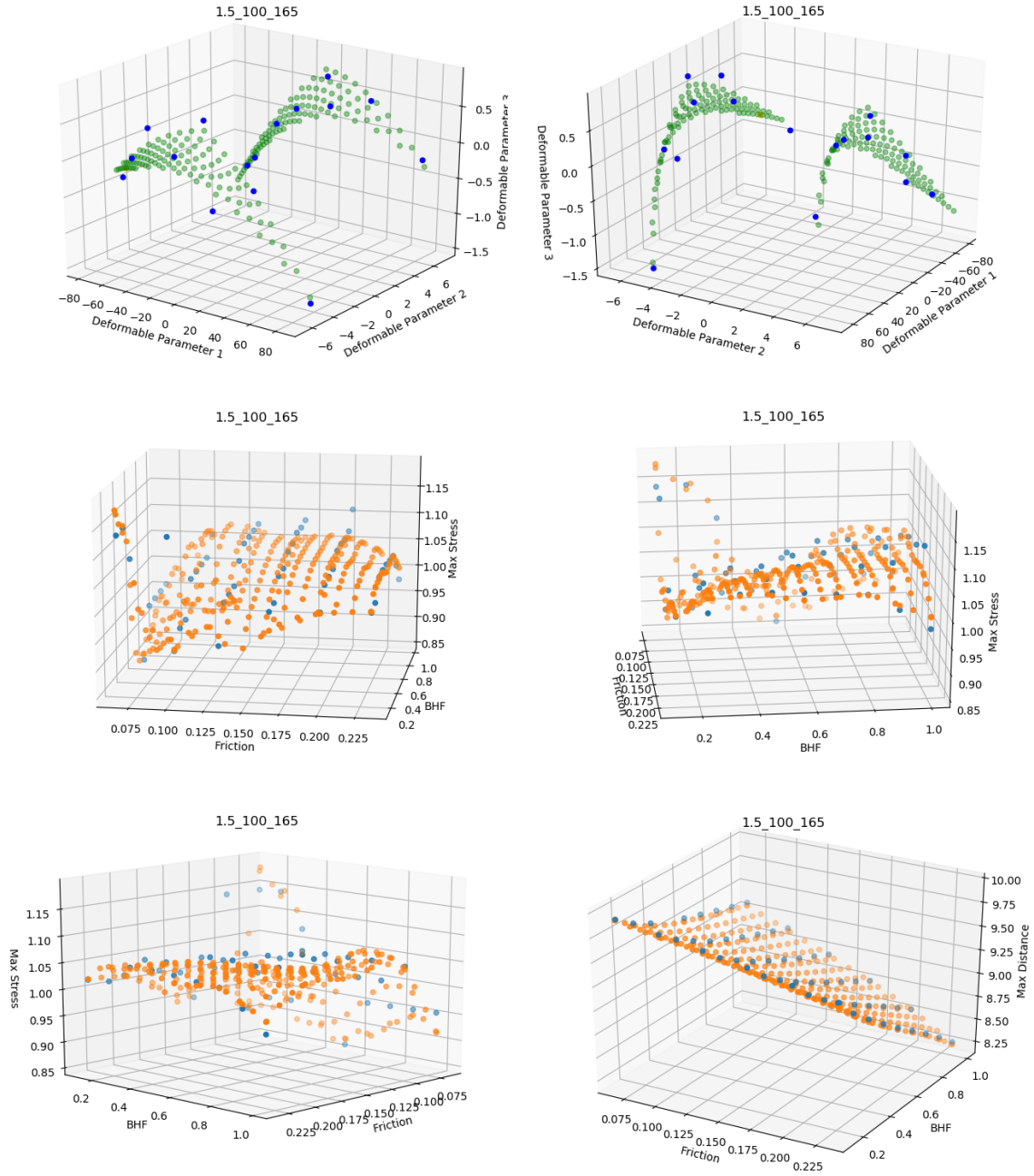


Figure B.2: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

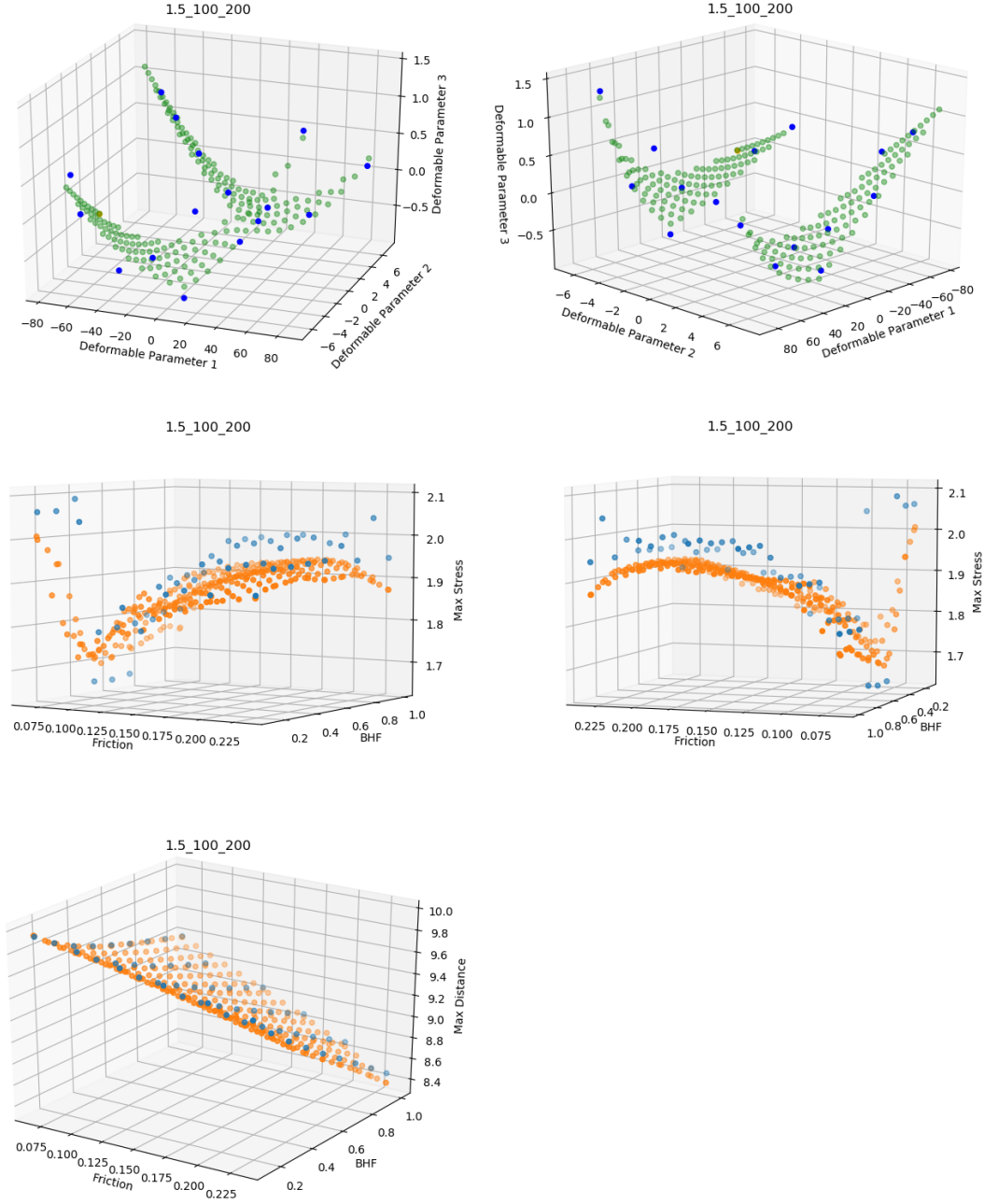


Figure B.3: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

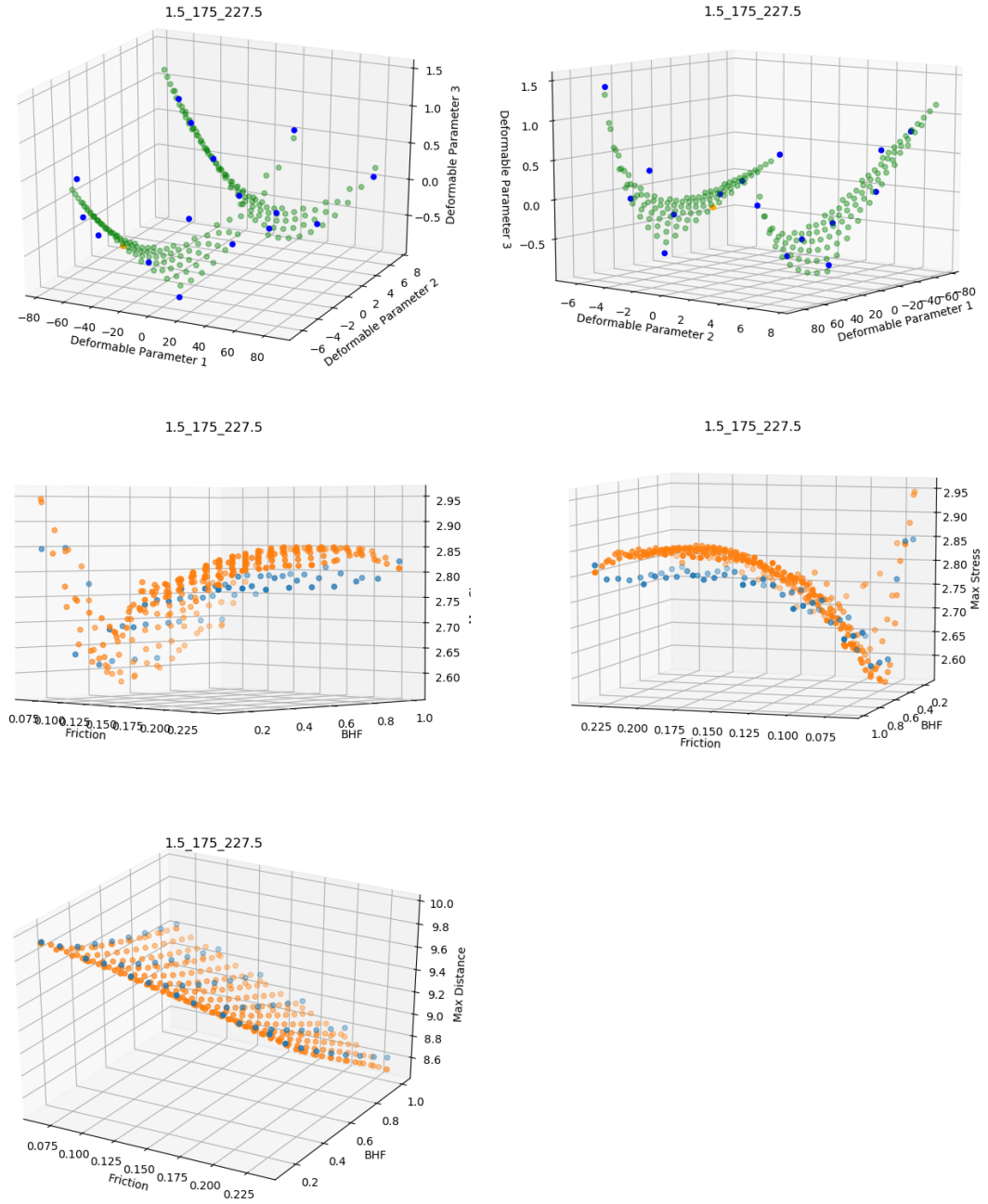


Figure B.4: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

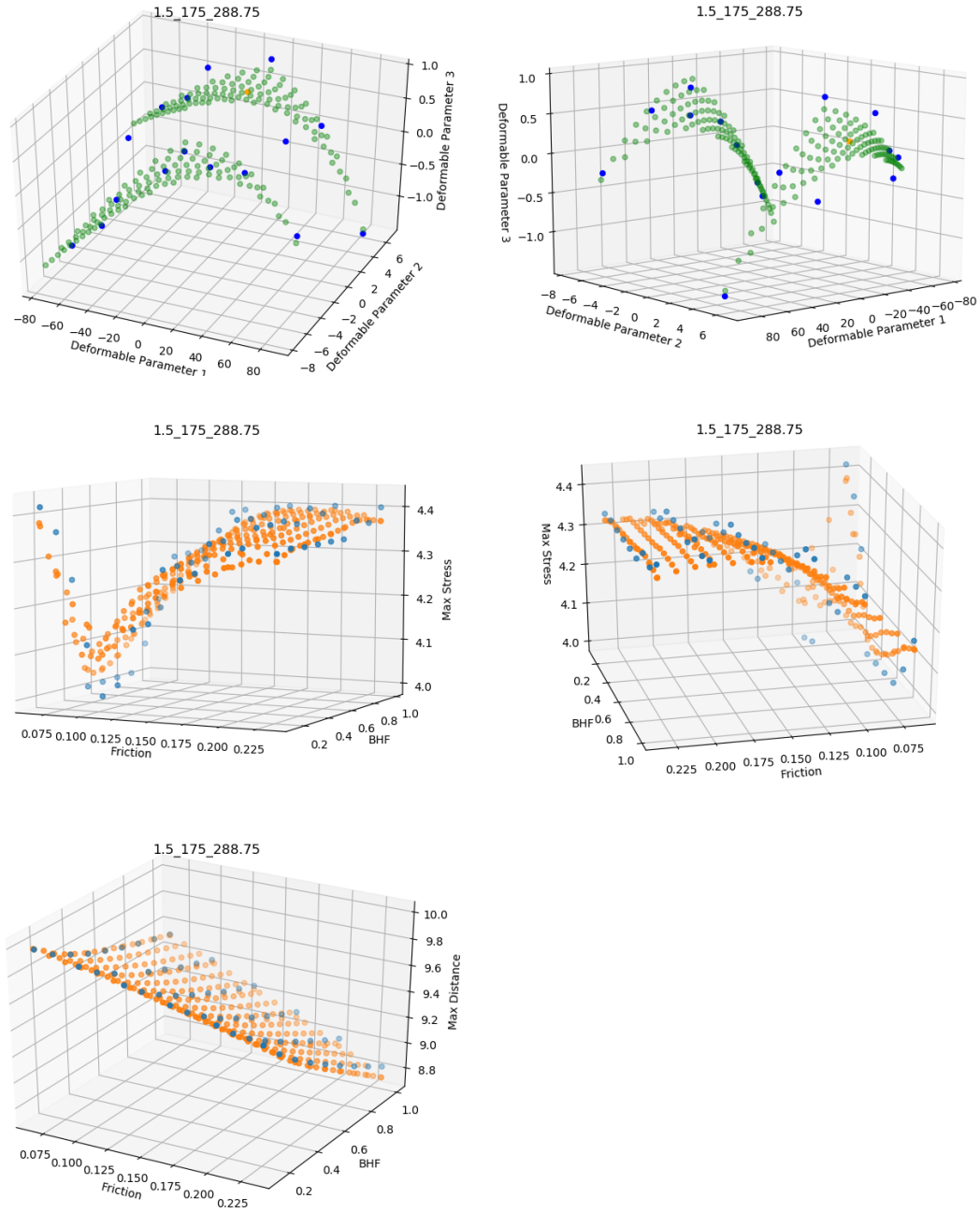


Figure B.5: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

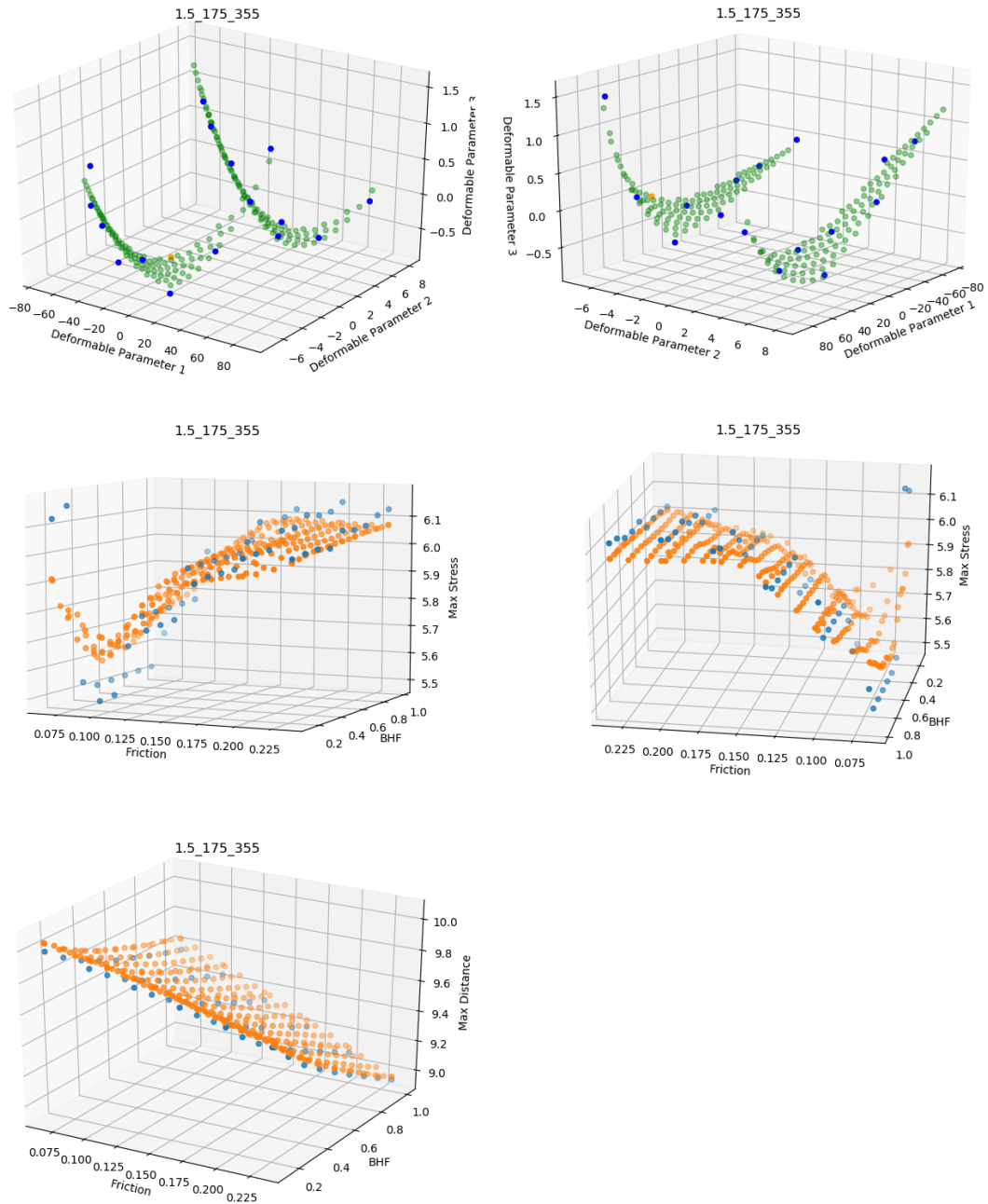


Figure B.6: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

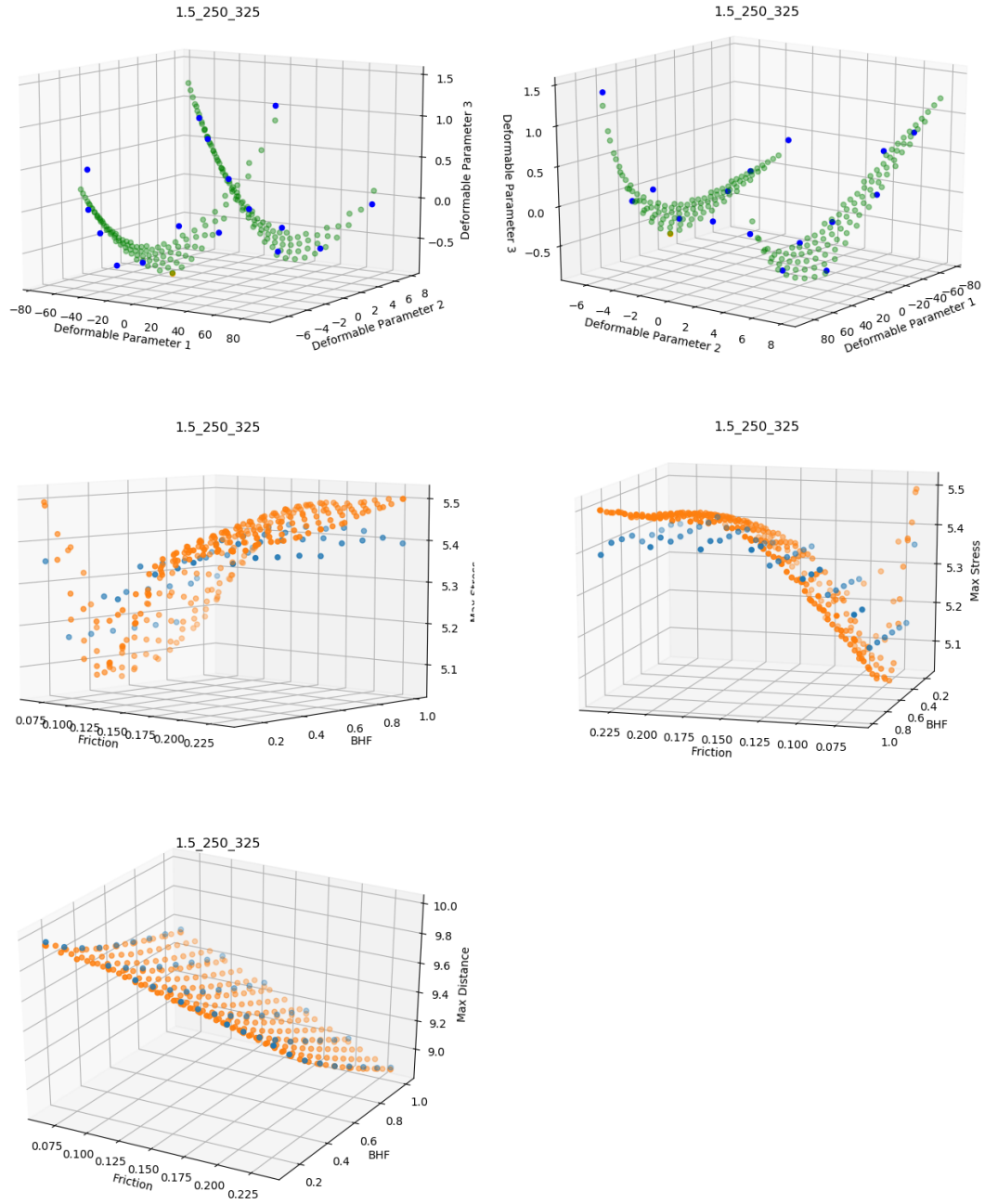


Figure B.7: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

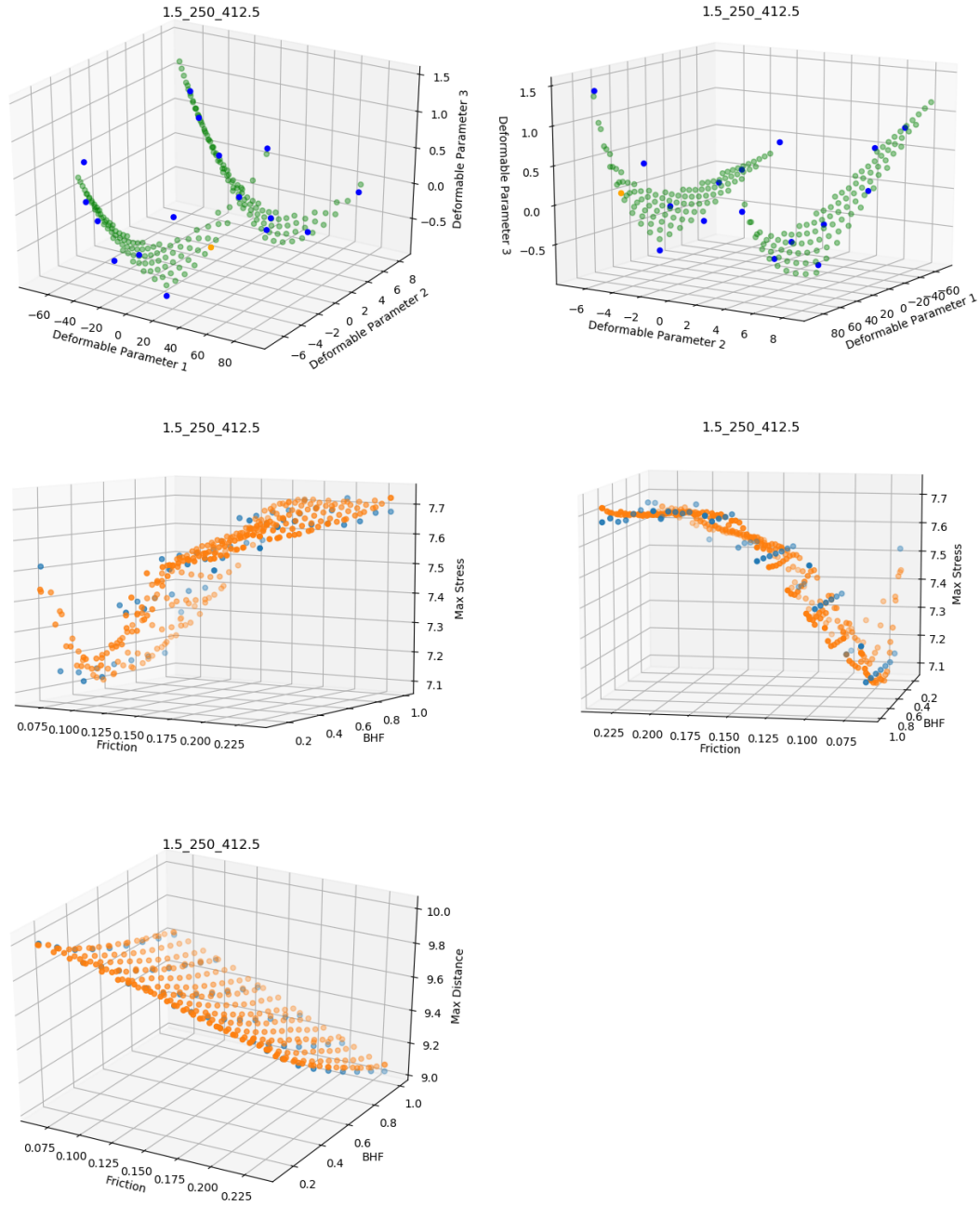


Figure B.8: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

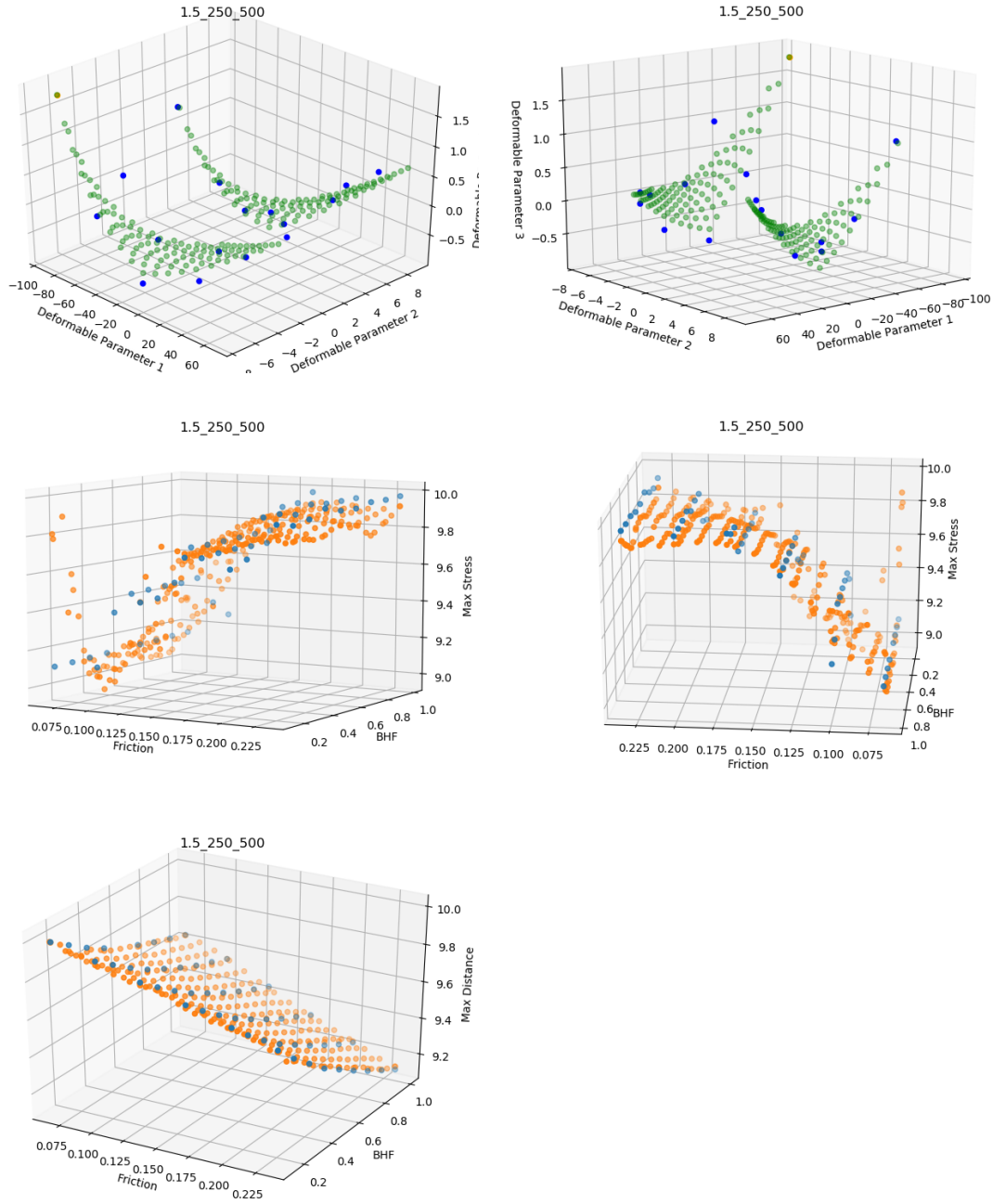


Figure B.9: Deformable parameters and process model predictions for Thickness=1.5, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

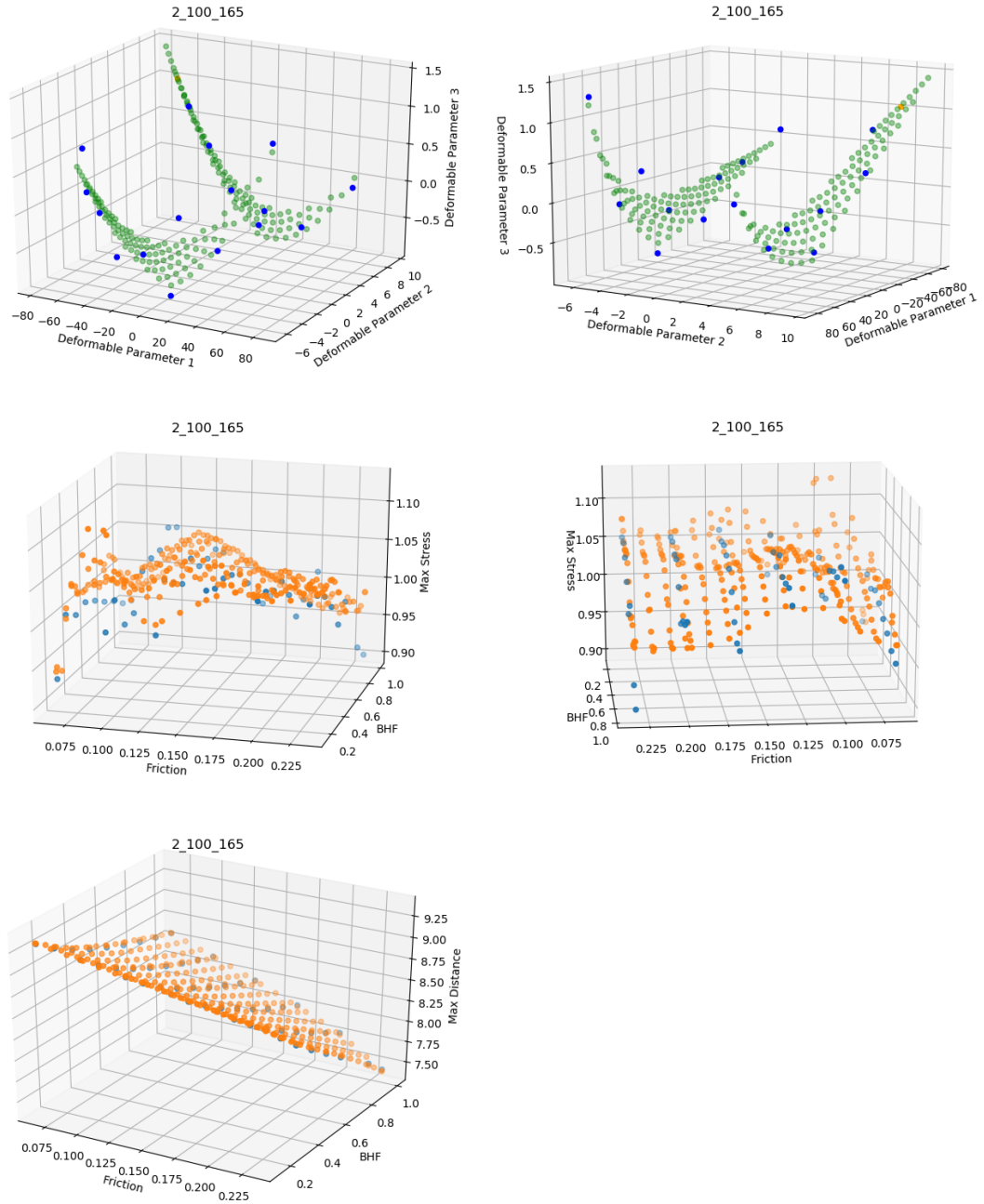


Figure B.10: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=100 and Saturation=165. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

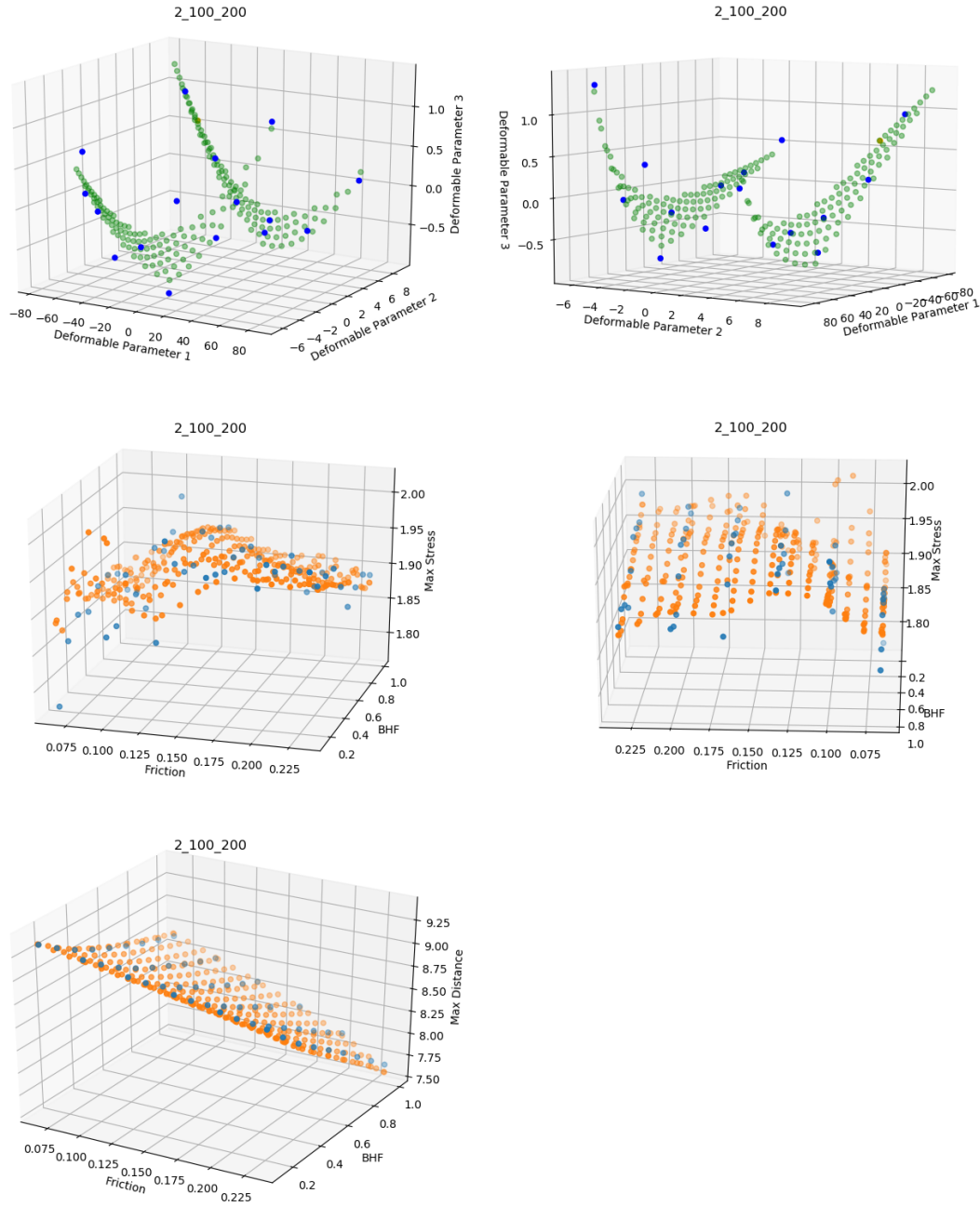


Figure B.11: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=100 and Saturation=200. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

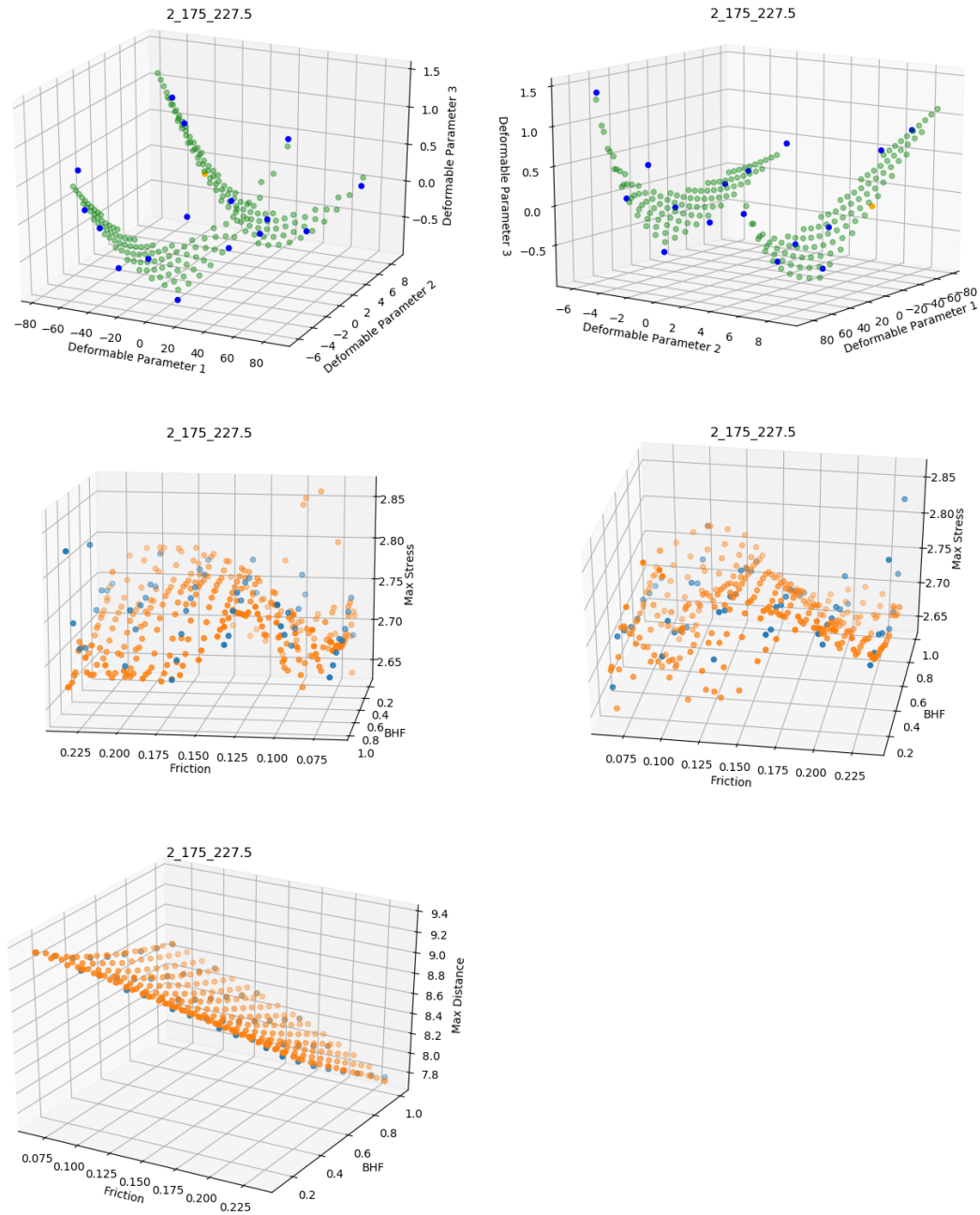


Figure B.12: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=175 and Saturation=227.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

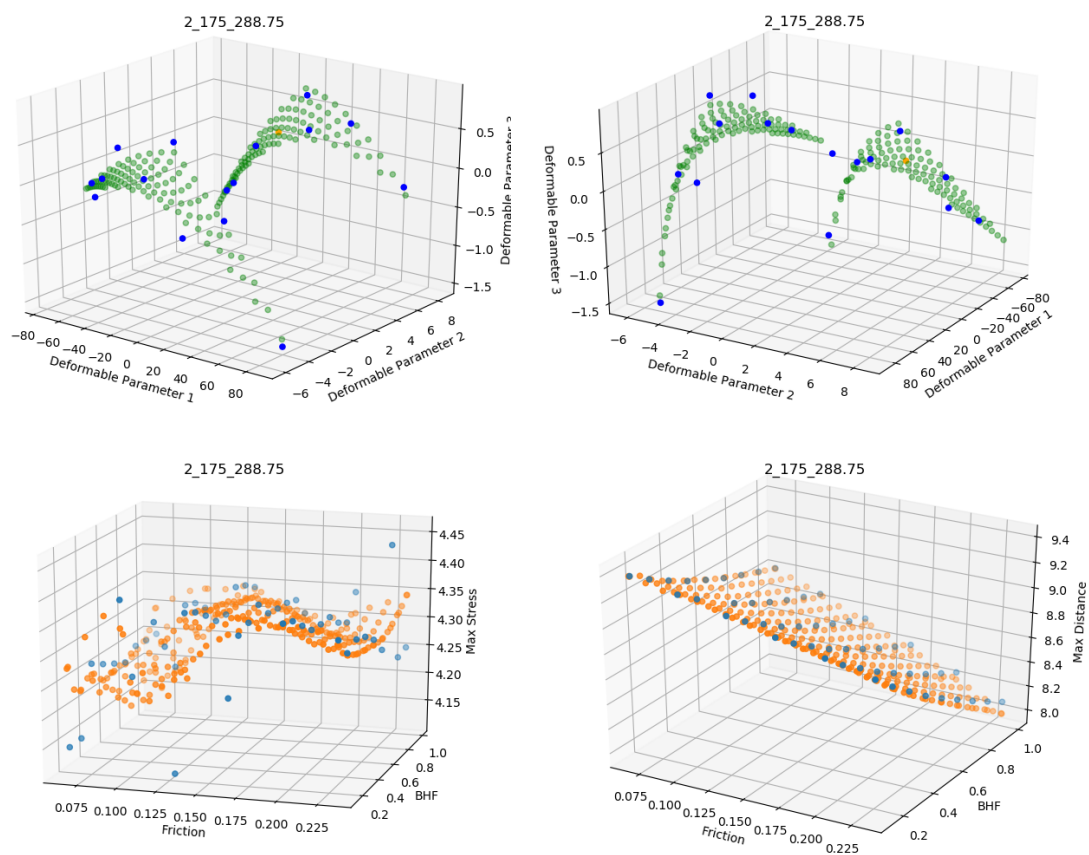


Figure B.13: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=175 and Saturation=288.75. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

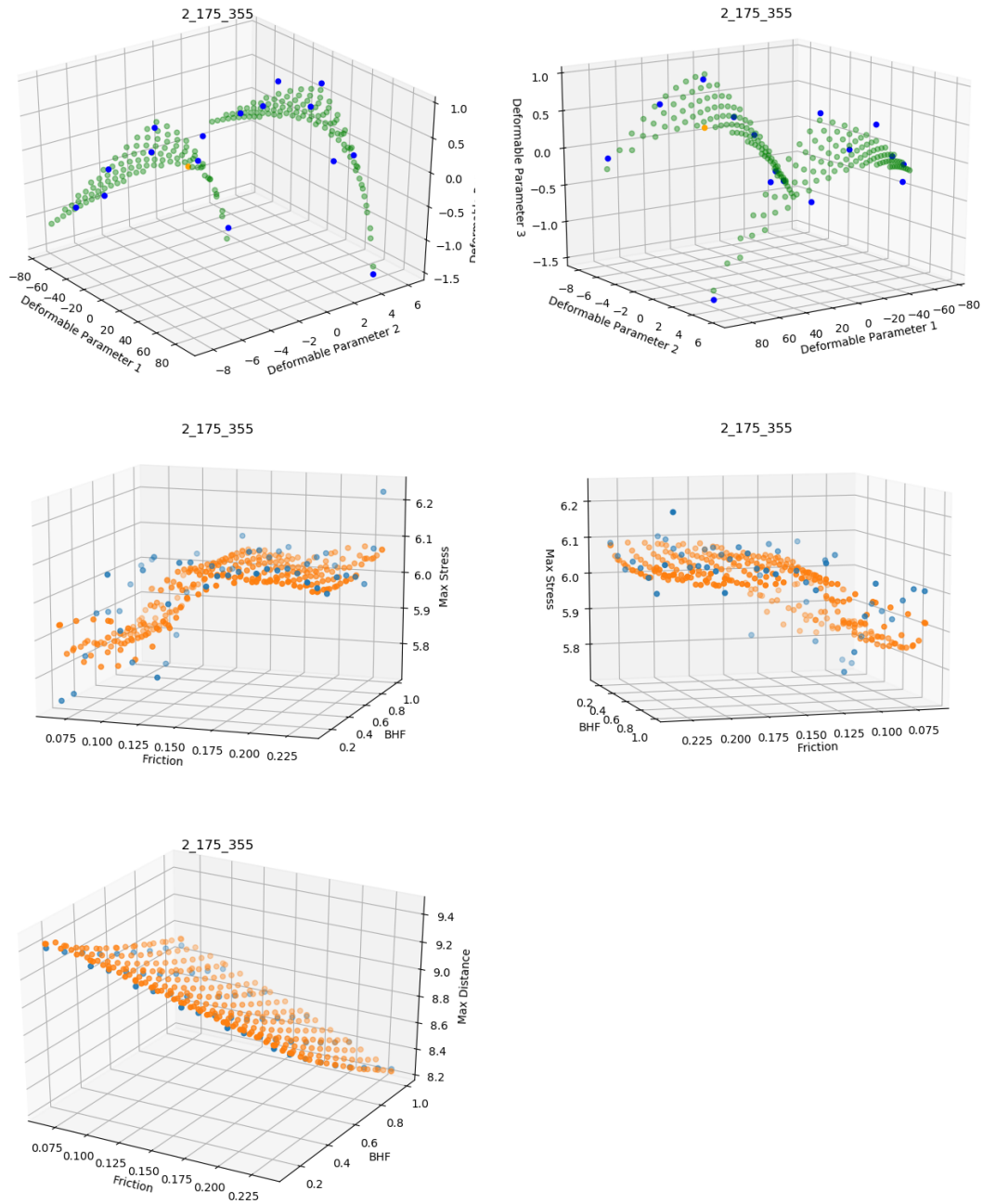


Figure B.14: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=175 and Saturation=355. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

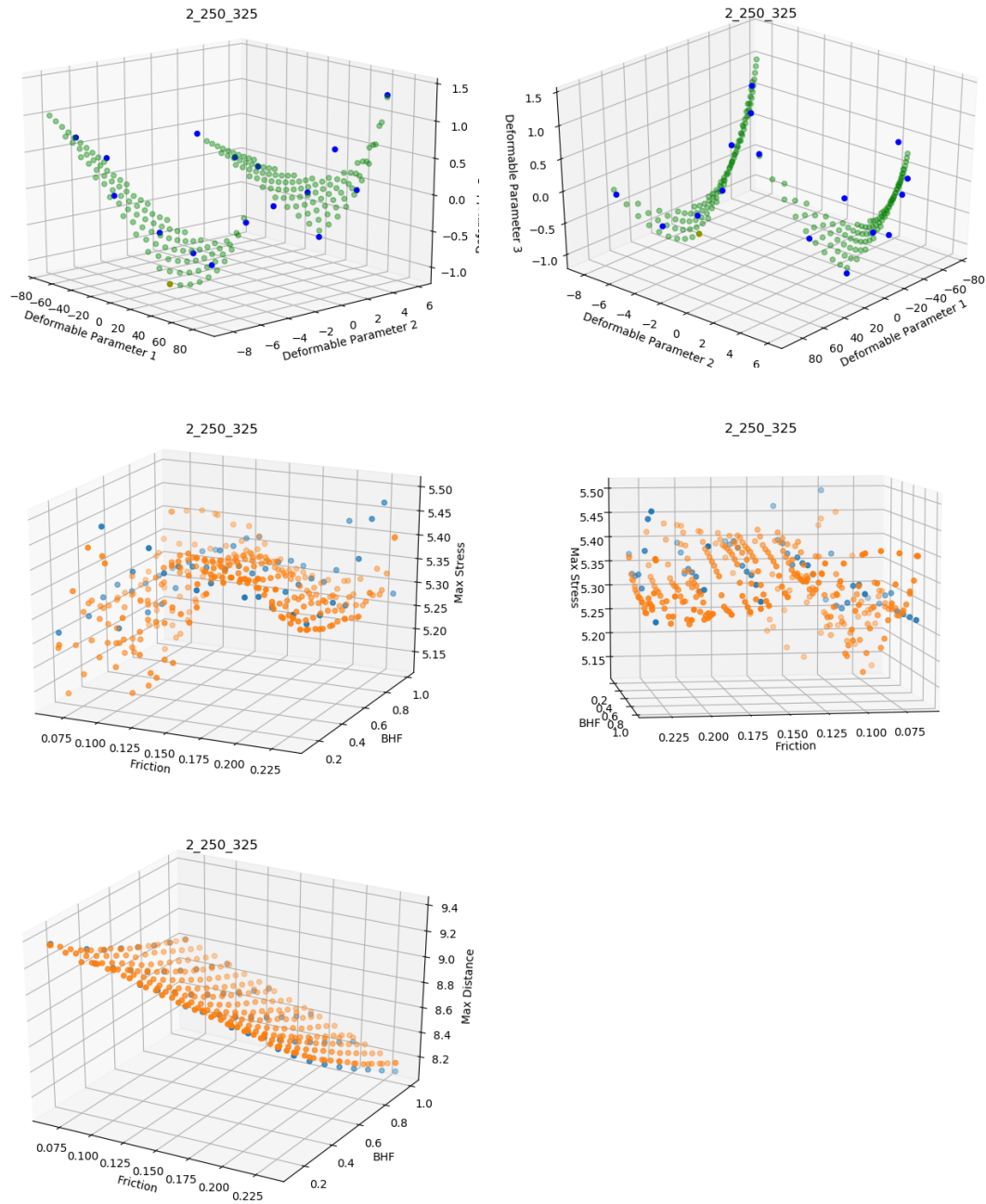


Figure B.15: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=250 and Saturation=325. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

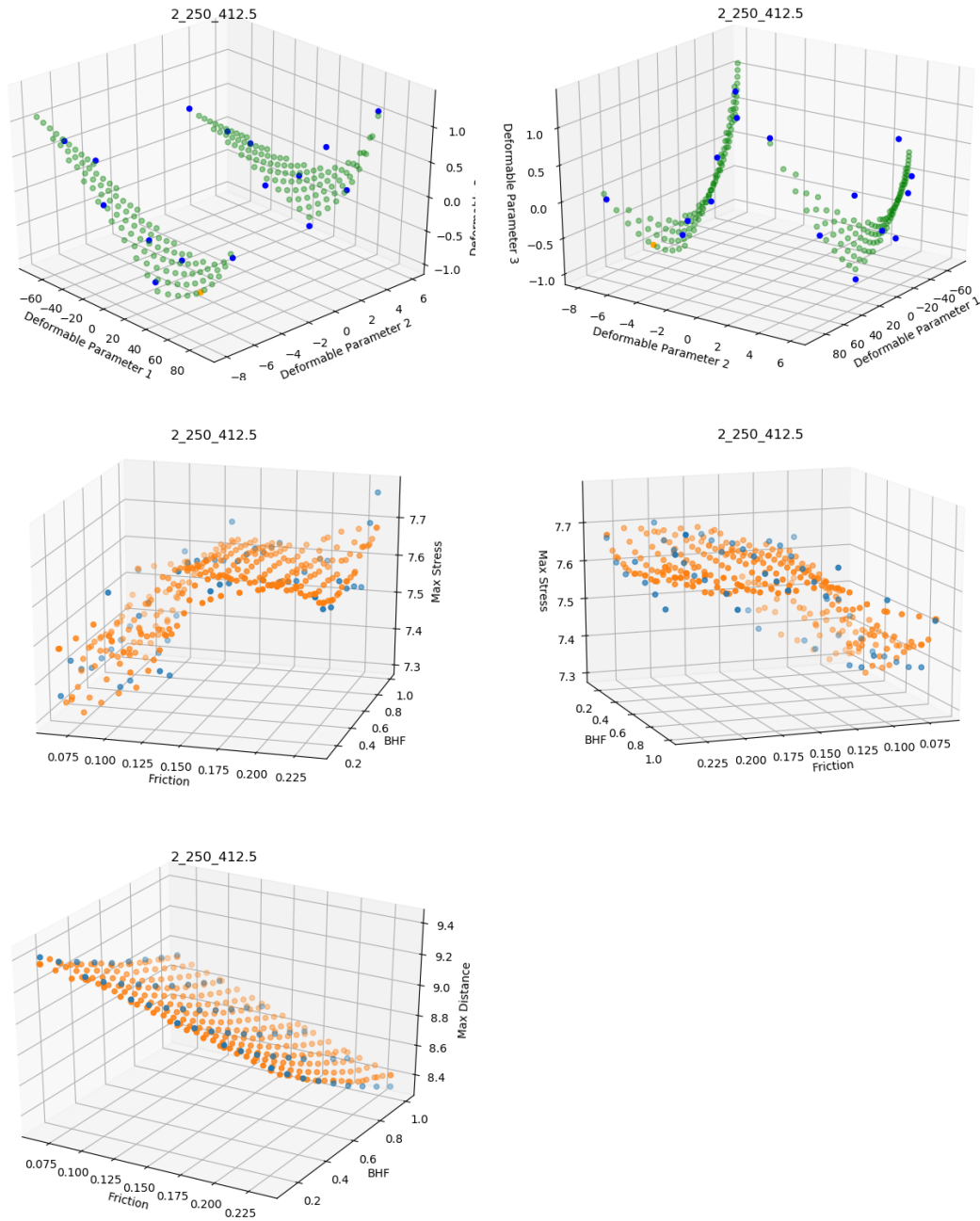


Figure B.16: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=250 and Saturation=412.5. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

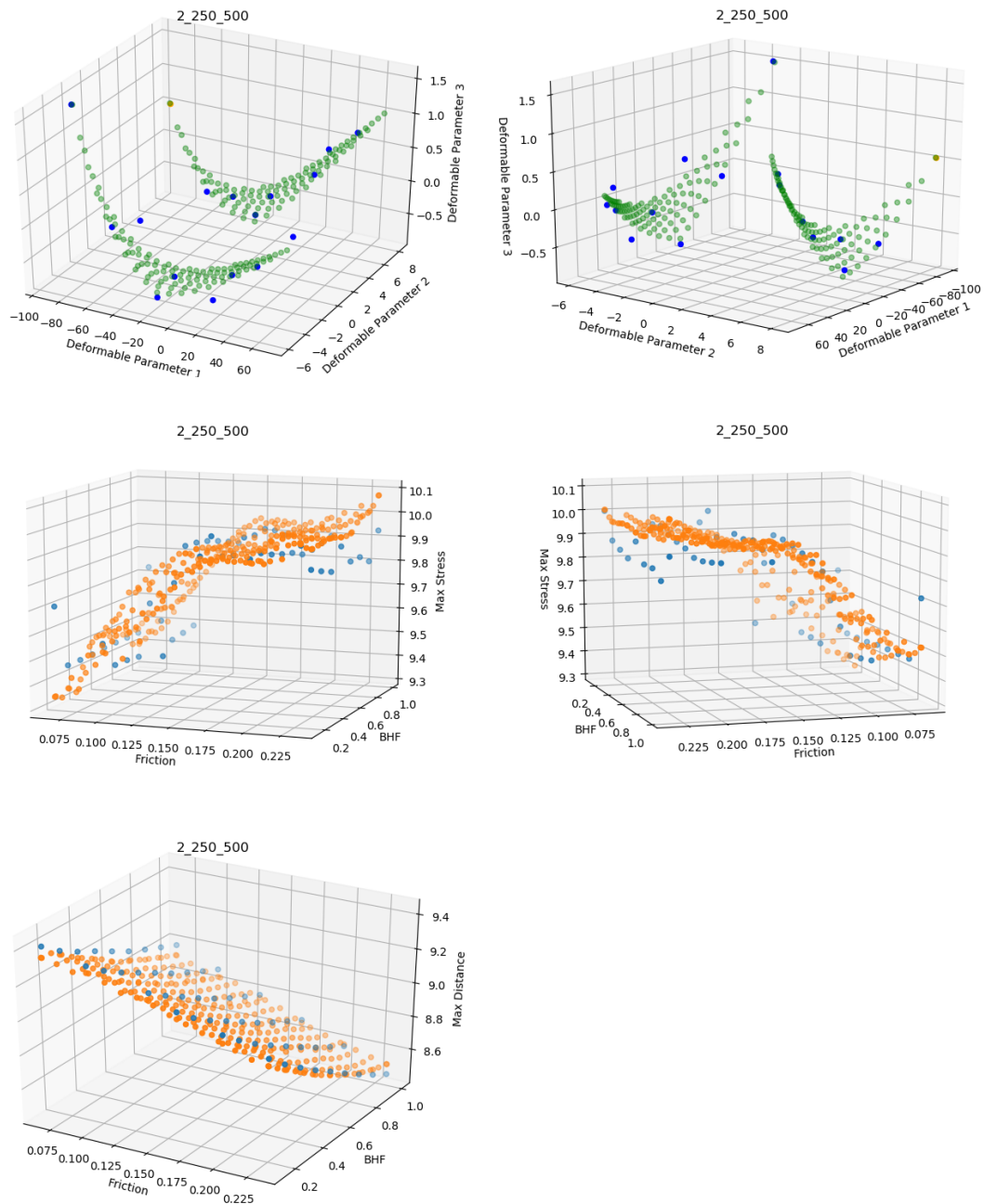


Figure B.17: Deformable parameters and process model predictions for Thickness=2.0, Initial Stress=250 and Saturation=500. For the process models, blue dots are the ground truth and orange the generated shape from HPM.

References

- Zineb Abderrahmane, Gowrishankar Ganesh, André Crosnier, and Andrea Cherubini. Haptic zero-shot learning: Recognition of objects never touched before. *Robotics and Autonomous Systems*, 105:11–25, 2018.
- Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826, 2013.
- Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015.
- Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7): 1425–1438, 2016.
- Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.
- Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.
- Marco Bortolini, Francesco Gabriele Galizia, and Cristina Mora. Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems*, 49:93–106, 2018.
- Augustin Cauchy. Méthode générale pour la résolution des systemes d’équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.
- Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016.
- Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 921–928, 2011.
- Timothy Cootes, Christopher Taylor, David Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.
- I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis*. Wiley Series in Probability & Statistics. Wiley, 1998.

- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- Rashmi Dwivedi and Geeta Agnihotri. Study of deep drawing process parameters. *Materials Today: Proceedings*, 4(2):820–826, 2017.
- Hoda A Elmaraghy. Reconfigurable process plans for responsive manufacturing systems. *Digital Enterprise Technology*, pages 35–44, 2007. doi: 10.1007/978-0-387-49864-5_4.
- Nicolaas KLAAS M Faber. A closer look at the bias–variance trade-off in multivariate calibration. *Journal of chemometrics*, 13(2):185–192, 1999.
- Bernhard Flury. *Multivariate Statistics: A Practical Approach*. Chapman & Hall, Ltd., London, UK, UK, 1988. ISBN 0-41-230020-6.
- Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 2. Springer series in statistics New York, 2009.
- Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.
- Joumana Ghosn and Yoshua Bengio. Bias learning, knowledge sharing. *IEEE Transactions on Neural Networks*, 14(4):748–765, 2003.
- Ariane Hellinger and Heinrich Seeger. Cyber-physical systems. driving force for innovation in mobility, health, energy and production. *Acatech Position Paper, National Academy of Science and Engineering*, (0):2, 2011.
- JE Hockett and OD Sherby. Large strain deformation of polycrystalline metals at low homologous temperatures. *Journal of the Mechanics and Physics of Solids*, 23(2):87–98, 1975.
- Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.
- David Isele, Mohammad Rostami, and Eric Eaton. Using task features for zero-shot knowledge transfer in lifelong learning. In *IJCAI*, pages 1620–1626, 2016.
- Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.
- Nour Karessli, Zeynep Akata, Bernt Schiele, Andreas Bulling, et al. Gaze embeddings for zero-shot image classification. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- Donald E Knuth. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422, 1992.
- Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

- Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel. Reconfigurable Manufacturing Systems, 1999. ISSN 00078506.
- Yoram Koren and Moshe Shpitalni. Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 29(4):130–141, 2010. ISSN 02786125. doi: 10.1016/j.jmsy.2011.01.001.
- T L_etc Lai, Herbert Robbins, and C Zi Wei. Strong consistency of least squares estimates in multiple regression ii. *Journal of Multivariate Analysis*, 9(3):343–361, 1979.
- Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.
- Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.
- Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008.
- Yann LeCun. Une procedure d’apprentissage ponr reseau a seuil asymetrique. *proceedings of Cognitiva 85*, pages 599–604, 1985.
- Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- Deep Drawing Library of Manufacturing. Deep drawing manufacturing process. http://thelibraryofmanufacturing.com/deep_drawing.html. Accessed: 2017-02-01.
- Pablo Mesejo, Oscar Ibáñez, Oscar Cordón, and Stefano Cagnoni. A survey on image segmentation using metaheuristic-based deformable models: state of the art and critical analysis. *Applied Soft Computing*, 44:1–29, 2016.
- Frederick Mosteller and John Wilder Tukey. Data analysis and regression: a second course in statistics. *Addison-Wesley Series in Behavioral Science: Quantitative Methods*, 1977.
- Tung Nguyen, Ceib Phillips, and Beatriz Paniagua. The use of spharm-pdm and mean latitude axis to evaluate airway changes. *The Angle Orthodontist*, 86(6):943–948, 2016.
- Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014.
- Diane Oyen, Terran Lane, et al. Leveraging domain knowledge in multitask bayesian network structure learning. In *AAAI*, 2012.
- Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

- Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- Jürgen Pollak and Norbert Link. From models to hyper-models of physical objects and industrial processes. In *Electronics and Telecommunications (ISETC), 2016 12th IEEE International Symposium on*, pages 317–320. IEEE, 2016.
- Jürgen Pollak, Alireza Sarveniazi, and Norbert Link. Retrieval of process methods from task descriptions and generalized data representations. *The International Journal of Advanced Manufacturing Technology*, 53(5-8):829–840, 2011.
- Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.
- Lingfeng Qiao, Hongya Tuo, Zheng Fang, Peng Feng, and Zhongliang Jing. Joint probability estimation of attribute chain for zero-shot learning. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 1863–1867. IEEE, 2016.
- Yogesh Rathi, Samuel Dambreville, and Allen Tannenbaum. Statistical shape analysis using kernel pca. In *Image processing: algorithms and systems, neural networks, and machine learning*, volume 6064, page 60641B. International Society for Optics and Photonics, 2006.
- João Reis and Gil Gonçalves. Laser seam welding optimization using inductive transfer learning with artificial neural networks. In *Emerging Technology and Factory Automation (ETFA)*. IEEE, 2018a.
- Joao Reis and Gil Gonçalves. Process modeling and parameter optimization for machine calibration in smart manufacturing for laser seam welding. In *The Seventh International Conference on Intelligent Systems and Applications (INTELLI)*. IARIA, 2018b.
- João Reis, Gil Gonçalves, and Norbert Link. Meta-process modeling methodology for process model generation in intelligent manufacturing. In *Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE*, pages 3396–3402. IEEE, 2017.
- Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951. doi: 10.1214/aoms/1177729586. URL <https://doi.org/10.1214/aoms/1177729586>.
- Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

- Mahsa Shakeri, Herve Lombaert, Alexandre N. Datta, Nadine Oser, Laurent Létourneau-Guillon, Laurence Vincent Lapointe, Florence Martin, Domitille Malfait, Alan Tucholka, Sarah Lippé, and Samuel Kadoury. Statistical shape analysis of subcortical structures using spectral matching. *Computerized Medical Imaging and Graphics*, 52:58 – 71, 2016.
- J. Shi, J. Wan, H. Yan, and H. Suo. A survey of cyber-physical systems. In *2011 International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6, Nov 2011. doi: 10.1109/WCSP.2011.6096958.
- Sang Min Shin, You-Min Kim, Na-Ri Kim, Yong-Seok Choi, Soo-Byung Park, and Yong-Il Kim. Statistical shape analysis-based determination of optimal midsagittal reference plane for evaluation of facial asymmetry. *American Journal of Orthodontics and Dentofacial Orthopedics*, 150(2):252 – 260, 2016.
- Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. Learning inter-task transferability in the absence of target task samples. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 725–733. International Foundation for Autonomous Agents and Multiagent Systems, 2015.
- Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.
- Martin Styner, Ipek Oguz, Shun Xu, Christian Brechbühler, Dimitrios Pantazis, James J Levitt, Martha E Shenton, and Guido Gerig. Framework for the statistical shape analysis of brain structures using spharm-pdm. *The insight journal*, (1071):242, 2006.
- Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.
- Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005.
- Vinay Kumar Verma and Piyush Rai. A simple exponential family framework for zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 792–808. Springer, 2017.
- David J Wales and Jonathan PK Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

- Shiyong Wang, Jiafu Wan, Di Li, and Chunhua Zhang. Implementing smart factory of industrie 4.0: an outlook. *International Journal of Distributed Sensor Networks*, 2016.
- Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, 2016.
- Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.
- Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.
- David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.
- Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016.
- Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018a.
- Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018b.
- Keegan M. Yates, Yuan-Chiao Lu, and Costin D. Untaroiu. Statistical shape analysis of the human spleen geometry for probabilistic occupant models. *Journal of Biomechanics*, 49(9):1540 – 1546, 2016.
- Meng Ye and Yuhong Guo. Zero-shot classification with discriminative semantic representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- Miaomiao Zhang and Polina Golland. Statistical shape analysis: From landmarks to diffeomorphisms. *Medical Image Analysis*, 33:155 – 158, 2016.
- Min Zhang, Hangfei Guo, Baofeng Huo, Xiande Zhao, and Jianbo Huang. Linking supply chain quality integration with mass customization and product modularity. *International Journal of Production Economics*, 2017.
- Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 4166–4174, 2015.
- Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via joint latent similarity embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6034–6042, 2016.