

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Hubris Benchmarking with AmbiGANs

Cátia Raquel dos Santos Teixeira



Master in Data Science and Engineering

Supervisor: Carlos Soares

Co Supervisor: Inês Gomes

Co Supervisor: Jan van Rijn

July 25, 2024

Hubris Benchmarking with AmbiGANs

Cátia Raquel dos Santos Teixeira

Master in Data Science and Engineering

Approved by:

President: Ana Aguiar

Member: Carlos Soares

Referee: Nuno Moniz

July 25, 2024

Resumo

Os algoritmos de machine learning e deep learning estão a proliferar com o uso crescente da inteligência artificial. Estes sistemas devem ser fiáveis e responsáveis. Testar a robustez de um modelo envolve criar e medir condições em que os modelos podem ser menos robustos. No entanto, testar um modelo e os seus parâmetros em todas as condições possíveis é um desafio. A falta de testes rigorosos destes modelos pode levar a falhas na implementação que podem até ser responsáveis por danos humanos. Este trabalho aborda a criação de conjuntos de dados de referência para tarefas de visão computacional que contêm amostras desafiantes ou ambíguas. Queremos melhorar a avaliação da robustez dos modelos de machine learning. Portanto, em vez de apenas avaliar quão corretas são as previsões, também medimos como o modelo se comporta quando enfrenta dados incertos.

Definimos uma métrica para medir a sobreconfiança de um modelo na sua previsão — Hubris. Formulámos uma metodologia para a Avaliação de Hubris para testar a robustez dos modelos. Propomos também a AmbiGAN — uma framework baseada em GAN para gerar dados realistas mas ambíguos. Um dos principais desafios desta abordagem é, portanto, a estimativa da ambiguidade.

Em investigações anteriores, o GASTeN propôs uma abordagem para sintetizar novos dados que se situam na fronteira de decisão de um único classificador. No entanto, imagens na região de baixa confiança de um modelo podem não ser ambíguas para um modelo diferente. Se tivermos um grande número de classificadores diversos, podemos determinar se um dado ponto de dados é ambíguo para todos eles. Assumimos que se a maioria concorda que o ponto de dados é ambíguo, então é universalmente ambíguo. Adaptámos o GASTeN para incorporar muitos modelos diversos para aproximar este conceito. A nova framework AmbiGAN incorpora um grupo de classificadores como Estimadores de Ambiguidade. Concluímos que usar classificadores diversos leva a uma melhor generalização do modelo.

Melhorámos a arquitetura original do GASTeN para obter um treino de GAN mais estável e resultados mais fiáveis. A introdução da perda Gaussiana no treino da AmbiGAN mostra um desempenho melhorado. Isto proporciona a vantagem de direcionar uma distribuição de resultados centrada na probabilidade de saída de 0.5 e permite ajustar a sua variância. Diferentes configurações de Estimadores de Ambiguidade foram testadas. Os melhores resultados foram alcançados utilizando a média das combinações de saída e a saída sem combinações com perda Gaussiana.

Usando a framework AmbiGAN, gerámos conjuntos de imagens universalmente ambíguas. Avaliámos então a nossa abordagem empiricamente testando um modelo RESNET-50 pré-treinado com este novo conjunto de dados para a avaliação de Hubris do RESNET-50 e comparamos os nossos resultados com o AmbiguousMNIST de Mukhoti *et al.*. Os nossos resultados mostram pontuações FID consistentes com a literatura. Os resultados de Hubris sugerem que o RESNET-50 apresenta sobreconfiança no nosso conjunto de dados sintéticos.

Palavras-chave: machine learning, deep learning, visão computacional, redes adversárias generativas, ambiguidade, inteligência artificial responsável

Abstract

Machine learning and deep learning algorithms are proliferating in the increasing usage of artificial intelligence. These systems should be reliable and accountable. Stress testing a model involves creating and measuring conditions where the models may be less robust. However, testing a model and its parameters under all possible conditions is challenging. Inadequate stress testing of these models can lead to deployment failures that could even be responsible for human harm. This work addresses the creation of benchmark datasets for computer vision tasks that contain challenging or ambiguous samples. We want to improve the evaluation of the robustness of machine learning models. Therefore, instead of only assessing how correct the predictions are, we also measure how the model behaves when facing uncertain data.

We defined a metric for measuring a model’s overconfidence in its prediction — Hubris. We formulated a methodology for Hubris Benchmarking for the stress testing of models. We also propose AmbiGAN — a GAN-based framework to generate realistic but ambiguous data. One of the main challenges of the approach is, therefore, the estimation of ambiguity. In previous research, GASTeN has proposed an approach for synthesizing new data that lies in the decision boundary of a single classifier. However, images in the low-confidence region of one model may be not ambiguous to a different model. If we have a large number of diverse classifiers, we can determine if a given data point is ambiguous for all of them. We assume that if the majority agrees the data point is ambiguous, then it is universally ambiguous. We adapted GASTeN to incorporate many diverse models to approximate this concept. The new AmbiGAN framework incorporates a group of classifiers as Ambiguity Estimators. We conclude that using diverse classifiers leads to an improved model generalization.

We improved the original GASTeN architecture to obtain more stable GAN training and reliable results. The introduction of Gaussian loss in the training of AmbiGAN shows an improved performance. It provides the advantage of targeting a distribution of results centered in 0.5 output probability and allows tuning its variance. Different Ambiguity Estimator configurations were tested. The best results were achieved by the mean and no output combinations with Gaussian loss.

Using the AmbiGAN framework, we generated sets of universally ambiguous images. We then evaluated our approach empirically by testing a pre-trained RESNET-50 model with this new dataset for the Hubris evaluation of RESNET-50 and compared our results against AmbiguousM-NIST from Mukhoti *et al.*. Our results show consistent FID scores with the literature. Hubris results suggest that RESNET-50 presents overconfidence in our dataset.

Keywords: machine learning, deep learning, computer vision, generative adversarial networks, ambiguity, responsible artificial intelligence

Acknowledgements

I want to thank my supervisors Carlos Soares, Inês Gomes, and Jan van Rijn for all the guidance and eye-opening discussions.

I thank my parents and my brother for accompanying and supporting me on this journey through life.

I am grateful to my boyfriend Miguel for his unwavering support and for always being by my side. His encouragement has been invaluable.

The computational resources of Google Cloud Platform were provided by the project CPCA-IAC/AF/594904/2023.

Cátia Teixeira

“I just hope they won’t die bitter and forgotten.”

Yan LeCun, 2024 ¹

¹ Yan LeCun. (2024, June 1). X (ex-Twitter). Retrieved from <https://x.com/ylecun/status/1796734866156843480>

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Goals and Research Questions	2
1.3	Publications	3
1.4	Outline	3
2	Literature Review	5
2.1	Background	5
2.1.1	Probability Models	5
2.1.2	Discriminative Methods	7
2.1.3	Generative Methods	8
2.1.4	Distance Measures and Loss Functions	10
2.1.5	Multi-objective Training	12
2.1.6	Evaluation	12
2.2	Diversity, Uncertainty and Ambiguity Estimation	14
2.3	Generative Adversarial Networks	15
2.3.1	Architectures	16
2.3.2	Adversarial Attacks	18
2.4	Stress Testing	18
2.4.1	Ambiguous Images in Classification	21
2.5	Summary	22
3	Methodology	23
3.1	Problem Statement	23
3.2	Hubris Benchmarking	24
3.2.1	Universal Ambiguity	25
3.2.2	Hubris Proposal	25
3.3	AmbiGANs	26
3.3.1	Generator Loss Function	27
3.3.2	DCGAN Architecture Changes	27
3.3.3	Ambiguity Estimators	28
3.3.4	Evaluation strategy of AmbiGAN	32
3.4	Hubris evaluation of RESNET-50	32
3.5	Experimental Setup	33
3.5.1	Datasets	33
3.5.2	Configurations and Hyperparameters	34
3.5.3	Training Process	35
3.5.4	Implementation Details	36

3.5.5	Visualizations	37
3.6	Summary	37
4	Results	38
4.1	AmbiGAN Evaluation	38
4.1.1	DCGAN Architecture Changes	38
4.1.2	Generator Loss Function	41
4.1.3	Ambiguity Estimators	44
4.2	Hubris Evaluation of RESNET-50	51
4.3	Summary	54
5	Conclusions	55
5.1	Key Findings and Contributions	55
5.2	Goals and Research Questions	57
5.3	Limitations and Threats to Validity	59
5.4	Future Work	60
	References	61
	Appendix A	72
	Appendix B	73
	Appendix C	74
C.1	Ambiguity Estimators Accuracy	74
C.2	Pre-trained RESNET-50 Accuracy	74
	Appendix D	75
D.1	7v1 Experiment	75
D.2	9v8 Experiments	78
	Appendix E	81
E.1	7v1 experiment	81
E.1.1	Mean Output Combination Estimator	81
E.1.2	Linear Output Combination Estimator	83
E.1.3	Meta-Learner Output Combination Estimator	85
E.1.4	No Output Combination Estimator	87
E.2	9v8 Experiment	89
E.2.1	Mean Output Combination Estimator	89
E.2.2	Linear Output Combination Estimator	91
E.2.3	Meta-Learner Output Combination Estimator	93
E.2.4	No Output Combination Estimator	95
	Appendix F	97
F.1	7v1 Experiment	97
F.1.1	Mean Output Combination Estimator	97
F.1.2	Linear Output Combination Estimator	100
F.1.3	Meta-Learner Output Combination Estimator	102
F.1.4	No Output Combination Estimator	104
F.2	9v8 Experiment	106

F.2.1	Mean Output Combination Estimator	106
F.2.2	Linear Output Combination Estimator	109
F.2.3	Meta-Learner Output Combination Estimator	111
F.2.4	No Output Combination Estimator	113
Appendix G		115
G.1	7v1 Experiment	115
G.1.1	Mean Output Combination Estimator	115
G.1.2	Linear Output Combination Estimator	116
G.1.3	Meta-Learner Output Combination Estimator	117
G.1.4	No Output Combination Estimator	118
G.2	9v8 Experiment	119
G.2.1	Mean Output Combination Estimator	119
G.2.2	Linear Output Combination Estimator	120
G.2.3	Meta-Learner Output Combination Estimator	121
G.2.4	No Output Combination Estimator	122
Appendix H		123
H.1	7v1 Experiment	123
H.2	9v8 Experiment	124

List of Figures

1.1	Examples of ambiguous images	1
2.1	Convolution process [28]	8
2.2	GAN strcutre	16
2.3	GASTeN Architecture [5]	19
3.1	Hubris Benchmarking Methodology	24
3.2	AmbiGANs Architecture	26
3.3	AmbiGANs with Gaussian Loss using Gradient Clipping.	27
3.4	AmbiGANs with Mean Output Estimator	30
3.5	AmbiGANs with Linear Output Estimator	30
3.6	AmbiGANs with Meta-Learner Output Estimator	31
3.7	AmbiGANs with No Output Combination Estimator	32
3.8	RESNET-50 Hubris Evaluation Methodology	33
4.1	Examples of images of 7v1 experiment for GASTeN and AmbiGAN architectures	39
4.2	Examples of images of 9v8 experiment for GASTeN and AmbiGAN architectures	40
4.3	Evolution of ACD during training for GASTeN and AmbiGAN architectures . . .	40
4.4	Histogram of results for 7v1 for CD and Gaussian losses with lower FID score . .	42
4.5	Boxplot of results for 7v1 for CD and Gaussian losses with lower FID score . . .	42
4.6	Examples of images of 7v1 experiment for CD and Gaussian loss	42
4.7	Histogram of results for 9v8 for CD and Gaussian losses with lower FID score . .	43
4.8	Boxplot of results for 9v8 for CD and Gaussian losses with lower FID score . . .	44
4.9	Examples of images of 9v8 experiment for CD and Gaussian loss	44
4.10	Histogram of results for 7v1 with different output combinations	45
4.11	Boxplot of results for 7v1 for different output combinations	46
4.12	Examples of images of 7v1 with different output combinations	46
4.13	Histogram of results for 7v1 with no output combination	47
4.14	Boxplot of results for 7v1 with no output combination	47
4.15	Histogram of No Output Combination with Gaussian loss (weight=0.005; var=0.001)	48
4.16	Examples of images of 7v1 experiment for No Output combination	48
4.17	Histogram of results for 9v8 with different output combinations	49
4.18	Boxplot of results for 9v8 with different output combinations	49
4.19	Examples of images of 9v8 with different output combinations	50
4.20	Histogram of results for 9v8 with no output combination	50
4.21	Boxplot of results for 9v8 with no output combination	51
4.22	Examples of images of 9v8 experiment for No Output combination	51

B.1	Experiments diagram	73
E.1	Comparison of ACD during training, Mean Estimator, CD loss weights 1 and 10 .	81
E.2	Comparison of ACD during training, Mean Estimator, CD loss weights 25 and 50	81
E.3	Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02	82
E.4	Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05	82
E.5	Comparison of ACD during training, Linear Estimator, CD loss weights 1 and 10	83
E.6	Comparison of ACD during training, Linear Estimator, CD loss weights 25 and 50	83
E.7	Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02	83
E.8	Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05	84
E.9	Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 1 and 10	85
E.10	Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 25 and 50	85
E.11	Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02	86
E.12	Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05	86
E.13	Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002	87
E.14	Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002	87
E.15	Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.03 and vars 0.005	87
E.16	Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.25 and 0.50	88
E.17	Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.75	88
E.18	Comparison of ACD during training, Mean Estimator, CD loss weights 1 and 10 .	89
E.19	Comparison of ACD during training, Mean Estimator, CD loss weights 25 and 50	89
E.20	Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02	90
E.21	Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05	90
E.22	Comparison of ACD during training, Linear Estimator, CD loss weights 1 and 10	91
E.23	Comparison of ACD during training, Linear Estimator, CD loss weights 25 and 50	91
E.24	Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02	91
E.25	Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05	92
E.26	Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 1 and 10	93
E.27	Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 25 and 50	93

E.28	Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02	94
E.29	Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05	94
E.30	Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002	95
E.31	Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002	95
E.32	Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.03 and vars 0.005	95
E.33	Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.25 and 0.50	96
E.34	Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.75	96
F.1	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01	97
F.2	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02	98
F.3	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03	98
F.4	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05	99
F.5	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01	100
F.6	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02	100
F.7	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03	101
F.8	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05	101
F.9	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01	102
F.10	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02	102
F.11	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03	103
F.12	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05	103
F.13	Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.005 and var 0.001 and KL div loss weight 0.25	104
F.14	Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.015 and var 0.002 and KL div loss weight 0.50	104
F.15	Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.03 and var 0.005 and KL div loss weight 0.75	105
F.16	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01	106
F.17	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02	107

F.18	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03	107
F.19	Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05	108
F.20	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01	109
F.21	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02	109
F.22	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03	110
F.23	Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05	110
F.24	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01	111
F.25	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02	111
F.26	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03	112
F.27	Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05	112
F.28	Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.005 and var 0.001 and KL div loss weight 0.25	113
F.29	Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.015 and var 0.002 and KL div loss weight 0.50	113
F.30	Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.03 and var 0.005 and KL div loss weight 0.75	114
G.1	Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, Mean Estimator, all hyperparams variations	115
G.2	Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, Linear Estimator, all hyperparams variations	116
G.3	Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, Meta-Learner Estimator, all hyperparams variations	117
G.4	Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, No Output Estimator, all hyperparams variations	118
G.5	Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, Mean Estimator, all hyperparams variations	119
G.6	Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, Linear Estimator, all hyperparams variations	120
G.7	Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, Meta-Learner Estimator, all hyperparams variations	121
G.8	Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, No Output Estimator, all hyperparams variations	122

List of Tables

3.1	Generator Model Summary	28
3.2	Discriminator Model Summary	28
3.3	Example Classifier Model Summary	29
3.4	Different Configurations and Hyperparameters	35
3.5	Training Steps Summary	36
4.1	DCGAN architecture changes comparison for 7v1	38
4.2	DCGAN architecture changes comparison for 9v8	39
4.3	Gaussian Loss comparison for 7v1	41
4.4	Gaussian Loss comparison for 9v8	43
4.5	Output combination comparison for 7v1	45
4.6	Output combination comparison for 9v8	49
4.7	RESNET-50's Hubris evaluation for 7v1	52
4.8	RESNET-50's Hubris evaluation for 9v8	53
A.1	Overview of different GAN architectures	72
C.1	Accuracy for different Ambiguity Estimators used in AmbiGAN training for im- age generation	74
C.2	Accuracy for different Ambiguity Estimators used in Hubris Evaluation	74
C.3	Accuracy of pre-trained RESNET-50 for Hubris Evaluation	74
D.1	Full results for Mean Output Combination Estimator 7v1	75
D.2	Full results for Linear Output Combination Estimator 7v1	76
D.3	Full results for Meta-Learner Output Combination Estimator 7v1	76
D.4	Full results for No Output Combination Estimator 7v1	77
D.5	Full results for Mean Output Combination Estimator 9v8	78
D.6	Full results for Linear Output Combination Estimator 9v8	79
D.7	Full results for Meta-Learner Output Combination Estimator 9v8	79
D.8	Full results for No Output Combination Estimator 9v8	80
H.1	RESNET-50's Hubris evaluation for 7v1 for all types of output	123
H.2	RESNET-50's Hubris evaluation for 9v8 for all types of output	124

Abbreviations

AAE	Adversarial Auto Encoder
ACD	Average Confusion Distance
ACGAN	Auxiliary Classifier Generative Adversarial Network
AE	Auto Encoder
AI	Artificial Intelligence
AmbiGAN	Ambiguous Generative Adversarial Network
AUC	Area Under the Curve
BiGAN	Bidirectional Generative Adversarial Network
CIFAR	Canadian Institute for Advanced Research
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CV	Computer Vision
CycleGAN	Cycle Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
DNN	Deep Neural Network
EM	Earth-Mover
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
GASTeN	Generative Adversarial Stress Test Network
GNLL	Gaussian Negative Log-Likelihood
GRAN	Generative Recurrent Adversarial Network
HMM	Hidden Markov Models
IBA	Index Balanced Accuracy
InfoGAN	Information maximization Generative Adversarial Network
IS	Inception Score
JS	Jensen-Shannon
KL	Kullback-Leibler
LeakyReLU	Leaky Rectified Linear Unit
LogL	Logistic Loss
MAE	Mean Absolute Error
MAP	Maximum <i>a posteriori</i>
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
OCR	Optical Character Recognition
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Oversample Technique
SVM	Support Vector Machine
Tanh	Hyperbolic Tangent
VAE	Variational Auto-Encoder
WGAN	Wasserstein Generative Adversarial Network
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

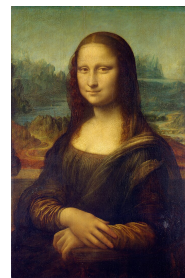
The increase in Artificial Intelligence (AI) applications in diverse fields has been evident in recent years. The current availability of large volumes of data and higher computational power is now practically granted to the average user. This wider availability also presents risks.

Intelligent systems progressively support and integrate decision-making processes in sectors such as autonomous driving, health care diagnosis, finance, and manufacturing. Taking an autonomous car as an example, the computer vision algorithm on these cars must be capable of dealing with unseen or even truly ambiguous scenarios since failure to do so can ultimately harm humans. Acknowledging these systems will also be presented with ambiguous information is important. At the same time, some models used to make these decisions are hard for humans to interpret and are considered black boxes. This lack of model interpretation leads to the need for methodologies and techniques that enable a responsible usage of Machine Learning (ML) and AI – the concept of responsible AI [1]. Stress-testing of models before deployment can help to improve its explainability and transparency. The availability of benchmark datasets, especially targeting ambiguous instances, could be a reliable tool for assessing a model’s robustness towards ambiguity.

Classic examples of ambiguity in images (“My Wife and My Mother-in-Law” and Leonardo DaVinci’s “Mona Lisa”) are shown in Figure 1.1. As a toy example for concept clarification, we are brought to the famous question: can a human classify if the woman in the painting is smiling?



(a) My Wife and My Mother in Law [2]



(b) Mona Lisa [3]

Figure 1.1: Examples of ambiguous images

1.1 Motivation

Responsible use of AI necessitates thorough documentation of algorithms to ensure transparency and accountability in their development and deployment [1]. Ensuring models are robust and cautious when faced with ambiguity is crucial for their safe and reliable application. However, more studies are needed to understand a model’s vulnerabilities and weaknesses.

Current benchmarks often fail to adequately evaluate how models handle uncertainty, which is critical in understanding these referred vulnerabilities. Specifically, the stress testing of models with challenging or ambiguous data is not yet well defined and understood. It is crucial not only for ambiguous data to be available but also to quantify a model’s behavior towards it, enabling a more comprehensive understanding of model performance in uncertain scenarios. This highlights the relevance of understanding the model’s overconfidence when facing ambiguous instances and quantifying its robustness in these scenarios.

By understanding and reporting the conditions under which models become less reliable, we aim to promote the responsible use of ML. This understanding can help ensure that AI systems generalize better to various contexts, including sensitive domains. Therefore, the responsible use of AI promotes technological innovation ensuring the sustainable industrialization of these domains.

1.2 Goals and Research Questions

This work addresses the common occurrence of ambiguous scenarios by stress testing models and considering ambiguity. These models should exhibit caution and limited confidence when faced with ambiguous or uncertain instances.

The primary goal of this work is to generate challenging datasets that test the overconfidence in machine learning models. We expect these datasets to function as companion datasets for benchmarking. We will apply this general approach to Computer Vision (CV) in the context of classification tasks. The objective behind generating companion datasets is to improve the development of AI models. We want to generate realistic images containing features of two or more classes simultaneously. In this situation, each classifier should present a class probability above zero for at least two possible classes [4]. This is one possible approach to tackle the concept of ambiguity in images.

Using the generative adversarial network framework and the previous work of Cunha *et al.* in GASTeN [5], the intended approach is to adapt its architecture to include an additional step of multiple classifiers. This group of diverse classifiers will act as ambiguity estimators. The output of this unit will serve as part of an additional loss term for the generator. We expect the generated images to be challenging because the data lies on the decision border of these several different classifiers.

To empirically validate the dataset generated by this approach, we first propose a metric to quantify the overconfidence of a given model: Hubris. In the second step, we propose to stress test

pre-trained models with our generated images and to determine their Hubris. For comparison, we can also stress test the pre-trained models with other ambiguous datasets in the literature. In this way, this work aims to answer the following research questions:

1. How to improve GASTeN architecture for increased performance?
2. How to empirically estimate the ambiguity of an example?
3. How to adapt GASTeN to generate universally ambiguous examples?
4. How to measure the robustness to ambiguity of a model on a set of ambiguous examples?

This work also aims to contribute to the Sustainable Development Goals, particularly Goal 9: Industry, Innovation, and Infrastructure [6], by enhancing the reliability and accountability of AI models. It intends to contribute to the development of robust infrastructure and to promote technological innovation. This, in turn, supports sustainable industrialization and fosters economic growth. By promoting responsible AI usage, this research aims to mitigate potential negative impacts on society, thus contributing to the broader aim of sustainable development.

1.3 Publications

The following conference paper based on the initial research conducted in this dissertation has been accepted:

Teixeira, C., Gomes, I., Cunha, L., Soares, C. & Rijn, J. N.

"GASTeNv2: Generative Adversarial Stress Testing Networks with Gaussian Loss"

Proceedings of the 23rd International Conference on Artificial Intelligence (EPIA 2024), Viana do Castelo, Portugal, September 2024

A second paper is being prepared for submission with the additional contributions of this dissertation.

1.4 Outline

Besides the current chapter, the remaining of this document contains the following sections:

- Chapter 2, **Literature Review** introduces essential concepts to understand better the proposed work and current state-of-the-art topics relevant to this research. It includes highlights of the advancements in the field and identifies gaps that this research aims to address.
- Chapter 3, **Methodology** describes the approach taken in this work and introduces our Hubris benchmarking proposal. It also describes generating the companion MNIST dataset, including designing and implementing our proposed framework, AmbiGAN. The chapter

outlines the experimental setup, discusses the Ambiguity Estimators created, the calculation of Hubris and Confusion Distance metrics, and the application of Gaussian Loss and other loss functions to the AmbiGAN implementation.

- Chapter 4, **Results** presents the results of the experiments conducted. The improved DC-GAN architecture is discussed in relation to the original GASTeN. It discusses the impact of using different combinations of the Ambiguity Estimator output and the impact on AmbiGAN performance. Hubris benchmark results for RESNET-50 are presented and discussed.
- Chapter 5, **Conclusion** summarizes the key findings and their implications, discusses the work limitations, and proposes directions for future work. The chapter answers the research questions posed at the beginning of the study, demonstrating how the findings address the initial hypotheses and contribute to the field.

Chapter 2

Literature Review

This chapter introduces theoretical concepts and applications useful for understanding the proposed work, providing insights into currently relevant areas of study and research. Section 2.1 discusses the probability models for discriminative methods, particularly image classification and generative methods (including image generation). Distance metrics, losses, multi-objective training, and evaluation methods for those tasks are also discussed. Section 2.2 approaches the topics of diversity, uncertainty and ambiguity estimation. In section 2.3, the topic of generative adversarial networks and several of its architectures that have been developed since its first original model are introduced in Section 2.3.1. The topics of adversarial attacks are addressed in Section 2.3.2. Stress testing is discussed in Section 2.4. Section 2.4.1 discusses ambiguous images in classification.

2.1 Background

We can divide ML into supervised and unsupervised learning based on whether the samples have labeled information. Supervised learning can then be further divided into classification problems and regression problems [7].

Like many ML tasks, computer vision aims to extract useful information from data (images). In this way, computer vision problems use the observed image data to infer information about the world [8]. A more precise description should state that we want to take some observations x and return the probability distribution $P(w|x)$ over world states w [9], or conversely, taking w and returning $P(x|w)$. Concerning the world state, it can either be continuous (e.g. the 3D location of an object) or discrete (e.g. a characteristic or a feature of the observed object). When the world state is continuous, we have a regression problem; when the state is discrete, we have a classification problem [10].

2.1.1 Probability Models

Learning one of the distributions $P(w|x)$ or $P(x|w)$ consists of finding a model's parameters θ , which different methods can determine. The most popular approach is probably maximum likelihood estimation [11].

To find the likelihood function $P(x_i|\theta)$ at a single point x_i , we evaluate the probability density function at x_i [12]. Consequentially, the likelihood for a set of points is the product of the individual likelihoods. This concept, first derived by Fisher [13], can be formalized in Equation 2.1.

$$\hat{\theta}_{\text{ML}} = \arg \max_{\theta} \prod_{i=1}^n P(x_i | \theta) \quad (2.1)$$

Other methods include Maximum *a posteriori* (MAP) and the Bayesian approach [10], [13]. The MAP approach considers previous information learned about θ from past observations. In this way, MAP maximizes the posterior probability of the parameters. This results in Equation 2.2, similar to Equation 2.1 except for the additional prior term. This method is standard for the probabilistic function approximation in Hidden Markov Models (HMM) [14].

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} \left[\prod_{i=1}^n P(x_i | \theta) \cdot P(\theta) \right] \quad (2.2)$$

In the Bayesian approach, the perspective is not to estimate single fixed values but rather to model probability distributions over possible values of parameters, given that many parameters may be compatible with the data. In this way, the computation is based on the Bayes rules, resulting in the following Equation 2.3. This approach is useful for its ability to incorporate prior knowledge and deal with uncertainty [15].

$$P(x^* | x_1 \dots x_I) = \int P_r(x^* | \theta) \cdot P(\theta | x_1 \dots x_I) d\theta \quad (2.3)$$

We can view the integral as a weighted sum of the predictions given by different parameters θ , with the weighting determined by the posterior probability distribution $P(\theta | x_1 \dots x_I)$ over the parameters — which represents the confidence that different parameter values are correct [10].

There is an alternative perspective in probability modeling that enables the maximization of uncertainty under certain constraints. Maximum entropy theory has many connections to maximum likelihood and dates back to early work on Information Theory by Shannon [16], Jaynes [17], Kullback [18] and Levine [19]. The approach proposes that a distribution should satisfy constraints on its moments while maximizing entropy to have the least amount of information (also referred to as the principle of indifference or insufficient reason). This means maximum entropy finds a distribution $P(x)$ whose entropy H is as large as possible while satisfying certain moment or expectations constraints [13]. Equation 2.4 formalizes the approach.

$$P(x) = \arg \max_x - \sum_x P(x) \log P(x) \quad (2.4)$$

One could minimize the Kullback-Leibler (KL) divergence or the relative entropy to the uniform distribution instead of maximizing the entropy. This will be further discussed in Sections 2.1.4, 2.1.6, and 2.3.1.4.

2.1.2 Discriminative Methods

Using discriminative methods, we can find the probability for the world states w given the observations x (referred in Section 2.1). Our goal is to model (θ) a distribution over w that we can apply in a new observation x [20]. The distribution over the state depends on both the data and these model parameters; we can define it as $P(w|x, \theta)$ and refer to it as the posterior distribution, which can be read as everything we know about the state after observing x [10].

Discriminative approaches are not concerned with modeling the underlying distributions of the variables and features in a system. They are only interested in optimizing a mapping from the inputs to the desired outputs. In this way, only the resulting classification boundary — or function approximation accuracy for regression — is adjusted [13]. This is an advantage in high-dimensional spaces where capturing the complete data distribution may be computationally challenging or unnecessary. The focus on optimizing the decision boundary enables discriminative models, such as Deep Neural Networks (DNN), to handle complex relationships within the data efficiently. They are well-suited for tasks like image classification, speech recognition, and natural language processing [21].

2.1.2.1 Image Classification

For the case of classification problems on images, the task is for a computational machine to examine an image and distinguish the class that the image falls under [22]. The images are categorized based on the recognized class categories for which the ML model was trained. Image classification solves real-world challenges, such as Optical Character Recognition (OCR), machine inspection, medical imaging, and self-driving vehicles [23].

ML models for image classification problems use architectures such as Convolutional Neural Networks (CNNs) to improve prediction performance. The capacity of CNNs can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images [24]. CNNs were first introduced in 1989 by LeCun *et al.*, which reported enormous potential in computer vision-related tasks [25]. LeCun *et al.* developed this new architecture by adding back-propagation and incorporating convolutional operations, already used on the Neocognitron architecture from Fukushima in 1980 [26]. Fukushima reported that Neocognitron had an implementation on which input patterns were classified even if they were shifted or distorted [26].

LeCun *et al.* laid the foundations for the modern computer vision problems approach, enabling the general application of this field in a much more efficient methodology and broadening the scope of its usage.

CNNs are trainable multistage architectures composed of multiple stages where the input and output of each stage are sets of arrays called feature maps [27].

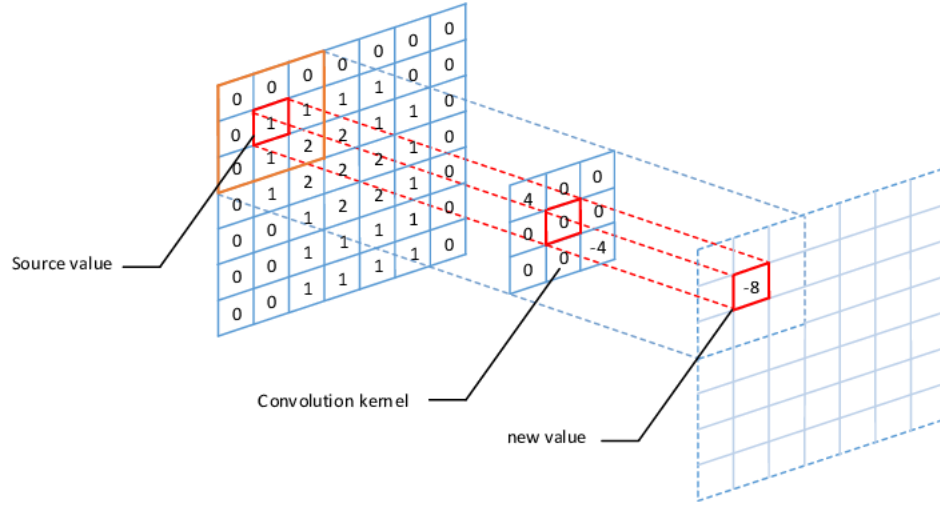


Figure 2.1: Convolution process [28]

CNNs make use of kernels, also known as filters. They are similar to weights that are trained to detect features throughout an image. They spatially convolve the kernels with a given input and check if the feature it is meant to detect is present in the input image [29]. This process is illustrated in Figure 2.1. The CNNs' architecture also usually includes non-linear processing units, sub-sampling layers, and fully connected layers [30]. The output of the convolutional kernels is assigned to the non-linear processing units (activation function), which helps in learning abstractions and embeds non-linearity in the feature space [31].

To speed up the training process and reduce the amount of memory consumed by the network, the convolutional layer is often followed by a pooling layer to remove redundancy in the input features [29]. Average and max pooling are the most common layers used for pooling, with approaches such as stochastic, spatial pyramid, and def pooling [32].

The fully connected layers, generally at the end of each CNN architecture, work as a CNN classifier. Their output can be used for image classification problems or transferred to a subsequent DNN [33].

2.1.3 Generative Methods

Approaches that explicitly or implicitly model the distributions of inputs and outputs are known as generative models [34]. Unlike discriminative methods, generative methods aim to find a model with parameters θ and make the distribution parameters a function of the world state w . In other words, we search for the θ that achieves $P(x|w, \theta)$ [10].

These models learn to capture the statistical distribution of training data. It allows us to synthesize samples from the learned distribution. In addition to synthesizing new data samples, these may be used for downstream tasks such as data augmentation [35], semantic image editing [36],

style transfer [37] and super-resolution imaging [38, 39]. Beyond these applications, the representations that such models learn are valuable for tasks like classification [40], image retrieval [40], anomaly detection [41], and facial attribute manipulation [42].

2.1.3.1 Image Generation

One of the uses of generative methods is synthetic image generation. The need to generate synthetic data mainly arises from the difficulties in data acquisition. Both the quantity and quality of images for training data purposes present challenges. In addition, the inherited uncertainty that manually annotated train datasets is also a concern [43]. Since most ML models usually require large quantities of labeled data, synthetic images are a key advantage for faster and cheaper training data generation and labeling process [44]. Before CNNs, image generation techniques relied mainly on manual feature extraction. With the rise of CNN models, the synthetic image generation task has become less time-consuming. It attracted researchers to explore this topic even more. This task can be divided into many categories, including image generation, text-to-image, image-to-image translation, sketch-to-image generation, conditional image generation, between others [45].

In the cases of synthetic image generation for data augmentation or enhancement, the synthetic data must be generated so that it is plausible and follows the underlying distribution of the original data [46]. Data bias is a known issue with synthesized images, often caused by the inheritance of the generation's input parameters [44].

Synthetic image generation techniques have also been successfully applied for generating adversarial examples, where the synthetic images look very similar to the real ones but can mislead classifier models so that the predicted output is wrong [47]. For example, adding a small vector of noise with specific values can result in synthetic adversarial images with error rates on misclassification that go up to 87% [48]. Section 2.3.2 further discusses these adversarial attacks.

Turhan and Bilge [49] categorize recent studies on deep generative ML models such as encoder-decoder based (AE), auto-regressive based, Generative Adversarial Network (GAN) based, and AE-GAN hybrid models. The authors conclude that GAN and Variational AE (VAE) based models are causing increasing interest since they can be widely and broadly used on computer vision tasks [49]. In the later case of VAE, the work originally developed by Kingma and Welling in 2013 [50] aims to model a parametric latent variable by maximizing the marginal log-likelihood of the training data [51]. Diffusion models [52] is also a state-of-the-art technique that can be used for image generation. These include a probabilistic generative model that iteratively deconstructs original data by injecting noise but then learns to reverse this process for new sample generation [53].

Nevertheless, image generation techniques are based on more than DNN models. Synthetic Minority Oversample Technique (SMOTE), proposed in 2002, is a well-known algorithm that tries to replicate a data distribution [54]. More traditional approaches also include random over and under-sampling techniques such as borderline-SMOTE (oversampling) [55], Tomeks' link (undersampling) [56], and cluster centroids (undersampling) [57].

2.1.4 Distance Measures and Loss Functions

Distance measures designed to quantify the dissimilarity or distance between probability distributions are of particular interest in the context of deep learning since the goal of training is to minimize the difference between the predicted and expected outputs of the model [58]. Hence, distance measures are often employed in loss functions when dealing with tasks that involve comparing or matching probability distributions.

Divergence or distance measures are also considered generalized entropy measures, reflecting a certain entropy distance between two distributions — say a prior and an empirical one — or two distributions competing under different scenarios or propositions [59].

The elementary distances and divergences between two distributions can be measured by several methods, including the Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) divergence, Earth-Mover (EM) distance (or Wasserstein distance) [60]. KL divergence is defined in Equation 2.5. Both P_r and P_g are assumed to admit densities for real and predicted or generated data. The KL divergence is asymmetric and possibly infinite at some points [60, 61]. JS divergence defined in Equation 2.6 is a symmetric and smoothed version of the KL divergence [61].

$$KL(P_r || P_g) = \int_{\mathcal{X}} \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) dx \quad (2.5)$$

$$JS(P_r, P_g) = \frac{1}{2} KL \left(P_r || \frac{P_r + P_g}{2} \right) + \frac{1}{2} KL \left(P_g || \frac{P_r + P_g}{2} \right) \quad (2.6)$$

While the KL and JS distances are information gain or loss measures when using one distribution to approximate another, they lack a direct geometric interpretation. To address this limitation, a measure such as the EM distance can be relevant in situations where the underlying geometry of the space is important. EM, also known as Wasserstein distance, is defined in Equation 2.7. where $\Pi(P_r, P_g)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively P_r and P_g . $\gamma(x, y)$ indicates how much *mass* must be transported from x to y to transform the distributions P_r into the distribution P_g . The EM distance can be viewed as the “cost” of the optimal transport plan [60].

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|] \quad (2.7)$$

These distances mentioned are commonly used as loss functions in the image generation task [58]. For image classification, the most common losses include but are not limited to cross-entropy loss with its different variations (binary, weighted binary, categorical, sparse categorical, and label smoothing), focal loss, and Hinge loss. Most of these losses, except Hinge loss, are centered around the information theory concept of entropy reflecting the dissimilarity between two or more probability distributions [58].

Hinge loss, which is popular on Support Vector Machines (SVMs) implementations [62] is, as previously said, not centered around the entropy theory and is commonly used in one-vs-all classification problems [58]. It is defined in Equation 2.8. Where y is the true label of the sample and $f(x)$ is the predicted label of the sample (which is -1 to 1 if in binary classification).

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x)) \quad (2.8)$$

Entering into the entropy-based losses, we can find focal loss [63] as designed to deal with imbalanced datasets, which are modified based on sigmoid cross-entropy [7]. It is defined in Equation 2.9. Where p_t is the predicted probability for the true class, α_t is a weighting factor controlling the importance of each example, and γ is a focusing parameter with values generally set between 2 and 4 [58].

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \quad (2.9)$$

Finally, one variant that deals with multi-class can be highlighted in cross-entropy-based losses. The categorical cross-entropy, also known as negative log-likelihood loss, is defined in Equation 2.10. Where N is the number of examples, C is the number of classes, y is the true label, and p is the predicted probability of the true class [64].

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C y_{i,j} \log(p_{i,j}) \quad (2.10)$$

The Gaussian Negative Log Likelihood Loss (GNLL) is used to estimate both the mean (μ) and variance (σ) of a target's probability distribution as a function of the input. It is based on a presumed model of the target's error-distribution [65]. The GNLL formula, as outlined in Equation 2.11, incorporates these parameters where σ^2 represents the variance, μ indicates the mean of the distribution, and y is the observed actual value. According to this formulation, a lower GNLL value suggests that the actual value is closer to the predicted mean, aligning more closely with the expected target distribution.

$$GNLL = \frac{1}{2} \log(\sigma^2) + \frac{(y - \mu)^2}{2\sigma^2} \quad (2.11)$$

One research work uses Gaussian loss to improve model performance in deep age estimation [66]. The researchers employed a CNN to predict age from images. Their approach incorporates a mean-variance loss, which includes a mean loss to minimize the discrepancy between the predicted mean of the age distribution and the actual age, alongside a variance loss that penalizes

the variance in the estimated age distribution, aiming to produce more concentrated predictions. This dual-component loss structure proved adequate, as the authors demonstrate that their model surpasses existing benchmarks in age estimation, achieving state-of-the-art results.

2.1.5 Multi-objective Training

The traditional ML methods perform single-objective optimization on some simple consolidation of the losses. However, when models need to balance multiple goals simultaneously, multi-objective functions come into play. Multi-objective training involves optimizing a model for a given set of objectives. This type of problem often has multiple optimal solutions [67].

A typical use case for training with multi-objective functions is using the Weighted Sum method. This method involves the selection of weighting coefficients $w_i \geq 0$, correspondingly to the objective function involved (f_i for $i = 1, \dots, K$). Usually, weights are normalized such that $\sum_{i=1}^K w_i = 1$, as defined in Equation 2.12.

$$\text{minimize} \quad \sum_{i=1}^K w_i f_i(\mathbf{x}) \quad (2.12)$$

Other approaches include the ε -constrained method, which optimizes one objective while treating others as constraints, and the Gradient-based method, which iteratively adjusts the weights and directions to navigate the trade-offs among objectives [68].

An example of multi-objective training is GANs with multiple discriminators [69]. The training process involves balancing the feedback from all discriminators. The generator must optimize its output to perform well against all discriminators simultaneously. In this example, multi-objective optimization is required since each discriminator provides a different objective function that the generator needs to satisfy.

2.1.6 Evaluation

Many criteria can be used to evaluate discriminative and generative models, and each set of one or more criteria is appropriate in different settings and conditions. On discriminative models, for example, we usually do not know how to train models to predict the true underlying probabilities [70]. In the generative models' case, we may lack an objective function that can be used to compare different models [71] or a metric that measures the effectiveness of synthetic data [44].

The evaluation of discriminative models or classification tasks usually involves dealing with three types of quantitative metrics: threshold-based, probability-based, and ranking-based [70].

Threshold-based metrics include accuracy, precision, recall, F-measure (F-score), and Kappa statistic; probability-based includes information gain and entropy metrics such as mean absolute error (MAE), mean squared error (MSE), and logistic loss (cross-entropy); whilst ranking metrics include area under the receiver operating characteristic (ROC) Curve — AUC [72, 73]. For imbalanced datasets, a threshold-based index balanced accuracy (IBA) is also recommended [74].

Logistic loss (LogL) is also known as cross-entropy and measures how good the probability estimates of the classifier are [73]. Equation 2.13 defines it.

$$\text{LogL} = \frac{-\sum_{j=1}^c \sum_{i=1}^m f(i, j) \log_2 \max(p(i, j), \epsilon)}{m} \quad (2.13)$$

With i being the example number, j class number when in multi-class classification, $f(i, j)$ represents the model prediction with a value between 0 and 1 for a given example/class pair, and $p(i, j)$ represents the boolean decision value which is 1 if model decided that example belongs to given class or 0 otherwise [73]. ϵ is typically set to a small value such as 1^{-5} , to avoid $\log_2(0)$ cases.

As per generative models, evaluating synthetic data is important since, as previously stated, the generated data must be plausible and follow the intrinsic real data distribution [46]. Qualitative and quantitative measurements can be used to evaluate the generated data, where qualitative is subject to interpretation and ranking; quantitative follows a set of numeric metrics or statistics [75].

Qualitative measurement can be done using human ranking, as applied in [76]. As expected, using qualitative measures is highly subjective to human perception and evaluation, which unavoidable may lead to high variation in ranking given a group of human evaluators [75].

As per quantitative measures, the default and most used common metric considered in literature is the average log-likelihood, an expansion of Equation 2.1. As Theis *et al.* summarised, this metric is flawed since models with large log-likelihood values can lead to visually unpleasing images or over-fitted data otherwise [77]. To circumvent this, metrics such as inception score (IS) and Fréchet inception distance (FID) come as an alternative.

As proposed by Salimans *et al.* in 2016, the IS evaluates the quality of generated images by simulating human judgment [71]. The authors of [71] state that IS values are correlated very well with human rating values when using a large enough number of samples (i.e. 50k). IS values are calculated by forwarding synthetic samples on a trained classification model and calculating the score defined by Equation 2.14. The IS computes KL-divergence [78] between the conditional distribution $P(w|x)$, and the marginal distribution $P(w)$, where x is the synthetic sample, and w the label predicted by the Inception Model [79]. The higher the IS values are, the better the generative model on quality and diversity of generated samples [75].

$$\text{IS} = \exp \left(\mathbb{E}_{x \sim P(x)} D_{\text{KL}}(P(w|x) || P(w)) \right) \quad (2.14)$$

FID, introduced by Heusel *et al.* in 2017, captures similarities between synthetic and real images better than IS [80]. It considers both the synthetic and real images' distributions based on the extracted visual features [75]. FID is calculated by taking as input the mean (m) and variance (C) of the two resulting Gaussian distributions (synthetic and real images' features) by using m_r ,

C_r for real images, and m_w, C_w for synthetic ones, as follows as described in Equation 2.15. Since FID represents a distance, the lower its value, the more similar the two distributions are [81].

$$FID((m_r, C_r), (m_w, C_w)) = \|m_r - m_w\|_2^2 + \text{Tr} \left(C_r + C_w - 2(C_r C_w)^{1/2} \right) \quad (2.15)$$

2.2 Diversity, Uncertainty and Ambiguity Estimation

Real-world AI systems must react to their inputs, even in unexpected circumstances. It is important to know the uncertainty of a model given unfamiliar examples [82]. Out-of-distribution examples are instances that differ significantly from the data distribution on which a machine-learning model was trained. In out-of-distribution scenarios, the feature statistics often show inconsistency with the training domain due to different domain characteristics [83]. For example, if a network trained on one dataset is evaluated on a completely different dataset, the network should output high predictive uncertainty as inputs from a different dataset would be far away from the training data [84].

It is important to acknowledge the cause of uncertainty in model predictions. Uncertainty in predictions refers to the degree of confidence a machine learning model has in its output. Predictions are typically made under uncertainty, which can be epistemic or aleatoric. Epistemic uncertainty originates from the model itself and can be removed by better training of the model and possibly on better training data. Aleatoric uncertainty is model-independent and is inherent in the instance itself due to nondeterministic events [4]. Aleatoric uncertainty is also known as data uncertainty and arises from the inherent ambiguity in the data [85].

In uncertainty estimation, diversity among models plays a crucial role in capturing and understanding ambiguity [84]. When multiple classifiers make independent predictions, the variance in their outputs provides insight into the uncertainty of the prediction. This concept is explored in various works, highlighting how ensemble methods benefit from diverse models [86].

Techniques such as bagging, which involves training multiple models on different subsets of the training data, can enhance diversity among classifiers. At the same time, the overall uncertainty estimation is improved [87].

Having diversity among models also results in them not fully agreeing with all predictions. This disagreement is also advantageous since it provides another useful qualitative way to evaluate a model uncertainty [84].

Lakshminarayanan *et al.* [84] proposed ensembles of multiple deep models for uncertainty estimation, which was simple and scalable. A deep ensemble comprises multiple neural networks of the same architecture. The individual learners are diversified by random initialization and random shuffling of training examples. The deep ensemble method effectively detects out-of-distribution inputs from image datasets [88].

Rahaman and Thiery [89] state the benefit of averaging the predictions of a set of DNNs: in the often countered regime when each network is overconfident, averaging the predictions of a set of neural networks generally leads to less confident predictions.

An advantage of ensembles using DNNs is the possibility of distinguishing between aleatoric and epistemic uncertainty. The different predictions mean can be interpreted as aleatoric uncertainty. Conversely, the standard deviation of these predictions represents epistemic uncertainty [90].

Bayesian Inference, Markov Chain Monte Carlo (MCMC), and Test-time Augmentation (TTA) are other techniques used to assess uncertainty in machine learning models. Bayesian inference updates beliefs about model parameters using Bayes' theorem, providing a probabilistic framework for uncertainty quantification [91, 92, 93].

Markov chain Monte Carlo (MCMC) samples from complex probability distributions, such as posterior distributions in Bayesian models, to estimate uncertainty and variability in predictions [94, 95, 96].

Test-time Augmentation (TTA) involves applying various transformations to the input data during the testing phase, running the model on each transformed version, and combining the results to produce a final prediction. When using TTA, multiple predictions are generated from various augmented versions of the same input. Analyzing the variation among these predictions can provide insights into the model's uncertainty. The goal of data augmentation is to obtain additional examples for a dataset so that a machine learning algorithm can learn from a larger and preferably better set of examples [97, 98, 99].

2.3 Generative Adversarial Networks

In 2014, Ian Goodfellow *et al.* [100] proposed a new framework for estimating generative models via an adversarial process corresponding to a minimax two-player game in the sense of game theory. GANs are implicit models that infer the probability distribution $p(x)$ without necessarily representing the density function explicitly [11].

GANs often consist of a generator (G) and a discriminator (D) that learn simultaneously. The generator aims to capture the potential distribution of real samples and generate new data samples. The discriminator is often a binary classifier, discriminating real samples from the generated samples as accurately as possible [101]. G and D are typically implemented by multilayer networks consisting of convolutional and fully connected layers [102]. Noise z is fed into G , which produces a fake sample $\hat{x}$. D receives either $\hat{x}$ or a real sample x and has to predict the sample class $\hat{y}$ (either real or fake). The generator has no direct access to real images and only learns through interaction with the discriminator [102].

The optimization goal is to reach Nash equilibrium [103] when the generator can be considered to have accurately captured the distribution of real examples. Hence, it terminates at a saddle point that forms a minimum with respect to the generator and a maximum concerning the

discriminator [34]. The original objective function in GANs can be formalized in the Equation 2.16.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.16)$$

The key to this technique lies in the loss functions. The loss function for D revolves around the discrepancy between real and generated data predictions. At the same time, for G , it focuses on the negative mean of the discriminator's evaluation of the generated data — this, and the adversarial setup, encourages the generator to produce increasingly realistic data that the discriminator will misclassify as real [100]. The visual representation of this process can be found in Figure 2.2.

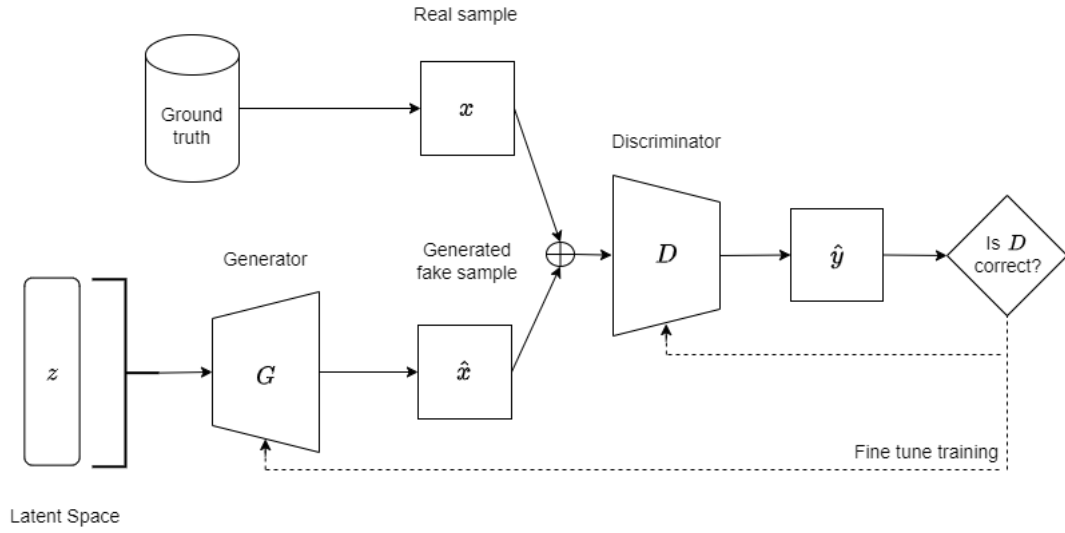


Figure 2.2: GAN strucutre

GANs appeared with various advantages over the existing models at the date, e.g. Boltzmann machines [104]. Since these techniques are based on Markov chains, GANs were designed to avoid this property due to their associated computational constraint. This was a serious issue solved by the newest techniques. Nevertheless, there was also a lack of generalization in these Markov chain-based techniques.

Despite the significant successes achieved to date, the application of GANs still presents significant challenges, such as the generation of high-quality images, diversity of image generation, and stabilizing training [105].

2.3.1 Architectures

Since the original formulation, several different GAN architectures have been proposed. Each of these architectures aims to solve a specific problem scope. Table A.1 aims to summarize the most commonly proposed architectures. In the following sections, we dive into the most relevant ones.

2.3.1.1 Deep Convolutional GANs

In 2015, Radford *et al.* proposed a class of GANs called the Deep Convolutional GANs (DCGANs) [40], establishing a more stable architecture for training GANs. The guidelines include replacing any pooling layer with stride convolutions for the discriminator and fractional-strided convolutions for the generator. They also include using batch normalization, removing all fully connected hidden layers, and using Rectified Linear Unit (ReLU) activation in the generator for all layers except the output - which uses the hyperbolic tangent (Tanh) activation function. Using a Leaky Rectified Linear Unit (LeakyReLU) is also advised for all layers of the discriminator. These modifications have contributed to the robustness and effectiveness of GAN training, making DCGANs a widely adopted standard in the field.

2.3.1.2 Conditional GANs

GANs have, in the original formulation, an unconditional generative model that cannot control the classes of images that are generated [100]. To control this process, additional conditional information can be learned by the GANs so that the generation process is also conditioned. The condition can be any auxiliary information about the data [106].

This Conditional GAN (CGAN) approach induces the condition on both the generator and discriminator during the training process, which aims to produce realistic images given the condition [107]. In CGANs, the conditions are concatenated into some layers of generator and discriminator, and the architecture itself must be able to effectively disentangle the intrinsic image factors (which are encoded in the latent space) and the extrinsic image factors (which are the conditions) [107].

2.3.1.3 GANs with Inference Models

The basic GANs architectures lack an efficient inference mechanism [107]. In simpler terms, GANs' latent space z cannot be inferred by having the generated images *a priori*. Some variations, such as adversarially learned inference [108], and bidirectional GAN (BiGAN) [109], can learn the representations within the GAN framework by using an auxiliary encoder [110]. These implementations learn both the x and z joint probabilities.

In these approaches, the discriminator receives samples from the generator and the z latent space variable. To converge the discriminator correctly, the auxiliary encoder is responsible for encoding the original images into the z latent space [110].

2.3.1.4 Adversarial Autoencoders

Adversarial Autoencoders (AAE) rely on GANs for performing variational inference, a concept introduced by Mordido *et al.* [111]. AAE was then later proposed by Makhazni *et al.*, relying on Mordido *et al.* and Goodfellow *et al.* original GAN [112]. The authors proposed a probabilistic autoencoder that relies on a GAN for matching the resulting probability of the hidden encoded

vector with an input distribution [110]. The autoencoder in AAE is trained with the dual objective of optimizing the reconstruction error and an adversarial error.

Makhazni *et al.* claim that AAE can be used in applications that include semi-supervised classification, unsupervised clustering, and data visualization. They also claim that their approach achieved competitive results in the generative field and semi-supervised classification tasks [112].

2.3.2 Adversarial Attacks

In the image classification domain, adversarial examples constitute a set of samples that are intentionally synthesized and appear realistic but can mislead a given classifier to provide wrong prediction outputs, resulting in an adversarial attack [47]. With the increasing deployment of machine learning systems in the real world, preventing such attacks is essential. Some options include defending against these attacks and testing the worst-case robustness of models [113].

Adversarial examples have been studied with increasing interest since they were first noticed by Szegedy *et al.* [114]. The authors proposed that it is possible to transform an input image by a small amount in the image classification domain, leading the classifier model to change its prediction. The total amount of change on the input image was often so small that it was almost undetectable.

Dunn *et al.* introduced GANs as a novel way to generate unrestricted adversarial inputs [115]. The authors show that their method is adaptive, efficient, and scalable and successfully generates images for perturbation attacks against state-of-the-art classifiers. The authors highlight that their code base can easily be applied with other GAN architectures for ImageNet [116], proving that scalability can be accomplished. The adaptability arises from the fact that tuning the generator's weights can be accomplished to target a specific classifier. As per efficiency, the authors based this on the fact that once the generator is trained, it can generate adversarial images in a single forward pass [115].

Depending on the goal, an adversarial attack may cause a given classifier to misclassify a given input or cause a given classifier to misclassify a specific input from a class into another specific target class. This last technique has been referred to as source-target misclassification attack [117], or targeted attack [118].

There are several reasons why such adversarial examples can mislead a classifier. According to some authors, it can result from the high dimension of input space in comparison to model linearity [48]. Sensitivity on well-generalized features learned by the models [119] can also be one of the causes. At the same time, a higher quantity of prior data could be lacking for model generalization [120]. Additionally, the foundations for building robust systems to adversarial examples have been slow compared with the attack research field evolution [113].

2.4 Stress Testing

Smith-Miles and Bowlie suggest that testing an algorithm and its chosen parameters under all potential real-world conditions poses a significant challenge [121]. The authors also claim that

poor research practice in academia and deployment failures can be due to the lack of stress-testing of models. Stress-testing models involve creating and measuring conditions where the model may be less robust. This results in failures to understand model strengths and weaknesses. The authors advocate for the urgency in methods to generate many new test instances for a given problem, with controllable characteristics, to enable more rigorous testing of algorithms [121]. The proposed approach from Smith-Miles *et al.* is based on instance space visualization to identify where an algorithm is predicted to perform well based on previous empirical evidence. Machine learning methods can be used to define the boundary of good performance, known as the algorithm footprint [122].

In 2023, Cunha *et al.* proposed a variation of GANs that aim to generate realistic data for which a specific model performs poorly, classifying them with low confidence [5]. Cunha *et al.* theorized that these generated examples can be used to gain better knowledge of the frontier between classes in a dataset since the newly generated images with low confidence can describe a feature in model cards [123]. These model cards consist of a report documenting a model with information relevant to the user, such as the intended use case and evaluation metrics for different conditions [5]. They also claim that the approach can generate images close to these frontiers while maintaining realistic aspects. They named this new approach *GASTeN: Generative Adversarial Stress Test Networks* [5]. The summary of GASTeN architecture is shown in Figure 2.3.

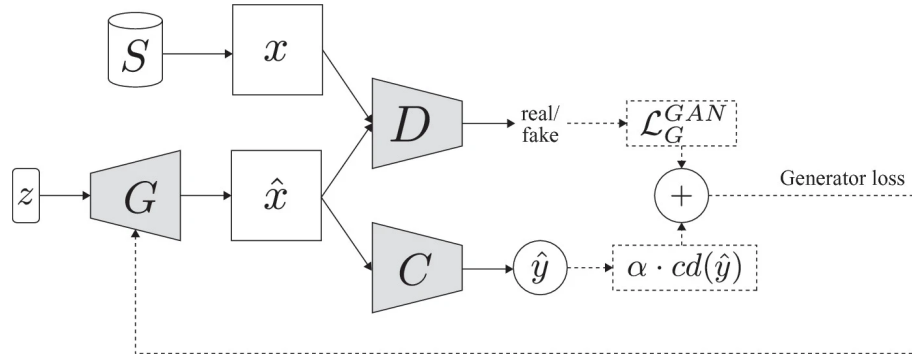


Figure 2.3: GASTeN Architecture [5]

GASTeN relies on a GAN framework to generate realistic samples. They extended this GAN architecture to include the target classifier (C) so that the classifier's output on the newly generated images is also used partly during the generator's optimization. This approach is similar to the one proposed by Dunn *et al.* back in 2019 [115]. GASTeN relies on DCGAN architecture and in the GAN loss proposed by Goodfellow *et al.* [100]. Additionally, a new confusion distance (cd) factor is introduced in the generator loss as shown in Equation 2.17.

$$\mathcal{L}_G^{GASTeN} = \mathcal{L}_G^{GAN} + \alpha \cdot cd(C(G(z))) \quad (2.17)$$

In the original GASTeN paper, α ranged from 0 to 30, and $|C(G(z)) - 0.5|$ defines cd [5]. GASTeN presents its experiments using MNIST and FashionMNIST benchmark datasets. The

authors use two different evaluation metrics to evaluate the results: FID for image quality and Average Confusion Distance (ACD). The authors also performed a visual inspection of the generated samples.

FID presents limitations that can have affected the quantitative evaluation in GASTeN. One of the limitations of FID is that the images used to pre-train the inception network have different characteristics from those present in the datasets used in GASTeN work. While ImageNET contains RGB images of isolated objects, MNIST dataset contains grayscale images of digits [5]. ACD presents its limitations as well. Using the average of the CD can be misleading in some scenarios where the spread of results is wide. The evaluation of the GASTeN results could benefit from additional metrics that characterize the distribution of results. The authors found it challenging to identify images that had both low ACD values and low FID scores [5]. Therefore, they concluded that it might not always be feasible to develop a generator that consistently deceives the target classifier while still producing images that are considered realistic. At the same time, human visual inspection of image quality can be biased [5].

GASTeN's work presents an important step in approaching the generation of low-confidence instances for model stress testing, but not without its limitations. GASTeN generates images that look both plausible and ambiguous. However, it does not always generate confusing images with satisfactory FIDs, and the GAN training frequently fails to converge [5]. Additionally, the results do not present an output distribution centered in 0.5 probability. This means that a post-processing selection phase is always necessary to filter images in the desired low-confidence region.

Model stress testing arises as an increasing necessity for researching model vulnerabilities and explainability.

Implementing machine learning implies building models from extensive training data and then using these models to perform predictions. The model is then applied to unseen test samples, hoping for satisfactory generalization. However, if generalization fails and an incorrect result is produced for a test input, it is often difficult to debug the cause of the failure [124]. These errors can occur because of several factors, such as incorrect algorithm implementation, incorrect parameters, or incorrect selection of features. In the same way, training data also affects the model's quality. This is reflected in a new class of debugging problems where ML classifiers are trained with data containing bugs, which are extremely hard to debug due to the high volume of instances [124]. Several efforts have been made in the past years to implement methods for model debugging [124, 125, 126, 127, 128, 129]. Similarly, several other studies have focused on assessing the data quality used for these models [130, 131, 132, 133].

The responsible AI field is proliferating due to the increased importance of AI usage [1]. Barredo *et al.* state that AI and ML have achieved a notable momentum that, if correctly used, can deliver improved solutions for many fields' problems [1]. Authors also claim that eXplainable AI (XAI) is widely acknowledged as a crucial feature for model deployment. The concept of model explainability arises within the context of model debugging [134]. Explainability provides explanations of the functioning of a model, ultimately finding the main factors that influence the predictions in general, for the whole dataset, exposing its vulnerabilities and biases [135]. From

the need for explainability and the determination of model vulnerabilities, model stress testing arises as an increasing necessity.

2.4.1 Ambiguous Images in Classification

The effectiveness of models depends on the availability of high-quality data for training and evaluation. However, in real-world scenarios, models may encounter ambiguous situations and data, making applying these technologies challenging. As pointed out by Aroyou and Paritosh in [136], the lack of ambiguous data in datasets used for machine learning is a major issue often ignored during testing and evaluation. As a consequence, Weiss *et al.*, claim that this lack of ambiguous test and validation data results in a lack of studies on true ambiguity by the scientific community [4].

There has yet to be a consensus on the definition of ambiguity to the present date. According to Weiss *et al.*, ambiguous images may represent an input for which multiple classes can be assigned correctly. One input may have a probability higher than zero for multiple classes simultaneously and with the same probability value [4]. According to Weiss *et al.*, a more formal definition of true ambiguity can be:

Given a data point x , x is truly ambiguous if and only if $P(c|x) > 0$ for more than one class c [4].

Weiss *et al.* identify true ambiguity as a special case of uncertainty disentanglement and reasoning following the previous work of Lines and Clements *et al.* [137, 138].

It is important to distinguish between the true ambiguous images defined above and images that can be defined as ambiguous to a certain model. The latter is related to an image positioned in that model's decision boundary. Decision boundaries may vary depending on the model and may not coincide with the true ambiguous definition [4, 139, 140]. Images generated by GASTeN are ambiguous for a single model since it explores the decision boundary of a specific CNN. Conversely, AMBIGUESS proposed by Weiss *et al.* aims to create an ambiguous dataset independent of the tested model and its supervisor [4].

Mukhoti *et al.* proposed, in 2021, the AmbiguousMNIST dataset containing samples with multiple plausible labels in addition to the standard MNIST dataset [141]. These multiple plausible labels lead to higher uncertainty in models.

Both AMBIGUESS and AmbiguousMNIST architectures relied upon VAEs for synthetic data generation. AMBIGUESS authors claim to outperform the AmbiguousMNIST dataset. Either AMBIGUESS, AmbiguousMNIST, or GASTeN performs a posterior selection in the generated samples. Meaning not all of the images generated by these methods are in the low-confidence region.

2.5 Summary

In this chapter, we describe the background behind the rise of generative AI, with a particular focus on GANs. Stress testing and ambiguity were also approached, along with adversarial attacks and probability models.

We started the chapter by describing the concepts for modelling probability (Section 2.1.1) and introducing discriminative and generative methods (Sections 2.1.2 and 2.1.3). In these sections, the tasks of image classification and image generation (Sections 2.1.2.1 and 2.1.3.1) were also defined and described. Then, common distance measures and loss functions were approached in Section 2.1.4. In Section 2.1.5, we approached the Multi-Objective Optimization problem. Section 2.1.6 highlighted some common evaluation metrics for the problem proposed in this work, such as inception score and Frechét inception distance.

Section 2.2 approached diversity and estimation of ambiguity. It also described the concepts of uncertainty and how are they related to ambiguity.

Moving forward, section 2.3 presents the important concepts regarding the GAN inner working and training. Several variations of its architectures are described in Sections 2.3.1.1, 2.3.1.2, 2.3.1.3, and 2.3.1.4. We introduced Adversarial attack concepts and related studies in Section 2.3.2 since this area is related to the problem being addressed in this work.

Finally, Section 2.4 presents the stress testing topic, highlighting its origin and pertinence in machine learning models and presenting relevant advancements in the area that can serve as a reference for the work that was developed. The topic of ambiguity in images described in Section 2.4.1 is an ongoing investigation that lacks a clear and consensual definition until the current date. This topic deals with both perception and machine learning model uncertainties.

Chapter 3

Methodology

This chapter presents the methodology that was used in this work. In Section 3.1, we present our problem statement. Section 3.2, details our proposed methodology for Hubris Benchmarking. We address the topic of universal ambiguity in Section 3.2.1. we propose a new metric for testing the overconfidence of models in Section 3.2.2. Section 3.3, details our architecture for the implementation done. Section 3.4 details the evaluation of RESNET-50 using our Hubris Benchmark. Finally, Section 3.5 details our experimental setup.

3.1 Problem Statement

Our goal is to stress-test machine learning binary classifiers by measuring their overconfidence in CV tasks. As stated in Section 2.4, we stress-test models when we create and measure conditions where the model may be less robust. In this case, we are stress-testing models against their overconfidence, that is when the model creates confident predictions for conditions where the model should be low-confident. Instances that should be predicted as low confidence by any model for a given dataset, are called universally ambiguous instances, as stated in Section 2.4.1.

To stress-test our model, we propose a framework dubbed *Hubris Benchmarking*. Given a model and the respective training dataset, we generate synthetic universally ambiguous instances, called the *companion dataset*. To generate our ambiguous companion dataset, we decided to adapt the GASTeN framework. GASTeN generates synthetic images close to the decision boundary of one classifier, that is, in the low-confidence region of the given classifier. Therefore, we adapt it to generate instances in the decision boundaries of a given set of diverse classifiers, intending to approach the generation of universally ambiguous images, as we assume that if an instance is challenging for many diverse models, it is generally challenging for most of the existing models. This GASTeN adaptation is dubbed *AmbiGAN*.

Finally, we propose a novel procedure to test a model’s overconfidence towards these challenging images. For this purpose, we developed a new metric to measure how overconfident a model is - dubbed *Hubris*.

The research on these topics is especially relevant for the application of AI in critical fields. As an example, consider the case of a cancer detection model using X-rays that is faced with an ambiguous X-ray image. If the model is certain that the X-ray does not indicate cancer presence, when in fact it is ambiguous, it means that the situation should have been reported to a doctor. However, an overconfident model will instead discard the X-ray as not having cancer. With this research, we can provide a new metric besides the performance estimation ones. This metric can be filled in model cards such that we can avoid situations that can harm humans.

3.2 Hubris Benchmarking

Our work generates a companion dataset for CV models given as input an original dataset for reference. The new companion dataset presents challenging samples that try both to be difficult to classify for a model and a human observer. The goal is to ensure that models, when confronted with these samples, should produce predictions with limited confidence or no clear prediction at all [142]. This is because the samples in the new dataset do not belong to any of the predefined classes in the original dataset.

The benchmark methodology that this work formulates involves the analysis of the predictions that a given model under stress test outputs for the new companion dataset. Since our challenging samples do not belong to any specific class, we expect the predictions of the model to have lower confidence than the original ones. The Hubris (as further explained in Section 3.2), along with the Average Confusion Distance metric [5], shall be calculated, and a comparison of results performed. The Hubris metric will measure the overconfidence of the model when classifying ambiguous samples. The Average Confusion Distance metric will quantify the distance between how far the predicted class is from the low confidence region of the classifier [5]. Figure 3.1 provides a visualization of the *Hubris Benchmarking* concept. The model under stress refers to the model that is being evaluated for overconfidence and is similar to the Model under test concept from AMBIGUESS [4].

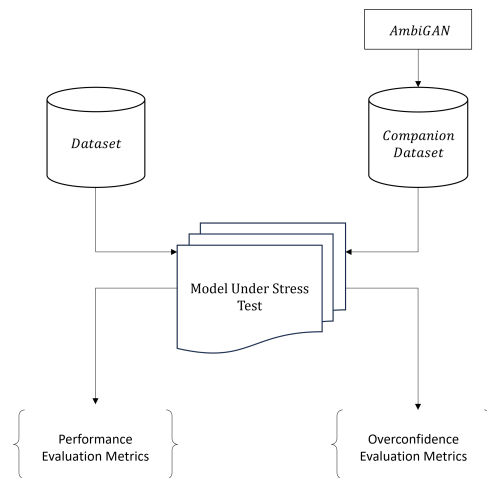


Figure 3.1: Hubris Benchmarking Methodology

3.2.1 Universal Ambiguity

As referred to in Section 2.4.1 from Chapter 2, there is no consensus on the definition of ambiguity. We define *universal ambiguity* when human observers and ML models find a given sample equally difficult to classify for a pre-defined set of available labels. For our research, we restrain the problem to binary classification. In this case, the ML model or a human observer should give equally probable predictions for both labels. This goal proves challenging due to the inherent differences in perception and processing between humans and algorithms. We aim to maximize and try to reach the Universe of Ambiguity by simplifying its constraints. Therefore, instead of targeting universal ambiguity, this work aims to maximize challenging sample generation by relying upon diversity and size within a collection of classifiers as described in Section 2.2. To ensure a wide range of perspectives and decision rules, we adopted a strategy based on the principle that diverse classifiers will establish different decision boundaries. We propose creating a set of diverse classifiers with varying architectures, each trained with sub-sets of the experiment data - similar to the bagging approach [87]. The diversity among classifiers is crucial because it allows us to find regions in the feature space where these classifiers disagree.

All in all, given a large number of classifiers with different boundaries, our research measures these distances to the boundaries and minimizes them for each iteration. Our resulting samples that lie in these boundaries are challenging since they contain features from both classes.

3.2.2 Hubris Proposal

Hubris ("Excessive pride, presumption or arrogance (originally toward the gods)"), is proposed in this research as a metric to estimate the overconfidence of a model towards challenging samples. It relies on the concept of KL-Divergence to measure distribution distance but is normalized and scaled towards our goal.

We derive from binary classification tasks, and we assume that our sets of samples are challenging to label. A model classifies these samples with a given probability $\hat{y}$. In the ideal case of an unbiased and calibrated model, the best case that captures characteristics equally from both classes will be a 0.50 probability output. Since in the perfect case, that is, no overconfidence, the model will predict our samples as having 0.50 for prediction value, we use a reference, *Ref*, as a Dirac distribution centered in 0.50. Additionally, for reasonable scaling purposes, we use a uniform distribution (*U*) ranging between $[0, 1]$. *U* represents the worst classifier case that predicts the samples under evaluation with random predictions. Hubris is defined as in Equation 3.1.

$$Hubris = 1 - e^{-\frac{KL(\hat{y}|Ref)}{KL(U|Ref)}} \quad (3.1)$$

Considering that Hubris means overconfidence, we take the KL Divergence (*KL*) between our ideal reference and the predictions and scale it into $[0, 1]$ using an inverse exponential scale. Finally, we invert the result so that we have 0 for no overconfidence and 1 for total overconfidence or arrogance. That is, the lower the Hubris, the more robust the model is for overconfidence.

We expect robust modules will not exhibit overconfidence towards any specific class when the input is ambiguous. Since we previously simplified our task to binary classification, we expect that the evaluated model classifies our samples as lying within the decision border.

3.3 AmbiGANs

In the previous Section 3.2, we designed a framework to measure Hubris. However, to measure it, we need to have low-confidence samples. Finding these images would be difficult because the available CV benchmark datasets contain few examples in this condition. The available low-confidence examples are not enough to evaluate the performance of models under stress tests. Therefore, we propose to build a companion dataset with synthetic images.

To generate our benchmark dataset, we developed a new approach dubbed *AmbiGANs* as in *Ambiguous Generative Adversarial Networks*. Starting from the GASTeN framework, as illustrated in Figure 2.3 from Section 2.4, we implemented several improvements to the original, namely:

- DCGAN Architecture changes.
- Multi-objective training loss function of the generator.
- Introduction of multiple Ambiguity Estimators.

Given that GANs are, per nature, already trying to achieve an equilibrium between real and fake samples, the generator is faced with an additional objective. This objective is to generate real samples that fool the discriminator and generate samples that confuse the classifier in its prediction. AmbiGANs solve this issue by training using a multi-objective loss and trying to balance sample realism and overall ambiguity. Figure 3.2 presents a summary of the proposed architecture. Different AmbiGAN variants are described in the following sub-sections.

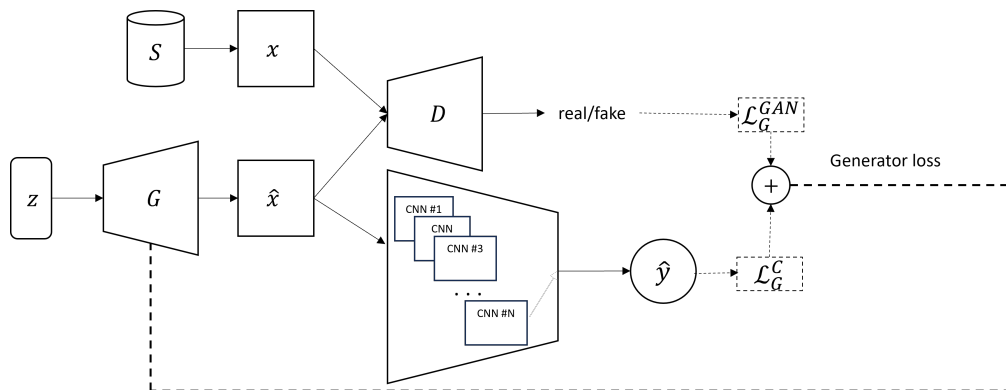


Figure 3.2: AmbiGANs Architecture

Similarly to GASTeN, AmbiGANs have the generator and discriminator trained in advance. This procedure improves the GAN stability throughout the rest of the training of the whole AmbiGAN structure. Also, AmbiGAN relies upon Ambiguity Estimator predictions for its optimization. Therefore, this estimator is trained beforehand and has all weights frozen during the AmbiGAN training process. Further details on the training process can be found in Section 3.5.3.

3.3.1 Generator Loss Function

AmbiGANs use the Gaussian Loss as an additional term for the generator loss. We calculate this loss considering the predictions given by the classifiers and the expected reference of 0.50 that we have previously defined.

Since having an objective of always trying to achieve $\hat{y} = 0.50$ can be too challenging and restrictive for the generator, the new loss tries to mitigate that risk. In fact, by using Gaussian Loss, we try to incentivize the generator to derive samples that result in a classifier’s prediction distribution as being Gaussian but centered in our target of 0.50.

We applied Gradient Clipping for both loss terms to prevent high dominance of Gaussian Loss over the original discriminator loss. For the discriminator loss term, we always used 0.50 as the maximum value of the L2-norm for all gradients of the generator. For the Gaussian loss term, the factor 0.50 is also used but multiplied by a hyper-parameter α , which we will use as a weight term in the experiments. We summarise this approach in Figure 3.3.

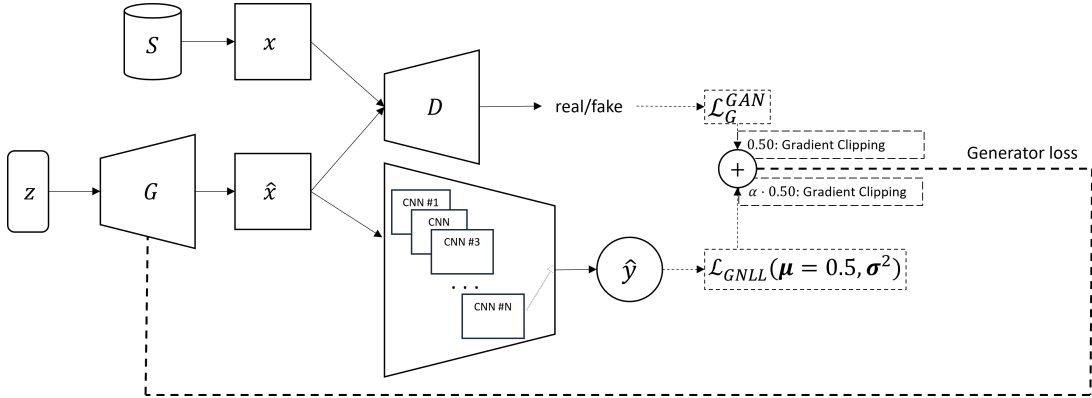


Figure 3.3: AmbiGANs with Gaussian Loss using Gradient Clipping.

Originally, in GANs, the generator was already faced with the Nash equilibrium against the discriminator. By relaxing the variance used in Gaussian loss, we can tailor our final samples as predicted by the classifiers, on average, as not belonging to any specific class. Gaussian loss will also smoothen the loss such that samples that are not within the desired distribution are largely penalized.

3.3.2 DCGAN Architecture Changes

When comparing the GASTeN implementation [5] with the original DCGAN [40], our analysis revealed two discrepancies: the authors replaced the initial convolutional layer in the generator

with a linear layer, and similarly, the last convolutional layer in the discriminator with a linear layer. Our first measure was to revert these changes to preserve the architecture of the original DCGAN.

AmbiGANs are based on the original DCGAN architecture. Only convolutional and transposed convolutional layers (also known as fractionally-strided convolution) are used within the discriminator and generator.

The complexity of the discriminator was increased to have twice as many features as the generator. This was done to enlarge the robustness of the discriminator against potential adversarial attacks that the generator may create when multi-objective training is being performed.

Table 3.1: Generator Model Summary

Layer	Output Shape	#Params
Latent Space Input	[128, 1, 1]	–
ConvTranspose2d	[128, 7, 7]	401,408
BatchNorm2d	[128, 7, 7]	256
ReLU	[128, 7, 7]	–
ConvTranspose2d	[64, 14, 14]	131,072
BatchNorm2d	[64, 14, 14]	128
ReLU	[64, 14, 14]	–
ConvTranspose2d	[1, 28, 28]	1,025
Tanh	[1, 28, 28]	–

Table 3.2: Discriminator Model Summary

Layer	Output Shape	#Params
Image Input	[1, 28, 28]	–
Conv2d	[128, 14, 14]	2,176
LeakyReLU	[128, 14, 14]	–
Conv2d	[256, 7, 7]	524,288
BatchNorm2d	[256, 7, 7]	512
LeakyReLU	[256, 7, 7]	–
Conv2d	[1, 1, 1]	12,544
Flatten	[1]	–
Sigmoid	[1]	–

The number of times the discriminator was trained regarding the generator was five times superior. This was performed to achieve better stabilization, meaning that the discriminator is optimized five times for each batch of samples, while the generator is optimized one time. The original GASTeN the discriminator was optimized the same number of times as the generator. This is an improvement that helps the discriminator to be more robust in detecting fake samples. It prevents mode collapse and promotes image diversity.

A detailed description of the generator’s and discriminator’s layer structures can be found in Tables 3.1 and 3.2 respectively.

3.3.3 Ambiguity Estimators

This section expands on the types of estimators used for training AmbiGANs during the multi-objective mode. In this mode, the GAN tries to balance the realism and ambiguity of samples. We assume that the multiple classifiers provide an estimation of universal ambiguity.

The classifiers are trained in an initial step (before GAN training) with a subset of the training data to act as our Ambiguity Estimator. These subsets of training data were not overlapped. This resulted in each classifier being trained with, in fact, different samples and learning different sets of features. This step is crucial for establishing the baseline performance of the classifiers before integrating them into the AmbiGAN framework.

Depending on the type of Estimator, this training step can have either one or two phases. The next sub-sections provide the details on these phases.

All classifiers used in the estimators were based on a CNN architecture, but we assigned each layer a random number of features between 1 and 5. We also assigned each classifier a random number of layers between 1 to 4. At the end of the convolutional layers, a fully connected layer is responsible for flattening the input for final binary classification with the sigmoid activation function. Since we randomly assigned the number of layers and features for each CNN, the set resulted in different but precise models. A practical example with 2 layers, having 1 and 5 features for each layer respectively, can be found in Table 3.3. The different approaches to gathering the outputs of the individual classifiers are discussed in the following subsections.

Since each classifier will have different configurations in terms of features and layers and is trained with different subsets of the dataset we assume they will have different decision boundaries. If we have more classifiers we expect we will have a higher number of different decision boundaries. This is our approach to tackling universal ambiguity. We chose to use 50 classifiers to maximize the available computational resources.

Table 3.3: Example Classifier Model Summary

Layer	Output Shape
Image Input	[1, 28, 28]
Conv2d	[1, 28, 28]
ReLU	[1, 28, 28]
MaxPool2d	[1, 14, 14]
Conv2d	[5, 14, 14]
ReLU	[5, 14, 14]
MaxPool2d	[5, 7, 7]
Flatten	[245]
Linear	[1]
Sigmoid	[1]

The estimators combine the trained classifiers such that AmbiGANs use the predictions to optimize the generator towards ambiguity. The output of these estimators is then used in the Generator Loss calculation.

3.3.3.1 Mean Output Combination

This type of estimator combines all classifier outputs and averages the result into a single prediction. We decided to follow this approach because it is the simplest approach possible. Figure 3.4 summarizes the approach.

This combination is expected to maintain the diversity since each classifier will have the same weight in the final prediction.

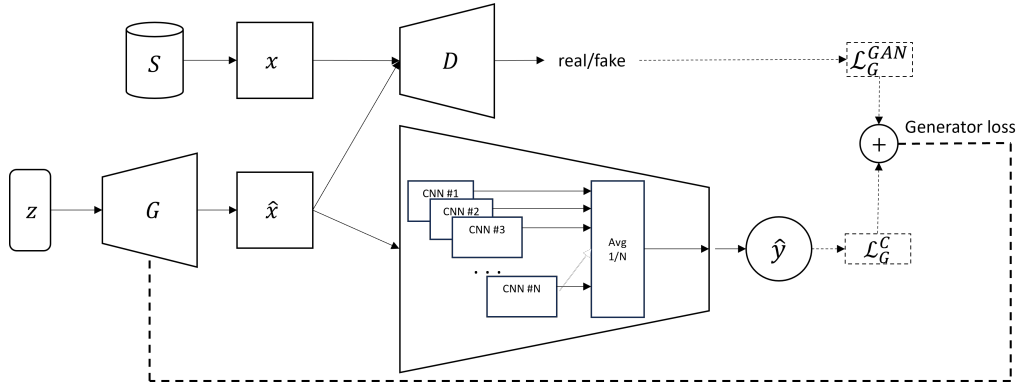


Figure 3.4: AmbiGANs with Mean Output Estimator

3.3.3.2 Linear Output Combination

This type of estimator combines all classifier outputs using a linear layer trained to achieve the best weight between each classifier. The output of this layer is passed through a sigmoid activation, resulting in a single prediction ranging between $[0, 1]$. A representation of this approach can be found in Figure 3.5.

For the training step of this type of estimator, in the first phase, the individual classifiers were trained, and their weights were frozen. In this case, a subset of the initial trainset that was used to train the classifier, and the GAN was held out to train the linear layer on a second phase. After this, the weights of the linear layer are frozen and not changed during AmbiGAN training.

It is expected that this type of combination will tend to give more importance to individual classifiers that have better accuracy. Hence, we may lose the diversity within the overall predictions.

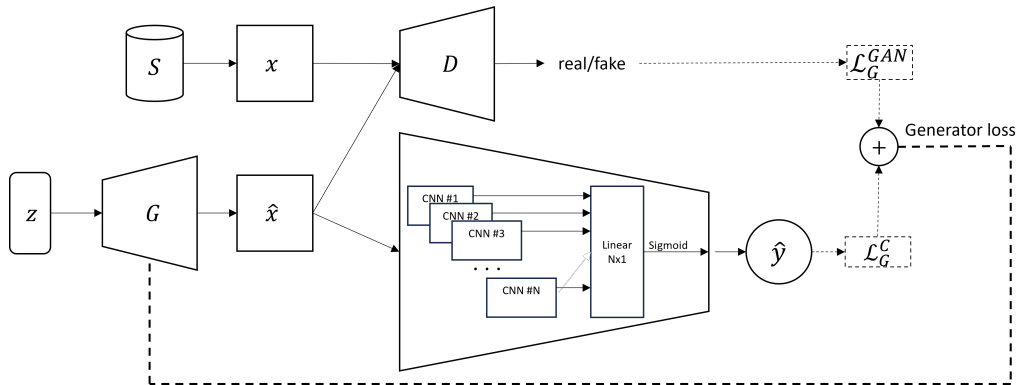


Figure 3.5: AmbiGANs with Linear Output Estimator

3.3.3.3 Meta-Learner Output Combination

The meta-learner approach lets a Neural Network with three linear layers learn the best output combination. The hidden layers used ReLU activation, and the final output passed a sigmoid

activation, resulting in a single prediction ranging between $[0, 1]$. This approach is represented in Figure 3.6.

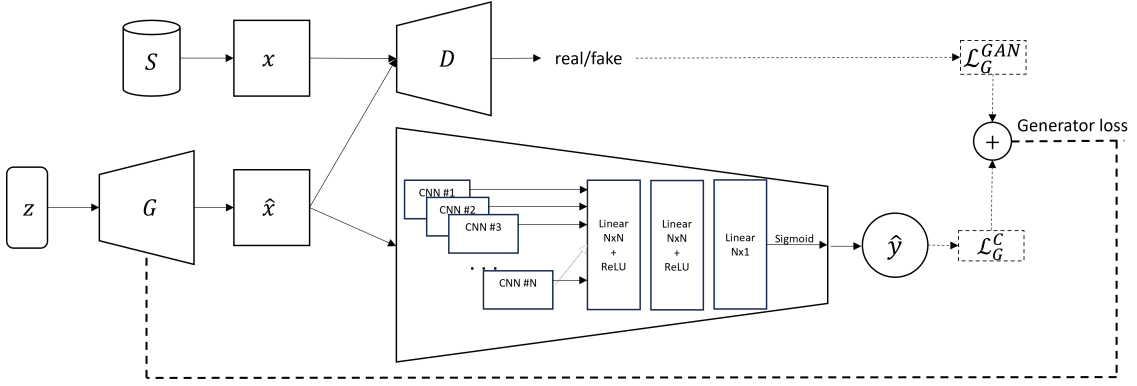


Figure 3.6: AmbiGANs with Meta-Learner Output Estimator

For the training step of this type of estimator, in the first phase, the individual classifiers were trained, and their weights were frozen. In this case, a subset of the initial trainset that was used to train the classifier, and the GAN was held out to train the Meta-Learner layer on a second phase. After this, the weights of the Meta-Learner layer are frozen and not changed during AmbiGAN training.

Similar to the Linear Combination case, this combination is expected to result in better prediction scores but less diversity.

3.3.3.4 No Output Combination with Gaussian Loss

The problem of reduced diversity in the variants using a combination of outputs can be avoided. If we do not combine any classifier and use the raw predictions from each classifier in the AmbiGANs training process, we expect diversity to be preserved.

The *no output combination* results in a list of predictions used to compute the ambiguity estimator loss. The losses from each classifier are summed and weighted for the final generator loss as in Equation 3.2. Notice that no gradient clipping was used in this approach. The loss terms were just summed and weighted together. The visual representation of this approach can be found in Figure 3.7.

$$L_G^{AmbiGAN} = L_G^{GAN} + \alpha \cdot \sum_{i=1}^N L_i^{GNLL} \quad (3.2)$$

3.3.3.5 No Output Combination with KL Loss

The last *no output combination* used the KL-Divergence loss. In this case, the list of predictions was directly used when computing the ambiguity estimator loss, as represented in Equation 3.3.

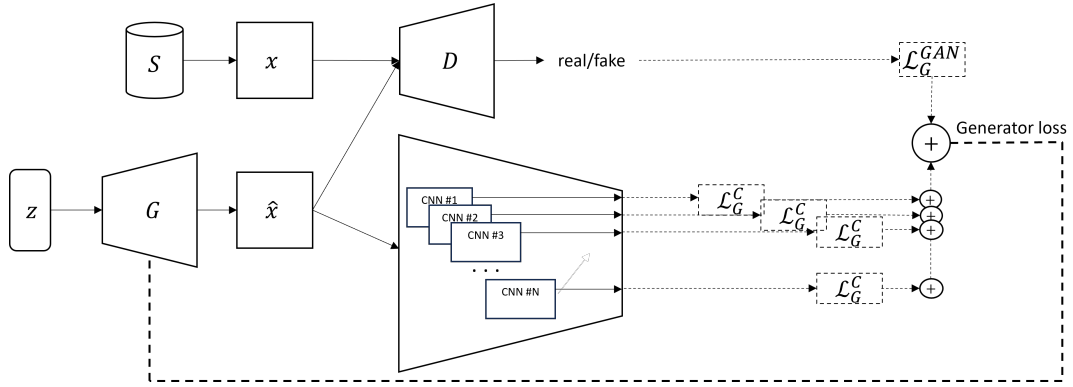


Figure 3.7: AmbiGANs with No Output Combination Estimator

Notice that no gradient clipping was used in this approach. The loss terms were just summed and weighted together.

$$L_G^{AmbiGAN} = L_G^{GAN} + \alpha \cdot \sum_{i=1}^N L_i^{KL} \quad (3.3)$$

3.3.4 Evaluation strategy of AmbiGAN

In measuring our performance, we employ the same metrics as Cunha *et al.*, facilitating direct comparison of results. We use accuracy to assess classification performance. Then, to evaluate GASTeN's and AmbiGAN's performance, we use FID and ACD scores. The FID score measures the realism of generated images, while the ACD score assesses the generator's capacity to produce images with low classification confidence.

Both ACD and FID scores should be minimized. However, since our primary goal is to generate images near the decision boundary, we prioritize the ACD metric, provided the FID scores remain within a reasonable range.

Visualizing relative frequencies of probability outputs for the generated images through histogram plotting provides insights into the shape of the distribution and its magnitude. Boxplots are used to characterize the resulting distributions further.

Additionally, we visually inspect the generated samples to provide a qualitative assessment. This inspection facilitates the rapid identification of collapsed GANs and allows us to evaluate the ambiguity of the images.

3.4 Hubris evaluation of RESNET-50

We aim to test our framework using an independent model that was not developed by us. We decided to test a pre-trained model that was well-known and established. RESNET-50 has demonstrated strong performance across diverse image classification tasks. Testing it could provide knowledge into how a versatile model handles overconfidence issues.

We used *Hubris* metric along with the original *Average Confusion Distance* metric from GAS-TeN’s previous work [5] for the test. We followed the previously formulated benchmarking setup that predicted the class of our examples when faced with a pre-trained ML model classifier.

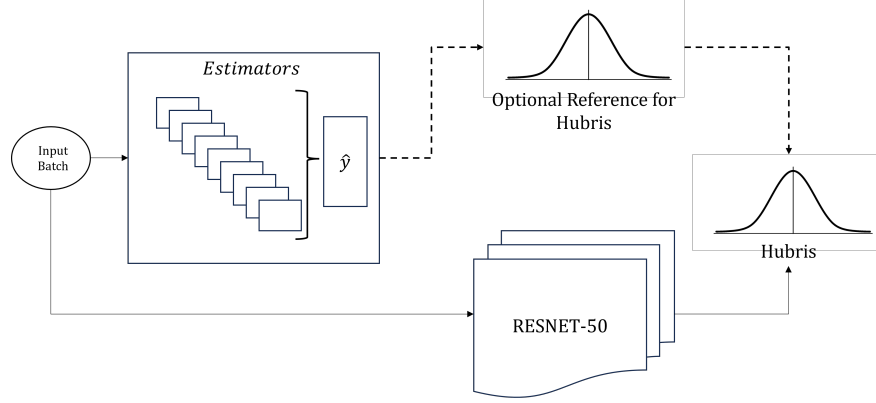


Figure 3.8: RESNET-50 Hubris Evaluation Methodology

Absolute Hubris ($Hubris_A$) along with the ACD metric, was calculated for results comparison. Absolute Hubris means Hubris in reference to the ideal ambiguous image output probability distribution. We also measured Relative Hubris ($Hubris_R$), taking as a reference the predictions given by a group of Classifiers as Ambiguity Estimators. A scheme of this procedure is summarized in Figure 3.8. FID was also measured for reference.

3.5 Experimental Setup

3.5.1 Datasets

All experiments were conducted using the MNIST dataset. This dataset contains a collection of labeled handwritten digits [143]. Having 60,000 samples for training and 10,000 samples for testing, this dataset is very well known for common CV model benchmarking. The samples are all 28x28 pixels wide and possess only one color channel (grayscale), which results in faster training times than using RGB colourspace. We chose this dataset because it is interpretable without needing expert knowledge. At the same time, since the original GASTEEN work was also initially trained with the MNIST dataset, we kept this choice for comparison consistency.

All the experiments were conducted with a two-class subset from the original dataset, as was also done in the previous work. Having a two-class subset can be simplified as a binary classification task. This means that for each experiment, there is a negative class and a positive class that, in this case, represents two different numbers from the MNIST dataset. Selecting, for example, the number 7 for the positive class and the number 1 for the negative class results in a binary classification task. In the remainder of this work, we named the experiments as PvN , where P is the positive class, and N is the negative class. For the previous example, the experiment name would be $7v1$. For this research, we conducted all experiments on $7v1$ and $9v8$ subsets.

For the training of the ambiguity estimators (Table 3.5 - Step #1 and #2), we filter the MNIST dataset by two classes for each experiment. That results in 12,000 samples for training and validation and 2,000 samples for testing in each experiment. The validation set uses 1/6 of the training set. As per the training of GAN (Table 3.5 - Step #3 and #4), the full 12,000 training samples are used.

Besides its simplicity and popularity, selecting the MNIST dataset allowed the comparison between the new benchmark dataset generated with this work with an already available one that also contains challenging samples: AmbiguousMNIST [141]. To our knowledge, AmbiguousMNIST is the only publicly available dataset with these challenging examples.

3.5.2 Configurations and Hyperparameters

Table 3.4 presents the different configurations and hyperparameters tested in our experiments and the schematic representation can be found in Appendix B.

For the Mean, Linear, and Meta-Learner Output Estimator types, the weight represents the value of a multiplying factor for the Gradient Clipping operation, as previously described in Figure 3.3. For the No Output Combination Estimator types, the weight represents the multiplying factor for the sum of the individual classifier losses as described earlier in Figure 3.7.

CD represents the original Confusion Distance loss from GASTeN. For the different values of CD, we considered the original weights from GASTeN. We added the weight of 50 to explore giving higher importance to this loss term.

Gaussian loss requires specifying a variance (σ^2). We experimented with variances of 0.01 and 0.005. A variance of 0.01 corresponds to a standard deviation of 0.1, implying that 68.2% of the values will fall between classifier probabilities of 0.4 and 0.6. A variance of 0.005 leads to a standard deviation of 0.07, meaning that 95.4% will lie between 0.64 and 0.36. We selected these variance values to explore different tolerances on the estimator confidence. A higher variance allows greater tolerance and a lower variance leads to a closer concentration of images around the decision boundary.

The values for the weights for Gaussian and KL div losses were tested and chosen manually. For the Mean, Linear, and Meta-Learner Estimator types, only weight 1 was experimented with - resulting in a maximum gradient of 0.5. In these types of estimators, we are combining the outputs into a single output, and therefore we have a single loss resulting from these estimators. The magnitude of the gradients of this loss is comparable with the discriminator gradients. We know this because in initial experiments we manually inspected the gradient values of the discriminator during training. With this value for weight, we can achieve a balance between the two loss terms, without one dominating the other. After this, the parameter that can fine-tune the output probability of our results is the variance. Because of this, we focused on varying this parameter.

For the No Output Combination Estimator types, several values for the weight were tested, including the variants with Gaussian Loss. This is because with this configuration the weight factor has a higher impact. The No Output Combination sums all individual classifier losses which has a much larger magnitude when compared to a single discriminator loss. We need to balance

this proportion giving low weight values for this variant. Otherwise, the estimator’s loss would prevail over the discriminator, resulting in non-realistic images. These concepts are in line with multi-objective training described in Section 2.1.5. For the No Output Combination with Gaussian Loss weights experiments ranged from 0.005 to 0.03. For No Output Combination with KL div loss, we varied the weights between 0.25 and 0.75.

Table 3.4: Different Configurations and Hyperparameters

Architecture	Ambiguity Estimator	Loss	Weight	Variance
GASTeN	Mean, Linear, Meta-Learner	CD	1, 10, 25, 50	N/A
GASTeN	Mean, Linear, Meta-Learner	Gaussian	1	0.01, 0.02, 0.03, 0.05
GASTeN	No Output Combination	Gaussian	0.005, 0.015, 0.03	0.001, 0.002, 0.005
GASTeN	No Output Combination	KL div	0.25, 0.50, 0.75	N/A
AmbiGAN	Mean, Linear, Meta-Learner	CD	1, 10, 25, 50	N/A
AmbiGAN	Mean, Linear, Meta-Learner	Gaussian	1	0.01, 0.02, 0.03, 0.05
AmbiGAN	No Output Combination	Gaussian	0.005, 0.015, 0.03	0.001, 0.002, 0.005
AmbiGAN	No Output Combination	KL div	0.25, 0.50, 0.75	N/A

Although all the variants in Table 3.4 we tested, we structured our experimental approach into three phases for comparison. Initially, we assess whether the updated architecture improves the previous one –GASTeN; then, we select the best architecture to evaluate the efficacy of the new loss function; finally, we evaluate different output combinations for our Ambiguity Estimators.

We consider the Mean Output Combination variant as our baseline since it is the simplest approach. This means that this configuration is used throughout our comparisons as a starting point.

When we refer to *GASTeN architecture* we consider the original GAN architecture used in GASTeN. The GASTeN classifier was modified to use our Ambiguity Estimators. This decision was made so that the results were comparable. All the tables and figures in this document mentioning *GASTeN* refer to this configuration.

The results presented for ACD and FID are the results for the last epoch.

3.5.3 Training Process

The training follows several steps designed to incrementally build the blocks necessary for generating ambiguous images. The steps are summarized in Table 3.5.

In the first step, we trained 50 individual classifiers for 50 epochs with different subsets of the data as described in Section 3.3.3.

An additional step involves training a combiner for the classifiers’ outputs required for Linear and Meta-Learner combinations. This requires an additional subset of training data to be held out

Table 3.5: Training Steps Summary

Step	Description
# 1	Train Classifiers for 50 epochs
# 2	For Linear and Meta-Learner: Train Classifiers output combiner for 50 epochs
# 3	Train GAN for 20 epochs
# 4	GASTeN/AmbiGAN training for 40 epochs

and used only after completing the first step. This optional step is also trained 50 epochs and aims to optimize the combined output of multiple classifiers.

Next, the generator and discriminator are trained for 20 epochs. Finally, the AmbiGAN is trained for 40 epochs. This phase integrates the components trained in previous steps and fine-tunes the overall structure to achieve the desired performance and accuracy.

3.5.4 Implementation Details

All experiments were conducted using a machine equipped with one Tesla T4 GPU, ensuring sufficient computational power for training and inference phases.

All code was developed using Python3. The base code from GASTeN was retrieved from GitHub. A new repository was created by cloning the original repository and maintaining the git history. Modifications and improvements are also tracked within the new repository in GitHub. The artifacts generated during the execution were stored in the computer filesystem, and metrics were logged to the Weights and Biases online platform. All metadata required for the reproducibility of experiments is stored through this platform.

Several experiments with different hyperparameter combinations were performed for all variants described in Section 3.3. The PyTorch framework was used for Neural Network training as in the original source code.

For the Hubris evaluation of RESNET-50, the pre-trained model from the TorchVision Python package was tested. As performed in all experiments, the datasets were filtered to contain a positive and negative class, turning the task into a binary classification problem. Since the original RESNET-50 model was trained for ImageNet and expected RGB images, we adapted the first convolutional layer for our use case (greyscale, 28x28 images). The final fully connected layer was replaced with a linear layer and a sigmoid output. This resulted in having a RESNET-50 model with a binary prediction. The model was then retrained with the original training sub-set from MNIST. Results were evaluated with MNIST, our generated samples, and AmbiguousMNIST test sets filtered by the experiment classes.

The ideal ambiguous image reference distribution for the Absolute Hubris calculation is an output probability distribution of mean 0.5 and variance 0.

For the relative Hubris calculations and to prevent data leakage, we used a different set of Ambiguity Estimators. This means we used a different seed from the one used in AmbiGAN

training to generate random numbers of features and layers. These estimators were trained for 100 epochs (double the training performed for the AmbiGAN estimators). The accuracy of the Ambiguity Estimators can be consulted in Appendix C.

3.5.5 Visualizations

Visualizations were performed with matplotlib, seaborn and scienceplots libraries. The results were retrieved from Weights and Biases and the local filesystem for plotting.

All histogram visualizations throughout this document are plotted with 20 bins. The x-axes always vary in the same ranges (0 to 1), meaning bin width is the same for all comparisons (0.05). We employ histograms with normalized y-axis values to ensure unbiased comparisons across the different sets of experiments. Each histogram’s y-axis represents relative frequencies, calculated by dividing each bin’s observations count by the total number of samples (2048 in all cases). In the legend of these visualizations, we also present the average FID score for the experiment to provide additional information and facilitate comparison.

In all boxplot visualizations, the mean is also represented in these visualizations by the dashed line. The average FID for each variation is also presented on the y-axis to give more information for easier comparison.

3.6 Summary

This chapter explains the methodology followed in this work. Section 3.1 defines our problem statement. In Section 3.2, we formulate a benchmark for analyzing model overconfidence when faced with challenging images. In 3.2.1, we approach the Universal Ambiguity and formulate an attempt to simplify its constraints. Hubris metric is proposed in 3.2.2 to quantify the level of overconfidence exhibited by a model.

Section 3.3 describes the new AmbiGANs framework used to generate the challenging images. Gaussian loss usage and DCGAN architecture changes are further described in 3.3.1 and 3.3.2, respectively. In 3.3.3, we describe the Ambiguity Estimators that were developed for this work along with the different types of output combinations that were experimented with mean, linear, meta-learner, and no-combination. The evaluation performed on RESNET-50 using Hubris is further described in Section 3.4, using the previously formulated benchmark setup as a reference.

The experimental setup was described in 3.5, where specific implementation details were provided. The training process is introduced in 3.5.3.

Chapter 4

Results

This chapter presents the results obtained from the experiments conducted. In Section 4.1 we present AmbiGAN evaluation results. Section 4.1.1 presents the AmbiGANs architecture results compared to the previous GASTeN. Section 4.1.2 presents the results regarding using Gaussian Loss for the multi-objective training. Section 4.1.3 includes the results for the different ambiguity estimators developed. Finally, Section 4.2 concludes with the Hubris evaluation results for RESNET-50. The full list of experiments and results, including the accuracy of the estimators is available in Appendix B, C, D, E, F, G and H.

4.1 AmbiGAN Evaluation

4.1.1 DCGAN Architecture Changes

Table 4.1, and 4.2, show the results when comparing the AmbiGANs architecture against the previous work — GASTeN. These initial experiments were conducted using the Confusion Distance loss.

Table 4.1: DCGAN architecture changes comparison for 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Mean	Confusion Distance	1	N/A	0.38	15.04
GASTeN	Mean	Confusion Distance	10	N/A	0.38	11.97
GASTeN	Mean	Confusion Distance	25	N/A	0.38	12.63
GASTeN	Mean	Confusion Distance	50	N/A	0.37	16.74
AmbiGAN	Mean	Confusion Distance	1	N/A	0.37	22.45
AmbiGAN	Mean	Confusion Distance	10	N/A	0.37	20.88
AmbiGAN	Mean	Confusion Distance	25	N/A	0.33	22.48
AmbiGAN	Mean	Confusion Distance	50	N/A	0.06	58.76

We present the results in Table 4.1 for the 7v1 experiment. The GASTeN architecture shows relatively stable FID scores across different loss weights. The ACD score remains almost constant, indicating that the architecture is less sensitive to changes in loss weight.

The AmbiGAN architecture exhibits higher FID scores than GASTeN, indicating a trade-off where reducing ACD results in lower image quality. As the loss weight increases, the ACD score decreases significantly. However, this comes at the cost of higher FID scores, with the highest FID (58.76) at a loss weight of 50.

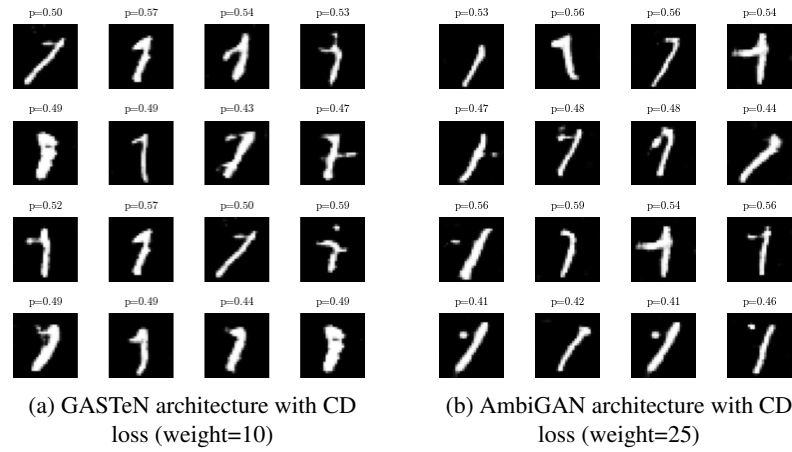


Figure 4.1: Examples of images of 7v1 experiment for GASTeN and AmbiGAN architectures

Some examples of the generated images for the 7v1 experiment are presented in Figure 4.1. By comparing the two sets of examples we notice the images from GASTeN present more noise than AmbiGAN. We can see that some images have features from both classes (1 and 7). This is more evident in images from AmbiGAN architecture. This is supported by the lower ACD score achieved with AmbiGAN, showing that the new architecture effectively increases ambiguity.

Table 4.2: DCGAN architecture changes comparison for 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Mean	Confusion Distance	1	N/A	0.31	11.85
GASTeN	Mean	Confusion Distance	10	N/A	0.31	10.76
GASTeN	Mean	Confusion Distance	25	N/A	0.30	17.02
GASTeN	Mean	Confusion Distance	50	N/A	0.01	188.79
AmbiGAN	Mean	Confusion Distance	1	N/A	0.27	21.41
AmbiGAN	Mean	Confusion Distance	10	N/A	0.21	21.91
AmbiGAN	Mean	Confusion Distance	25	N/A	0.08	24.04
AmbiGAN	Mean	Confusion Distance	50	N/A	0.06	38.78

In Table 4.2, we have the results for the 9v8 experiment. In this case, it is even more evident that AmbiGAN architecture can achieve lower ACD values than GASTeN architecture. While

GASTeN maintains a balance between ACD and FID at lower loss weights, its performance deteriorates at higher weights, leading to unusable images. It is noticeable as well that for one of the cases (9v8 experiment, loss weight 50), the original GASTeN collapsed, turning the FID score significantly higher (>100). In contrast, AmbiGAN consistently achieves lower ACD values across increasing loss weights, with a slight decline in image quality, highlighting its effectiveness in reducing ACD.

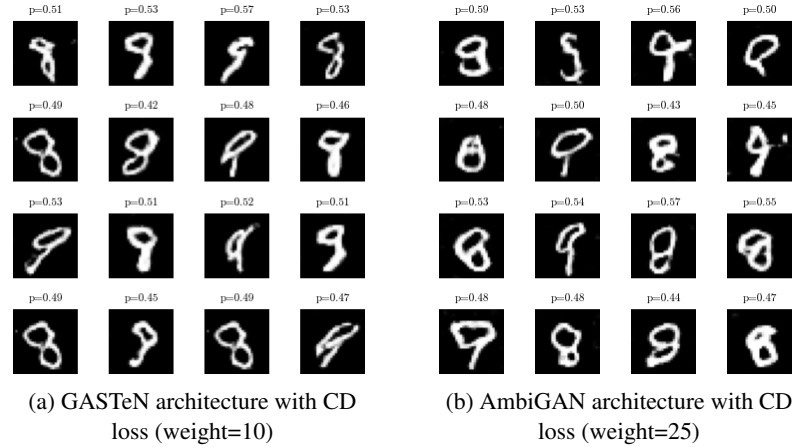


Figure 4.2: Examples of images of 9v8 experiment for GASTeN and AmbiGAN architectures

In Figure 4.2, we can see some examples of the 9v8 experiment. Also, in this case, we can say the images appear realistic, as in the 7v1 experiment.

Globally, FID scores in the new architecture increased to a certain degree, although at still acceptable values. On the other hand, we can see that this architectural change makes the GAN more permissive regarding the additional loss factor. We can confirm this by looking at the ACD scores. There is a noticeable reduction in the ACD score for the AmbiGAN architecture as the loss weight increases. This indicates that the new architecture is better at resolving ambiguities as the weight loss increases.

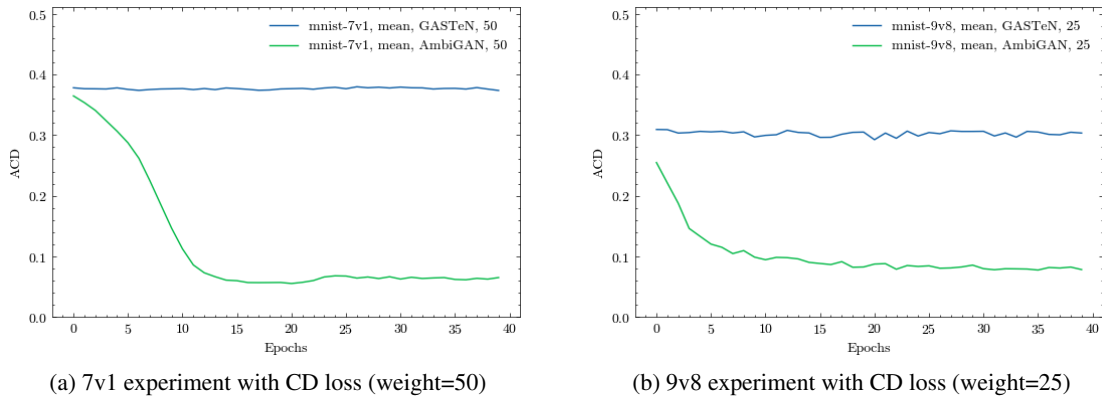


Figure 4.3: Evolution of ACD during training for GASTeN and AmbiGAN architectures

The comparison of ACD evolution through training between GASTeN architecture and AmbiGAN architecture (loss weight=50 and 25) is compared in Figure 4.3. The reduced ACD scores in the AmbiGAN architecture indicate its potential for more effective multi-objective training. The entire comparison of ACD values during training for all experiments can be found in Appendix E. The results suggest a potential trade-off between the ambiguity of the resulting images (as measured by the ACD score) and the quality of the generated images (as measured by the FID score). We can conclude that this change improves GAN stability as advised in [40]. At the same time, it improves permissivity toward our goal of targeting multi-objective training. For this reason, we consider these architectural changes the most advantageous.

4.1.2 Generator Loss Function

From the previous experiment, we selected AmbiGANs as the best approach. In this test, we will compare different losses. Table 4.3, and 4.4 presents the usage of Gaussian loss against the Confusion Distance loss. These experiments were conducted with the architectural changes mentioned in Section 4.1.1.

Table 4.3: Gaussian Loss comparison for 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
AmbiGAN	Mean	Confusion Distance	1	N/A	0.37	22.45
AmbiGAN	Mean	Confusion Distance	10	N/A	0.37	20.88
AmbiGAN	Mean	Confusion Distance	25	N/A	0.33	22.48
AmbiGAN	Mean	Confusion Distance	50	N/A	0.06	58.76
AmbiGAN	Mean	Gaussian	1.00	0.01	0.08	51.51
AmbiGAN	Mean	Gaussian	1.00	0.02	0.08	36.67
AmbiGAN	Mean	Gaussian	1.00	0.03	0.08	42.20
AmbiGAN	Mean	Gaussian	1.00	0.05	0.08	44.67

In Table 4.3, we notice an increase in the FID score for Gaussian loss compared to Confusion Distance loss. However, the ACD score decreases for values that Confusion Distance loss only achieves when using a significant weight value (50) at the expense of a higher FID (58.76).

Besides achieving better ACD scores, the Gaussian Loss offers a significant advantage: it allows tuning the variance hyperparameter, enabling adjustable tolerance for generating images around the mean of 0.50 probability output.

We can see that the results of using Gaussian loss lead to a higher number of samples classified as 0.50 by the estimator. This can be observed in the histogram of Figure 4.4. This visualization represents the number of images by classification probability output. For Gaussian loss, this distribution is centered in the half-scale of the prediction range. This is an advantage since it improves performance by generating more images in our region of interest. At the same time, it avoids the effort of performing posterior selection of the resulting images.

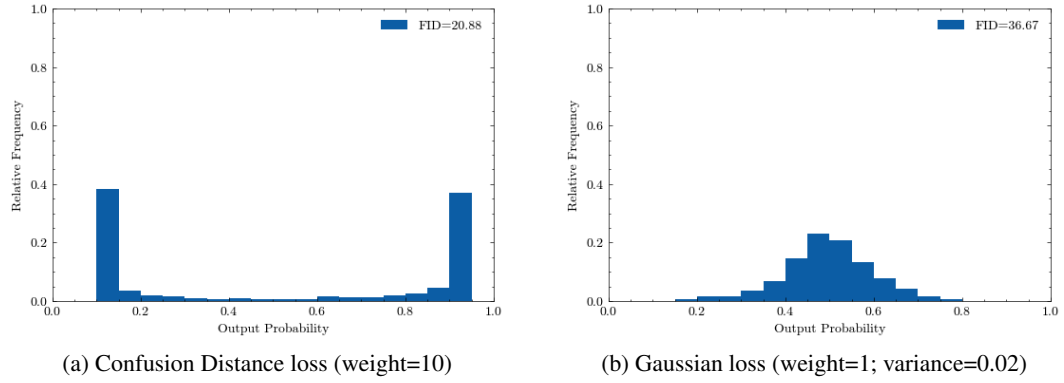


Figure 4.4: Histogram of results for 7v1 for CD and Gaussian losses with lower FID score

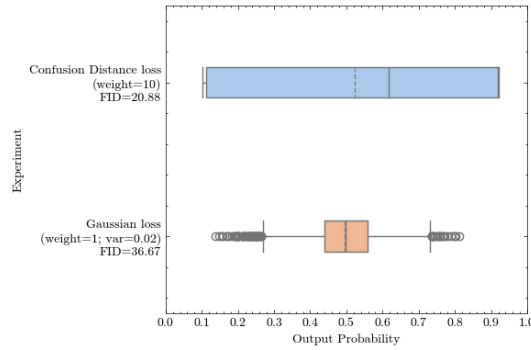


Figure 4.5: Boxplot of results for 7v1 for CD and Gaussian losses with lower FID score

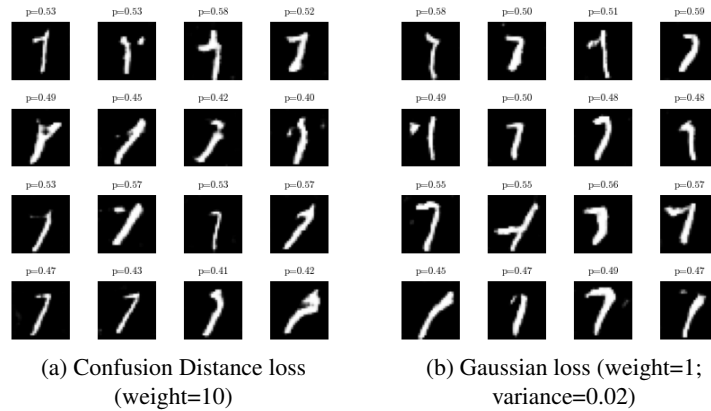


Figure 4.6: Examples of images of 7v1 experiment for CD and Gaussian loss

The boxplot in Figure 4.5 confirms these observations. For Gaussian loss, both the median and mean are centred at 0.5. CD loss presents some left skewness and a wider variability and spread in the output probability results — approximately between 0.10 and 0.90. In contrast, Gaussian loss presents less variability and spread — approximately between 0.30 and 0.70. However, 50% of the data is within approximately 0.45 and 0.55 output probability. The two halves of the box from Gaussian loss are symmetric, which suggests that the results are not skewed. Some outliers are

present in Gaussian loss results below Q1 and above Q3. Even taking these outliers into account, Gaussian loss presents a lower spread when compared to CD loss. The boxplots for the full results can be found in Appendix G. We can find examples of generated images for the 7v1 experiment in Figure 4.6. The both sets of images present realistic examples with low noise. We can say that both of them are also ambiguous.

Table 4.4: Gaussian Loss comparison for 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
AmbiGAN	Mean	Confusion Distance	1	N/A	0.27	21.41
AmbiGAN	Mean	Confusion Distance	10	N/A	0.21	21.91
AmbiGAN	Mean	Confusion Distance	25	N/A	0.08	24.04
AmbiGAN	Mean	Confusion Distance	50	N/A	0.06	38.78
AmbiGAN	Mean	Gaussian	1.00	0.01	0.05	24.08
AmbiGAN	Mean	Gaussian	1.00	0.02	0.05	24.79
AmbiGAN	Mean	Gaussian	1.00	0.03	0.05	22.87
AmbiGAN	Mean	Gaussian	1.00	0.05	0.05	24.10

In Table 4.4, we observe the performance of AmbiGAN with Confusion Distance loss versus Gaussian loss for the 9v8 experiment. The results support the 7v1 conclusion that suggests Gaussian loss offers stable classification performance and allows for fine-tuning through variance adjustment.

Figure 4.7 shows the histograms for CD and Gaussian loss regarding the 9v8 experiment. The results of using Gaussian loss are consistent with 7v1 experiments and lead to a higher number of samples classified as 0.50 by the estimator. This indicates improved performance in generating images within the desired ambiguity range, proving the effectiveness of Gaussian loss for this application. The histogram results for the complete experiments can be found in Appendix F.

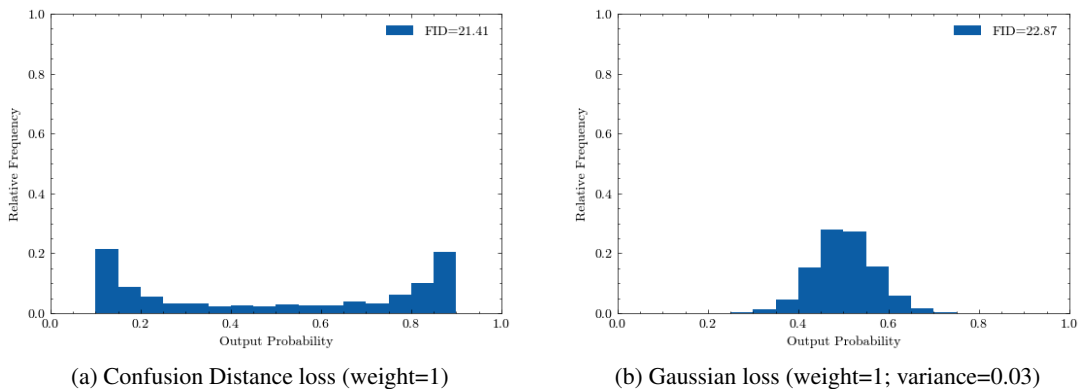


Figure 4.7: Histogram of results for 9v8 for CD and Gaussian losses with lower FID score

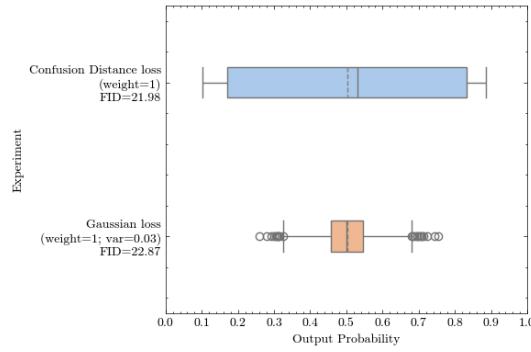


Figure 4.8: Boxplot of results for 9v8 for CD and Gaussian losses with lower FID score

The boxplot in Figure 4.8 supports these observations. For Gaussian loss, both median and mean are centered in 0.5, as previously observed in the 7v1 experiment. Some outliers are present in Gaussian loss results both below Q1 and above Q3, like in the 7v1 experiment.

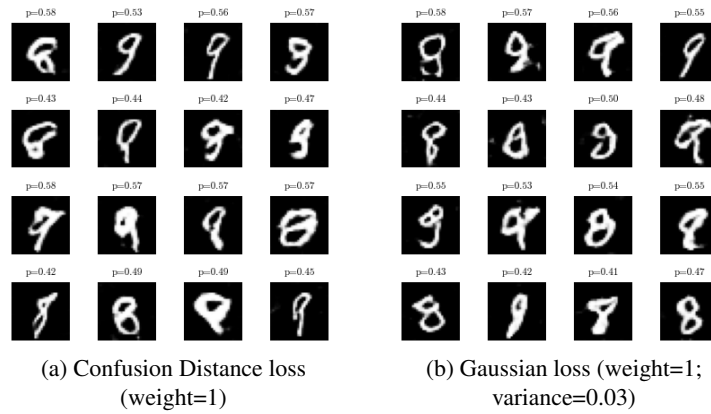


Figure 4.9: Examples of images of 9v8 experiment for CD and Gaussian loss

Figure 4.9 shows examples of the generated images for the 9v8 experiment. The images are realistic and present features from both classes. This is especially evident in the Gaussian loss set. We can even see a picture with an eight and a nine at the same time (third row, first column).

These findings highlight the effectiveness and stability of Gaussian loss compared to CD loss. Gaussian loss allows fine-tuning image quality and consistently produces more samples centered in the desired output probability range. For this reason, we selected Gaussian loss as the winner.

4.1.3 Ambiguity Estimators

From the previous experiment, we selected AmbiGANs and Gaussian loss as the best approach. In this test, we will compare different Ambiguity Estimators Output Combinations. So far, we were only using the Mean Estimator as baseline. Table 4.5, and 4.6 present results for 7v1 and 9v8 of different approaches for the Ambiguity Estimators outputs.

Table 4.5: Output combination comparison for 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
AmbiGAN	Mean	Gaussian	1.00	0.02	0.08	36.67
AmbiGAN	Linear	Gaussian	1.00	0.02	0.16	62.14
AmbiGAN	Meta-Learner	Gaussian	1.00	0.02	0.48	21.49
AmbiGAN	No-Output	Gaussian	0.005	0.001	0.22	49.20
AmbiGAN	No-Output	Gaussian	0.015	0.002	0.07	125.26
AmbiGAN	No-Output	Gaussian	0.03	0.005	0.11	68.06
AmbiGAN	No-Output	KL-Div	0.25	N/A	0.18	54.47
AmbiGAN	No-Output	KL-Div	0.50	N/A	0.09	122.42
AmbiGAN	No-Output	KL-Div	0.75	N/A	0.07	122.56

Table 4.5 compares the performance of various classifier outputs using different loss functions for the 7v1 dataset. For Mean, Linear and Meta-Learner, the variants with lower ACD and FID are presented. The full results for all hyperparameters can be found in Appendix D. For the Gaussian loss, a Mean classifier output shows a relatively balanced performance between ACD and FID. The Linear output, however, results in a significantly higher ACD and FID, indicating lower effectiveness. The Meta-Learner classifier output demonstrates the best FID but at the cost of a higher ACD.

For the No Output Combination classifier, various hyperparameter combinations are tested. With Gaussian loss increasing the weight improves ACD but worsens FID. For the KL-Div loss, higher weights generally result in better ACD but generally worse FID when compared to Gaussian.

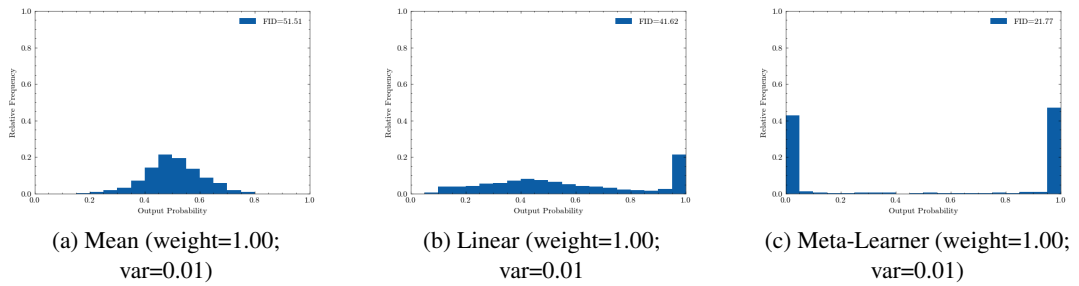


Figure 4.10: Histogram of results for 7v1 with different output combinations

In Figure 4.10, we can see the comparison in the distribution of results. The Mean output combination continues to show a balance in the distribution of the results compared to the Linear and Meta-Learner combinations. Although the Meta-Learner presents lower values of FID, we can observe this is due to the results being concentrated in 0 and 1 probabilities (not ambiguous images). Meta-Learner is creating the best combination based on the FID and not ACD. Between the different types of the estimators' output combinations, we can say the Mean combinator achieves

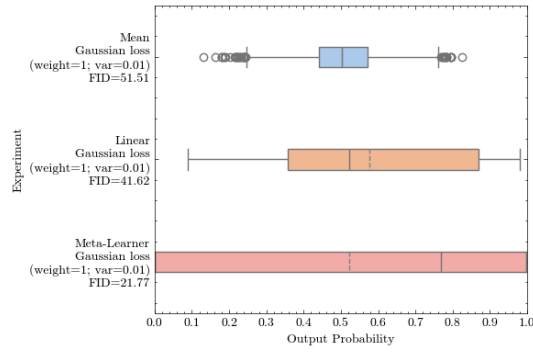


Figure 4.11: Boxplot of results for 7v1 for different output combinations

the best results with a good balance between ACD and FID and a distribution output centered in 0.5.

Figure 4.11 boxplot provides additional confirmation of these observations. For the Mean Output Combination, both the median and mean are centered at 0.50. For Linear the mean and median is roughly centered around 0.5. Meta-Learner Q2 is at 0.80. Linear Output Combination presents some right skewness; the meta-learner combinator is highly left skewed, while the results of the Mean combinator appear to present no skewness. The wider spread in the output probability results belongs to the Meta-Learner Combination (0.00 to 1.00), followed by the Linear Combination (approximately 0.1 to 1.0). In contrast, Gaussian loss presents less variability and spread (approximately between 0.30 and 0.70). However, 50% of the data is within approximately 0.45 and 0.55 output probability. Some outliers are present in Gaussian loss results both below Q1 and above Q3.

Examples of these images are shown in Figure 4.12. Images generated with the Meta-Learner Estimator appear very realistic when compared to the other estimators. However, they are also the less ambiguous of the three variations. Comparing images from Mean and Linear Estimators, the Linear set of images appears to be slightly more realistic. However, images from the Mean Estimator are more ambiguous.

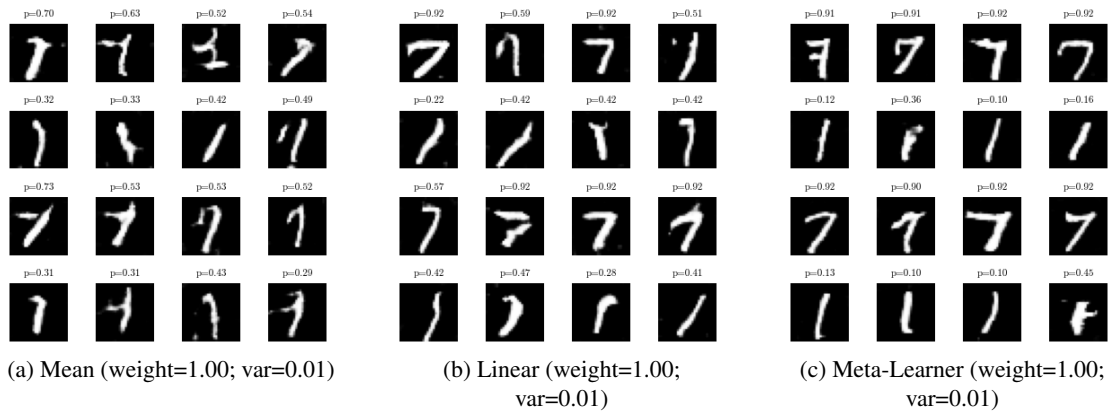


Figure 4.12: Examples of images of 7v1 with different output combinations

Figure 4.13 compares the no output combination distributions each using Gaussian and KL div losses. Since, in this experiment, there is no combination of the estimators' output for each image, the average probability from all estimators is calculated to plot the results. KL div loss results in worse FID and higher ACD in general. FID is slightly lower when compared with the best no output with Gaussian loss. However, we can see in Figure 4.13 that KL div loss results are more spread, which means we have a significant number of generated images that are not ambiguous. This explains the lower FID values. On the other hand, the Gaussian variant, with lower ACD scores, has a higher number of results centered at 0.5. This highlights that lower ACD should be considered a priority when choosing our approach, given it also maintains an acceptable FID score.

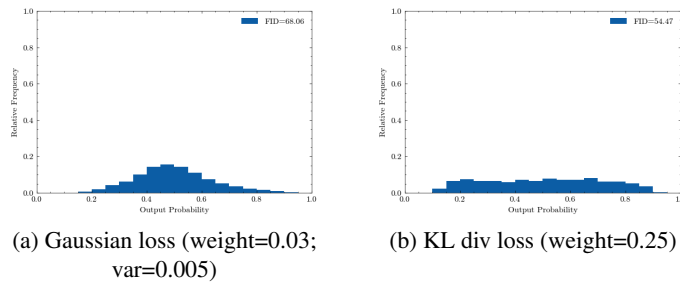


Figure 4.13: Histogram of results for 7v1 with no output combination

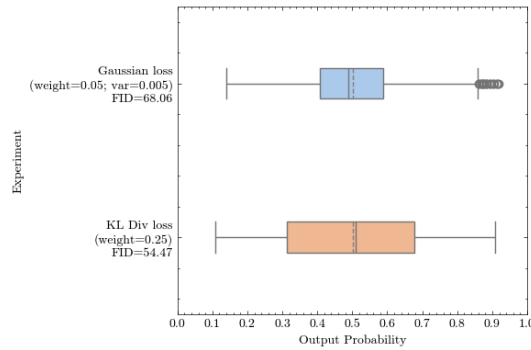


Figure 4.14: Boxplot of results for 7v1 with no output combination

Figure 4.14 gives us additional information. Gaussian loss presents a lower spread than KL Div loss, with 50% of the resulting images with an output probability between 0.4 and 0.6. This contrasts with KL div results which present 50% of the images between 0.3 and 0.7 approximately.

Let us look into another example from the Gaussian loss variant. The experiment with a weight of 0.005 and variance of 0.001 presents relatively low ACD (0.22) and acceptable FID (49.20). However, when we observe the histogram of results in Figure 4.15, we can see the higher frequency of results occurs near the probability of 1. This means most images generated were "7" digits and not ambiguous. That has an impact on low FID since the images are realistic. This highlights the importance of validating our results' distribution since we want most images

centered in 0.5. The relatively low ACD of this experiment is justified by the occurrence of some samples in between 0.3 and 0.7 probability output, which decreases the ACD. ACD indicates how centered our results might be, but cases like Figure 4.15 can still occur - one example of the limitation of using a metric based on averaging. We can see similar behavior in the results from the Linear combination (Figure 4.10).

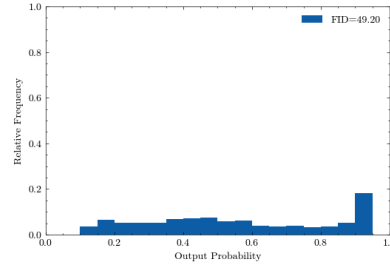


Figure 4.15: Histogram of No Output Combination with Gaussian loss (weight=0.005; var=0.001)

Examples of images obtained with each loss for the no output combination experiment are shown in Figure 4.16. The images from Gaussian loss appear to be more ambiguous than KL div loss. At the same time images from KL div, although presenting lower FID do not look like numbers. This also highlights the limitations of FID metric.

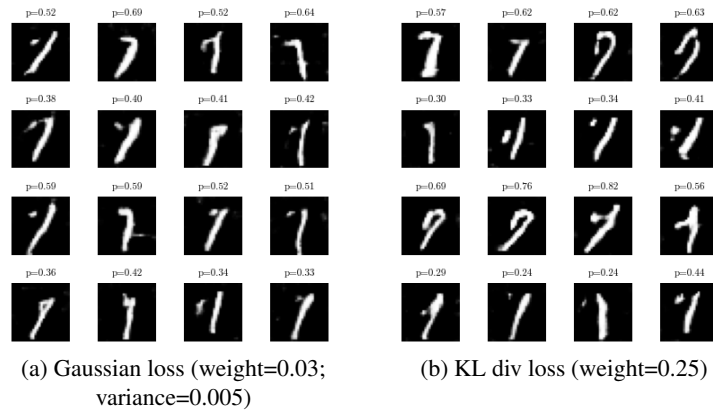


Figure 4.16: Examples of images of 7v1 experiment for No Output combination

Table 4.6 compares the performance of various classifier outputs using different loss functions for the 9v8 dataset. Generally, the results for 9v8 experiment present both lower ACD and FID than 7v1 across variants. This indicates that 9 and 8 might have more features in common, making it easier for AmbiGAN to generate images with features from both digits.

For Gaussian loss, the Mean Estimator achieves a balanced performance. The Linear Estimator results suggest lower effectiveness. The Meta-Learner Estimator output shows a moderate FID but a higher ACD. This is consistent with 7v1 experiment.

For the No Output Combination Estimator, we notice KL div achieves a better balance between ACD and FID than Gaussian, in contrast to the 7v1 experiment.

Table 4.6: Output combination comparison for 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
AmbiGAN	Mean	Gaussian	1.00	0.03	0.05	22.87
AmbiGAN	Linear	Gaussian	1.00	0.03	0.11	31.24
AmbiGAN	Meta-Learner	Gaussian	1.00	0.03	0.30	25.22
AmbiGAN	No-Output	Gaussian	0.005	0.001	0.08	36.66
AmbiGAN	No-Output	Gaussian	0.015	0.002	0.06	40.00
AmbiGAN	No-Output	Gaussian	0.03	0.005	0.07	32.64
AmbiGAN	No-Output	KL-Div	0.25	N/A	0.10	25.41
AmbiGAN	No-Output	KL-Div	0.50	N/A	0.08	30.14
AmbiGAN	No-Output	KL-Div	0.75	N/A	0.06	38.56

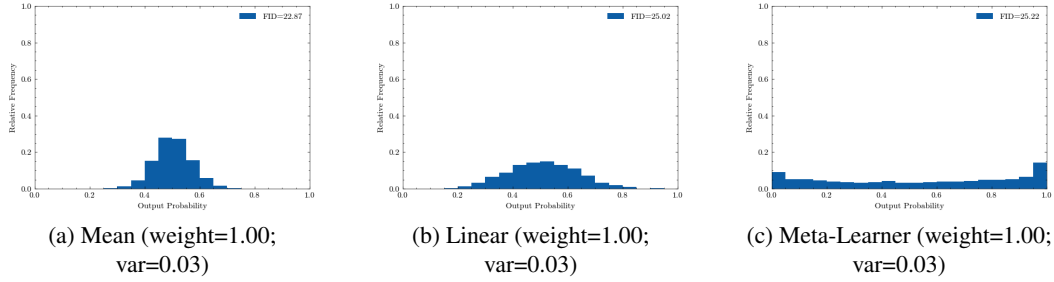


Figure 4.17: Histogram of results for 9v8 with different output combinations

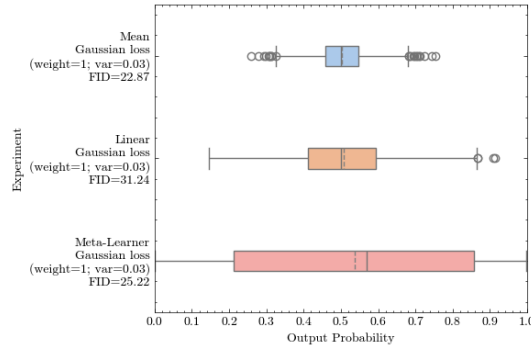


Figure 4.18: Boxplot of results for 9v8 with different output combinations

Figure 4.17 compares the distribution of results for the different combinations. For the Meta-Learner, we see a similar result as in the 7v1 experiment (Figure 4.10) with a higher density of results near 0 and 1 probability outputs. The Mean and Linear combinators' results appear centered in the 0.5 probability output. However, the Linear combinator presents higher ACD and FID values than the Mean Output combinator. This again suggests that the mean output is the technique that provides better results for estimators' output combinations.

In the boxplot from Figure 4.18, we can observe results similar to the 7v1's.

Examples of images generated with these types of combinations can be found in Figure 4.19. Most of the images appear realistic and without noise. As in the 7v1 experiment, the Meta-Learner appears to present less ambiguous examples. The Mean Estimator has a higher number of ambiguous examples.

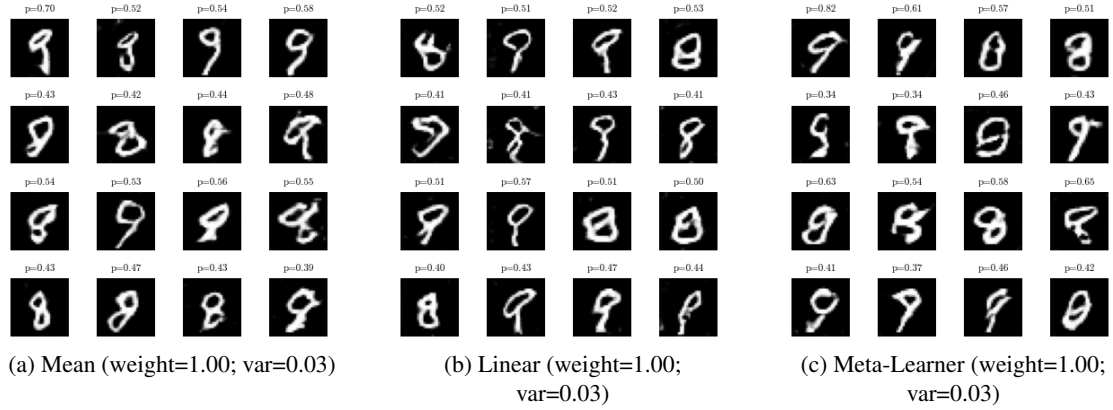


Figure 4.19: Examples of images of 9v8 with different output combinations

The comparison between the distributions with no output combination between Gaussian and KL div losses is shown in Figure 4.20. For the case of 9v8, we can observe that many results appear to be centered around 0.5 output probability for both losses. The results of the Gaussian loss variant present less variance than those of the KL div loss. This results in lower ACD values of Gaussian loss compared to KL div loss at the expense of slightly higher FID scores.

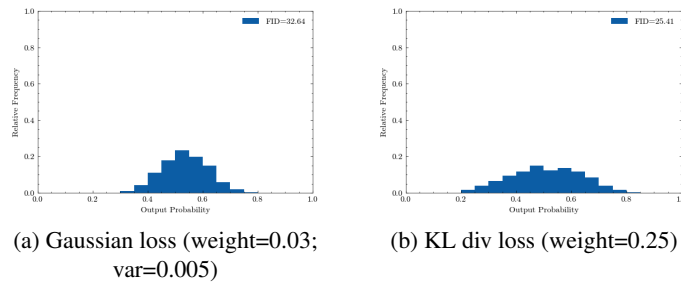


Figure 4.20: Histogram of results for 9v8 with no output combination

These observations can be confirmed in Figure 4.21. KL div loss presents Q2 in 0.5, while for Gaussian loss, it is slightly higher. Gaussian loss presents a lower spread of the results than KL div loss.

Examples of images generated with both variants can be seen in Figure 4.20. The visual results are consistent with 7v1 experiment.

Overall, the Mean classifier output with Gaussian loss offers the best balance between ACD and FID across experiments. However, the Meta-Learner output provides competitive FID values but at the cost of higher ACD. The No-Output classifier with Gaussian loss achieves good

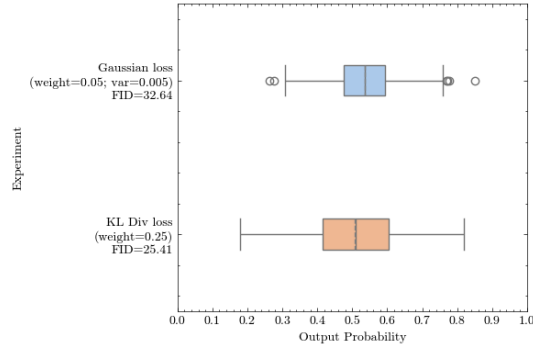


Figure 4.21: Boxplot of results for 9v8 with no output combination

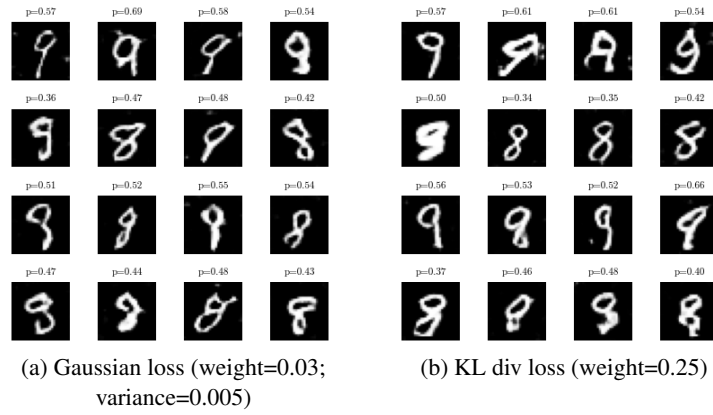


Figure 4.22: Examples of images of 9v8 experiment for No Output combination

ACD scores, though image quality, as indicated by FID, varies significantly with hyperparameter settings.

We can conclude that the Mean Output combination presents the most promising results for all output combinations tested in both the 7v1 and 9v8 experiments. In both 7v1 and 9v8 experiments for the No Output Combination, Gaussian loss presents better results for ACD and distribution of results while maintaining an acceptable FID score. This indicates Gaussian loss to be the preferable approach.

4.2 Hubris Evaluation of RESNET-50

Now that we have validated AmbiGAN architecture, we will test our proposed framework for Hubris benchmarking. We chose to use the datasets generated by the most promising combinator types. Hence, we decided to test the images from the Mean Output combination and the No Output combination, both with AmbiGAN architecture using Gaussian loss for the generator loss. Table 4.7 and Table 4.8 present the results for Hubris evaluation of RESNET-50 for 7v1 and 9v8, respectively, with different datasets. The original MNIST dataset was included in this evaluation

to present a reference for the results of a non-ambiguous dataset. The full results of this evaluation with all the datasets generated by AmbiGAN can be found in Appendix H.

Table 4.7: RESNET-50’s Hubris evaluation for 7v1

Dataset	AmbiGAN type	FID	ACD	Hubris _A ^a	Hubris _R ^b
MNIST [143]	-	3.409	0.500	0.972	0.053
AmbiguousMNIST [141]	-	68.955	0.497	0.971	0.259
AmbiGAN-MNIST	Mean Output Combination	52.796	0.483	0.966	0.891
AmbiGAN-MNIST	No Output Combination (Gaussian)	69.404	0.484	0.966	0.868

^a Absolute Hubris: Hubris(ref = 0.50)

^b Relative Hubris: Hubris(ref = Estimator Output Distribution)

Table 4.7 contains the results for the 7v1 experiment. FID scores were calculated for reference. For the MNIST dataset with RESNET-50, ACD shows a value of 0.5. This value is expected since MNIST contains realistic images that are far from the decision boundary. ACD reflects this distance. Absolute Hubris ($Hubris_A$) presents a high value (close to 1) because we measure the distance to the ideal ambiguous image distribution of 0.5. $Hubris_A$ is only appropriate for measuring the overconfidence of the model when we assume the dataset for that evaluation is ambiguous. In contrast, Relative Hubris ($Hubris_R$) presents a value closer to 0. This means the distribution of RESNET-50 results is very similar to those obtained with our Ambiguity Estimators. Therefore, for the MNIST dataset, we can assess that RESNET-50 presents low overconfidence.

For the AmbiguousMNIST, RESNET-50 results achieve an ACD slightly lower than for MNIST. However, the ACD value is still high. This means the average outputs of the pre-trained model are distant from the ideal 0.5 probability output. Since this dataset is targeted for ambiguous images, a model that is not overconfident should present an ACD value close to 0. FID scores for AmbiguousMNIST present an average result of 68.955. This magnitude is similar to our FID scores for the 7v1 experiment. This is a good indicator that the images generated by AmbiGAN present a comparable image quality with the current literature.

For the $Hubris_A$, the model presents a value close to one, indicating that RESNET-50 is very overconfident concerning the dataset and the ideal ambiguous distribution for reference. We cannot determine if AmbiguousMNIST is universally ambiguous. In other words, it’s unknown if AmbiguousMNIST presents an ideal ambiguous distribution (centered in 0.50 output probability with a variance of 0). $Hubris_A$ calculates the differences between the output distribution of RESNET-50 and the ideal ambiguous distribution. For this reason, it is convenient to compare the distribution outputs of the pre-trained with an empirical reference. In this case, the value of Relative Hubris ($Hubris_R$) is 0.259, indicating overconfidence compared to our reference Ambiguity Estimator output distribution.

When tested with both AmbiGAN-MNIST datasets, the resulting ACD scores are similar to the previous results. $Hubris_A$ values are similar to the previous ones, indicating high overconfidence towards this dataset concerning the ideal ambiguous distribution. However, when empirically compared, $Hubris_R$ values are closer to 1, indicating high overconfidence. Both AmbiGAN-MNIST datasets present similar $Hubris_R$ that are different from AmbiguousMNIST and MNIST datasets. When relatively compared, $Hubris_R$ presents higher values for AmbiGAN-MNIST datasets.

Table 4.8: RESNET-50’s Hubris evaluation for 9v8

Dataset	AmbiGAN type	FID	ACD	Hubris _A ^a	Hubris _R ^b
MNIST [143]	-	3.295	0.500	0.972	0.177
AmbiguousMNIST [141]	-	31.570	0.495	0.971	0.536
AmbiGAN-MNIST	Mean Output Combination	23.573	0.493	0.970	0.913
AmbiGAN-MNIST	No Output Combination (Gaussian)	32.349	0.492	0.970	0.904

^a Absolute Hubris: Hubris(ref = 0.50)

^b Relative Hubris: Hubris(ref = Estimator Output Distribution)

Table 4.8 presents similar results for the 9v8 experiment. ACD scores are lower for the AmbiGAN-MNIST datasets. FID scores are in the same range for the three ambiguous datasets. AmbiGAN-MNIST (Mean Output Combination) presents the lowest average FID, followed by AmbiguousMNIST. For AmbiguousMNIST and AmbiGAN-MNIST datasets, the $Hubris_A$ values are high, indicating a consistent distance to the reference value of 0.50 output probability. $Hubris_R$ shows varying degrees of alignment, with AmbiGAN-MNIST configurations achieving values close to 1.

All in all, the results suggest RESNET-50 presents overconfidence when facing ambiguous datasets since both Absolute and Relative Hubris have values close to 1. Given that it is not possible to determine if the ambiguous datasets present an ideal ambiguous distribution, Relative Hubris is useful for establishing a common reference between comparisons. The highest values of Relative Hubris were obtained with AmbiGAN generated datasets, meaning the pre-trained model presents a higher relative overconfidence in classifying examples from our datasets.

We can notice that in our experiments, ACD and $Hubris_A$ values remained very similar across the different datasets. However, $Hubris_R$ varied significantly. There are major differences between these two metrics. ACD measures average distances of probabilities, while Hubris measures distances in probability distributions. Additionally, while ACD always measures the distance to the ideal ambiguity example (located in the 0.5 probability output), Hubris presents more flexibility because it supports different types of references. In other words, we can use Hubris to measure the distance to any reference distribution. The ideal ambiguous distribution centered in 0.5 was used for our $Hubris_A$ reference. However, we also used $Hubris_R$, which relies on our ambiguity estimator distributions for reference.

Hubris can measure relative ambiguity to any ambiguity estimation methodology if the methodology outputs a probability. This is an advantage since it's hard to know the real ambiguity distribution of images. Using this relative Hubris, we can compare the probability distributions among models. Hence, we can quantify overconfidence, at least relatively. We notice this fact is evident in the experiments conducted in this research.

Finally, and since Hubris values are normalized between 0 and 1, it's easier for researchers to interpret the results. The closer the value is to one, the more overconfident the model is. So, we can view Hubris of 1 as the maximum that a model can have of *"Excessive pride, presumption or arrogance (originally toward the gods)"*.

4.3 Summary

This chapter presents and discusses the results of our work. In Section 4.1.1, we analyze the impact of architectural changes in AmbiGAN compared to the previous GASTeN framework using DCGAN. Section 4.1.2 provides a detailed comparison between Gaussian loss and Confusion Distance loss. The results from different approaches to the Ambiguity Estimator are discussed in Section 4.1.3. Finally, Section 4.2 presents the evaluation results of the pre-trained RESNET-50 model in the context of Hubris.

Chapter 5

Conclusions

In this Chapter, we present our conclusions. Section 5.1 describes our key findings and contributions. Section 5.2 provides a summarized answer to our research questions. In Section 5.3 we describe the limitations of this research and in Section 5.4 we detail some possible directions for future work.

5.1 Key Findings and Contributions

The availability of CV benchmark datasets targeting challenging or even truly ambiguous images is a crucial step for understanding models better before deployment. Additionally, ensuring transparency and accountability can foster wider AI adoption and ease human lives. This work aimed to approach the generation of these challenging images and evaluated the final result in one popular ML CV model (RESNET-50).

Our line of research started by addressing *universal ambiguity* by using an approximation that we denominated as ambiguity estimators. We propose a new GAN-based framework called *AmbiGAN*, to generate samples that these estimators would classify within the decision boundary while maintaining realistic image features. This approach is similar to adversarial attacks, but adversarial attacks aim to generate images with differences as slight as a pixel from the original image. In contrast, our images must be realistic even for the human eye.

In the initial stage, we noted that GASTeN architecture was unstable and had some minor discrepancies from the original DCGAN paper that could be corrected. We addressed these changes, and the results were significantly more stable with a small cost in the FID score. ACD values were lower for the new architecture. The results suggest that there exists a balance between ambiguity of the resulting images and quality. At the same time, we observed the AmbiGAN architecture is more permissive in minimizing the loss of the classifier during training. We consider the new AmbiGAN architecture to be the most advantageous.

Introducing the Gaussian loss in the Generator loss function was a crucial step. It allowed us to understand how to target the distribution of results that can lead to realistic data presenting features from both classes. The images and experimental results prove that using Gaussian loss along with

the new AmbiGAN architecture results in a higher proportion of examples being classified by the estimators in the decision boundary. When compared to CD loss, Gaussian loss presents a lower spread of results and stable performance. Gaussian loss also introduces an additional significant advantage by eliminating the necessity of a post-generation selection of images. Since we can tune the variance of the samples, we can adjust them to our desired range. Hence, we conclude Gaussian loss presents a superior performance than CD loss.

We introduced the concept of Ambiguity Estimators as a means to assess universal ambiguity. We created a set of 50 diverse classifiers with varying architectures, each trained with sub-sets of the experiment data. Our resulting samples lie in the decision boundaries of these classifiers. The outputs of the individual classifiers were handled in different approaches for a comparison of results. The Mean Output combination estimator shows a balanced performance between ACD and FID. Linear Output combination suggests lower effectiveness with higher ACD and FID as well as a wider spread in the output probabilities. Meta-Learner has the best FID although with also higher ACD - which proves not suitable for this purpose. Therefore, the Mean Estimator shows the most promising results for the Estimators that use a combination of outputs. As for the Estimators with no combination of outputs, Gaussian loss presents the best results in terms of ACD, FID and distribution of output probabilities. For this reason, both Mean Output combination and No Output Combination with Gaussian loss are considered the winners.

Through visual inspection, we can also state that most images contain features from both classes. In the case of the 7v1 experiment, we can see images that look like a 7 and a 1 simultaneously. The 9v8 experiment presents similar results and better FID scores. This fact can be due to some common features in the numbers 9 and 8 (such as the top upper circle) that can easily be captured by AmbiGAN.

Once our data was generated, we measured the predictions obtained from RESNET-50 and compared them with our ideal ambiguity distribution of 0.50. We formulated a metric based on KL-Divergence that we denominated as absolute Hubris. Hubris measures the overconfidence of a model when presented with ambiguous images. However, we do not know if a given dataset presents the ideal ambiguity distribution. Because of this we also presented the relative Hubris results to establish a common reference between comparisons. This type of Hubris references the predictions obtained by our set of Ambiguity Estimators instead of the ideal ambiguity 0.50 distribution. Our results have a reasonable average FID score compared to a similar dataset in the same challenging context — AmbiguousMNIST. In manual inspection, we can confirm that most samples present features from both classes while maintaining realistic features. Datasets generated by AmbiGAN (AmbiGAN-MNIST) presented the highest values of relative Hubris. This suggests RESNET-50 is most overconfident with our generated datasets.

This study explored various configurations and hyperparameters for GASTeN and AmbiGAN architectures. The results provide insights into their performance under different conditions that can be immediately useful. This breadth provides guidance for future research and practical applications leading to the creation of more transparent, accountable, and reliable AI systems.

By introducing the AmbiGAN framework and the concept of Ambiguity Estimators, we provide a novel method to create realistic images that reside within the decision boundaries of diverse classifiers. These images can be used to stress test models. At the same time, our Hubris Benchmarking methodology provides a procedure to test models before deployment. Our findings improve the understanding of model behavior in the presence of ambiguous data. Our research presents a starting point for building more reliable AI systems for different critical applications. This includes medical imaging, where distinguishing between similar-looking conditions is critical.

We also conclude this research contributed to the Sustainable Development Goals - 9: Industry, Innovation and Infrastructure [6]. The findings in this research present a step in the accountability of AI models, providing insights for further implementation of these systems in various fields. Therefore, fostering economic growth in a sustainable and responsible way.

5.2 Goals and Research Questions

The initial research questions of this work are presented in this section. We approach each one individually and address the question considering the results we have obtained and interpreted in Chapter 4.

How to improve GASTeN architecture for increased performance?

GASTeN initial formulation had some instabilities during training, as reported by Cunha *et al.* [5]. We changed DCGAN by replacing all linear layers with convolutional ones, as suggested by the original DCGAN authors. With this, we achieved a more stable GAN, with lower ACD values although with a small trade-off for the FID scores. Since the primary objective is to maximize the ambiguity in images, the new architecture proves to achieve the goal more effectively.

After these improvements, we changed the generator loss function, introducing Gaussian loss and increasing performance even further. The introduction of Gaussian loss added two new hyperparameters to our architecture: loss weight, and target distribution variance. The weight value impacts the balance between image ambiguity and realness. If a larger weight is used, it's expected that Gaussian loss will be more dominant than discriminator loss. On the other hand, the variance allows for fine-tuning image ambiguity in our desired range of output probability.

Gaussian loss consistently produced more samples centered in the decision boundary. This allowed us to experiment with different weights and variances while still maintaining diversity and good FID scores in the overall samples. ACD presents consistently lower values when compared with the original CD loss, highlighting its superior performance.

How to empirically estimate the ambiguity of an example?

The question of estimating the ambiguity of a given example was addressed by targeting an approximation to universal ambiguity as described in Chapter 3. In the case of CV, this work formulated a set of estimators. We used 50 CNNs with diverse decision boundaries, that collectively present high accuracy. Diversity stems from different layer configurations and features, as well as different subsets of data for their training.

This concept is applied for obtaining different predictions from each classifier that, globally, work as our ambiguity estimators.

How to adapt GASTeN to generate universally ambiguous examples?

We adopted the architecture changes and the new Generator loss function in our new AmbiGAN framework. To generate universally ambiguous examples, we included ambiguity estimators in our AmbiGAN framework.

We explored different methods for using the outputs from our 50 different CNNs. One way is to combine the outputs of all the classifiers into one single output. This can be done in several ways: Averaging the output of all classifiers; Adding a linear layer that learns the best combination of outputs; or adding a deep neural network that learns the best combination of outputs. Another approach is not to combine the outputs and use the sums of the losses of each classifier.

As described in Chapter 2, some researchers suggest that using deep ensembles allows distinguishing between aleatoric and epistemic uncertainty. One research also claims that the means of the predictions can be interpreted as aleatoric uncertainty. Since, in our case, we want to maximize this type of uncertainty, this claim aligns with the fact that the best results we obtained were with the ambiguity estimators with mean output combination. No output with Gaussian loss variation also presents satisfactory results in terms of ACD and FID.

We can conclude these methods achieve the desired outcome in generating universally ambiguous examples.

How to measure the robustness to the ambiguity of a model on a set of ambiguous examples?

We designed an architecture capable of generating ambiguous examples. We also defined a metric *Hubris*, that can measure a model's overconfidence. Finally, we formulated a methodology for assessing the robustness of ML models to ambiguous examples.

Relying upon our methodology, we tested a pre-trained RESNET-50 ML model with our generated samples, AmbiguousMNIST samples, and the original MNIST. Our evaluation suggests that RESNET-50 is overconfident in both AmbiguousMNIST and our samples, indicating that the model may present wrongly confident predictions when faced with challenging samples.

This evaluation of RESNET-50 allowed us to measure the robustness of this model when faced with different ambiguous examples. Our new methodology can be applied to measure the overconfidence of a model.

5.3 Limitations and Threats to Validity

We identify some limitations that may affect the conclusions obtained in this research. The findings related to DCGAN Architecture Changes (Section 4.1.1) and Generator Loss Function (Section 4.1.2) were already validated for statistical significance, and the results can be found in our conference article (Section 1.3). Due to time restrictions, we did not run our experiments several times with different random seeds and additional class pairs. Because of this limitation, each configuration was restricted to a single run. The Pareto efficiency was not calculated and the results presented were related to the last epoch run.

Due to time constraints, we did not apply the improvements presented in this research to different GAN architectures, such as WGAN. Therefore, we can not validate the behavior of AmbiGAN to other configurations of GANs. Also, due to time restrictions, this work did not complete a search for the best parameters for each type of experiment. In all experiments, our final best results achieved a relatively low FID score and ACD, which could be improved by further hyperparameters optimization.

The dataset used for training AmbiGAN was MNIST. This dataset is relatively simple and has very small image dimensions (28x28 pixels). Due to limited time, we did not train AmbiGAN with more complex datasets. We cannot assess how AmbiGAN will behave with datasets with RGB images and larger image sizes.

Although we performed a manual inspection in most experiments, our judgment could be biased. Human surveys could address this limitation, providing more confidence in our results.

FID application in this research and GASTeN have limitations, as referred to in Section 2.4. Because the inception network was trained with ImageNet examples, the FID may not truly capture the characteristics of the AmbiGAN-MNIST dataset.

As per our Hubris methodology, we applied it only to the RESNET-50 pre-trained model. This was due to time restrictions as well. Hubris methodology could benefit from more experimental evidence if other pre-trained models are stress-tested.

5.4 Future Work

Future work should address additional aspects to build upon the findings of this research. We should run each experiment at least three times. This would allow us to prove the generalization of our research and statistical significance.

Experimenting with different types of classifiers for the Ambiguity Estimators could be an important next step in expanding the universal ambiguity of the generated images.

In the future, it would be important to use additional datasets for AmbiGAN training so that we can assess its generalization more effectively. Fashion MNIST would improve the assessment in this direction as a first step. As a second step, expanding the AmbiGAN approach for more complex datasets with larger images and with more intrinsic features and classes could provide evidence for broader generalization.

Hubris methodology could be further validated if other pre-trained models are stress-tested. This would provide more experimental evidence to support the claims made in this research. Selecting diverse architectures and performing a quantitative analysis of the results could highlight the strengths and weaknesses of the methodology across different models.

References

- [1] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, “Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI,” *Inf. Fusion*, vol. 58, pp. 82–115, 2020.
- [2] Wikipedia, *My Wife and My Mother-in-Law* — *Wikipedia, the free encyclopedia*, <http://en.wikipedia.org/w/index.php?title=My%20Wife%20and%20My%20Mother-in-Law&oldid=1190352157>, [Online; accessed 26-January-2024], 2024.
- [3] Wikipedia, *Mona Lisa* — *Wikipedia, the free encyclopedia*, <http://pt.wikipedia.org/w/index.php?title=Mona%20Lisa&oldid=67120334>, [Online; accessed 26-January-2024], 2024.
- [4] M. Weiss, A. G. Gómez, and P. Tonella, “Generating and detecting true ambiguity: A forgotten danger in DNN supervision testing,” *Empir. Softw. Eng.*, vol. 28, no. 6, p. 146, 2023.
- [5] L. Cunha, C. Soares, A. Restivo, and L. F. Teixeira, “Gasten: Generative adversarial stress test networks,” in *IDA*, ser. Lecture Notes in Computer Science, vol. 13876, Springer, 2023, pp. 91–102.
- [6] U. Nations, *Transforming our world: The 2030 agenda for sustainable development*, <https://sdgs.un.org/2030agenda>, [Online; accessed 25-June-2024], 2015.
- [7] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A comprehensive survey of loss functions in machine learning,” *Annals of Data Science*, pp. 1–26, 2020.
- [8] D. A. Forsyth and J. Ponce, *Computer Vision - A Modern Approach, Second Edition*. Pitman, 2012.
- [9] J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D Heckerman, A. Smith, and M West, “Generative or discriminative? getting the best of both worlds,” *Bayesian statistics*, vol. 8, no. 3, pp. 3–24, 2007.
- [10] S. J. Prince, *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

- [11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [12] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.
- [13] T. Jebara, *Machine learning: discriminative and generative*. Springer Science & Business Media, 2012, vol. 755.
- [14] J. Gauvain and C. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 2, pp. 291–298, 1994.
- [15] N. Friedman and D. Koller, "Being bayesian about network structure. A bayesian approach to structure discovery in bayesian networks," *Mach. Learn.*, vol. 50, no. 1-2, pp. 95–125, 2003.
- [16] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
- [17] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.
- [18] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.
- [19] R. D. Levine and M. Tribus, "Maximum entropy formalism," in *Maximum Entropy Formalism Conference (1978: Massachusetts Institute of Technology)*, Mit Press, 1979.
- [20] I. Ulusoy and C. M. Bishop, "Comparison of generative and discriminative techniques for object detection and classification," in *Toward Category-Level Object Recognition*, ser. Lecture Notes in Computer Science, vol. 4170, Springer, 2006, pp. 173–195.
- [21] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, e2, 2014.
- [22] P. Kaur, S. K. Singh, I. Singh, and S. Kumar, "Exploring convolutional neural network in computer vision-based image classification," in *International Conference on Smart Systems and Advanced Computing (Syscom-2021)*, 2021.
- [23] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition* (Texts in Computer Science). Springer, 2022.
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [25] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

- [26] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
- [27] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *ISCAS*, IEEE, 2010, pp. 253–256.
- [28] Z. Guyu, G. Huang, H. He, H. He, and J. Ren, "Regional spatiotemporal collaborative prediction model for air quality," *IEEE Access*, vol. PP, pp. 1–1, Sep. 2019.
- [29] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. A. Velasco-Hernández, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *CVC (1)*, ser. Advances in Intelligent Systems and Computing, vol. 943, Springer, 2019, pp. 128–144.
- [30] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" In *ICCV*, IEEE Computer Society, 2009, pp. 2146–2153.
- [31] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2020.
- [32] R. K. Sinha, R. Pandey, and R. Pattnaik, "Deep learning for computer vision tasks: A review," *arXiv preprint arXiv:1804.03928*, 2018.
- [33] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100 134, 2021.
- [34] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3313–3332, 2023.
- [35] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *CVPR*, IEEE Computer Society, 2017, pp. 95–104.
- [36] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *ECCV (5)*, ser. Lecture Notes in Computer Science, vol. 9909, Springer, 2016, pp. 597–613.
- [37] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, IEEE Computer Society, 2017, pp. 2242–2251.
- [38] K. Fu, J. Peng, H. Zhang, X. Wang, and F. Jiang, "Image super-resolution based on generative adversarial networks: A brief review," 2020.

- [39] J. Guerreiro, P. Tomás, N. C. Garcia, and H. Aidos, “Super-resolution of magnetic resonance images using generative adversarial networks,” *Comput. Medical Imaging Graph.*, vol. 108, p. 102 280, 2023.
- [40] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *ICLR (Poster)*, 2016.
- [41] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised anomaly detection with generative adversarial networks to guide marker discovery,” in *IPMI*, ser. Lecture Notes in Computer Science, vol. 10265, Springer, 2017, pp. 146–157.
- [42] W. Shen and R. Liu, “Learning residual images for face attribute manipulation,” in *CVPR*, IEEE Computer Society, 2017, pp. 1225–1233.
- [43] H. Schieber, K. C. Demir, C. Kleinbeck, S. H. Yang, and D. Roth, “Indoor synthetic data generation: A systematic review,” *Computer Vision and Image Understanding*, p. 103 907, 2024.
- [44] K. Man and J. Chahl, “A review of synthetic image data and its use in computer vision,” *Journal of Imaging*, vol. 8, no. 11, p. 310, 2022.
- [45] M. Elasri, O. Elharrouss, S. Al-Máadeed, and H. Tairi, “Image generation: A review,” *Neural Process. Lett.*, vol. 54, no. 5, pp. 4609–4646, 2022.
- [46] A. Figueira and B. Vaz, “Survey on synthetic data generation, evaluation methods and gans,” *Mathematics*, vol. 10, no. 15, p. 2733, 2022.
- [47] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. K. Jain, “Adversarial attacks and defenses in images, graphs and text: A review,” *Int. J. Autom. Comput.*, vol. 17, no. 2, pp. 151–178, 2020.
- [48] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *ICLR (Poster)*, 2015.
- [49] C. G. Turhan and H. S. Bilge, “Recent trends in deep generative models: A review,” in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, IEEE, 2018, pp. 574–579.
- [50] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *ICLR*, 2014.
- [51] M. Tschannen, O. Bachem, and M. Lucic, “Recent advances in autoencoder-based representation learning,” *arXiv preprint arXiv:1812.05069*, 2018.
- [52] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 37, JMLR.org, 2015, pp. 2256–2265.
- [53] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M. Yang, “Diffusion models: A comprehensive survey of methods and applications,” *ACM Comput. Surv.*, vol. 56, no. 4, 105:1–105:39, 2024.

- [54] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.
- [55] H. Han, W. Wang, and B. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *ICIC (1)*, ser. Lecture Notes in Computer Science, vol. 3644, Springer, 2005, pp. 878–887.
- [56] I. Tomek, "A generalization of the k-nn rule," *IEEE Trans. Syst. Man Cybern.*, vol. 6, no. 2, pp. 121–126, 1976.
- [57] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.*, vol. 18, 17:1–17:5, 2017.
- [58] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, and E. A. Chavez-Urbiola, "Loss functions and metrics in deep learning. a review," *arXiv preprint arXiv:2307.02694*, 2023.
- [59] A. Golan and E. Maasoumi, "Information theoretic and entropy methods: An overview," *Econometric Reviews*, vol. 27, no. 4-6, pp. 317–328, 2008.
- [60] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, PMLR, 2017, pp. 214–223.
- [61] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.
- [62] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Comput.*, vol. 16, no. 5, pp. 1063–107, 2004.
- [63] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
- [64] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.
- [65] D. Nix and A. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 IEEE International Conference on Neural Networks*, vol. 1, 1994, pp. 55–60.
- [66] H. Pan, H. Han, S. Shan, and X. Chen, "Mean-variance loss for deep age estimation from a face," in *2018 IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5285–5294.
- [67] Y. Jin, *Multi-objective machine learning*. Springer Science & Business Media, 2006, vol. 16.
- [68] V. Guardieiro, "Multi-objective machine learning: A systematic review," Ph.D. dissertation, 2023.

- [69] I. Albuquerque, J. Monteiro, T. Doan, B. Considine, T. Falk, and I. Mitliagkas, “Multi-objective training of generative adversarial networks with multiple discriminators,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 202–211.
- [70] R. Caruana and A. Niculescu-Mizil, “Data mining in metric space: An empirical analysis of supervised learning performance criteria,” in *KDD*, ACM, 2004, pp. 69–78.
- [71] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” in *NIPS*, 2016, pp. 2226–2234.
- [72] M. Hossin and M. N. Sulaiman, “A review on evaluation metrics for data classification evaluations,” *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.
- [73] C. Ferri, J. Hernández-Orallo, and R. Modroiou, “An experimental comparison of performance measures for classification,” *Pattern Recognit. Lett.*, vol. 30, no. 1, pp. 27–38, 2009.
- [74] V. García, R. A. Mollineda, and J. S. Sánchez, “Index of balanced accuracy: A performance measure for skewed class distributions,” in *IbPRIA*, ser. Lecture Notes in Computer Science, vol. 5524, Springer, 2009, pp. 441–448.
- [75] S. S. Baraheem, T. Le, and T. V. Nguyen, “Image synthesis: A review of methods, datasets, evaluation metrics, and future outlook,” *Artif. Intell. Rev.*, vol. 56, no. 10, pp. 10 813–10 865, 2023.
- [76] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a laplacian pyramid of adversarial networks,” in *NIPS*, 2015, pp. 1486–1494.
- [77] L. Theis, A. van den Oord, and M. Bethge, “A note on the evaluation of generative models,” in *ICLR*, 2016.
- [78] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [79] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *CVPR*, IEEE Computer Society, 2016, pp. 2818–2826.
- [80] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *NIPS*, 2017, pp. 6626–6637.
- [81] M. Fréchet, “Sur la distance de deux lois de probabilité,” in *Annales de l’ISUP*, vol. 6, 1957, pp. 183–198.
- [82] R. McAllister, G. Kahn, J. Clune, and S. Levine, “Robustness to out-of-distribution inputs via task-aware generative uncertainty,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2083–2089.
- [83] X. Li, Y. Dai, Y. Ge, J. Liu, Y. Shan, and L.-Y. Duan, “Uncertainty modeling for out-of-distribution generalization,” *arXiv preprint arXiv:2202.03958*, 2022.

- [84] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” *Advances in neural information processing systems*, vol. 30, 2017.
- [85] A. Tahir, L. Cheng, and H. Liu, “Fairness through aleatoric uncertainty,” in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 2372–2381.
- [86] G. Brown, “Ensemble learning,” *Encyclopedia of machine learning*, vol. 312, pp. 15–19, 2010.
- [87] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, pp. 123–140, 1996.
- [88] Y. Tan, B. Jin, X. Yue, Y. Chen, and A. S. Vincentelli, “Exploiting uncertainties from ensemble learners to improve decision-making in healthcare ai,” *arXiv preprint arXiv:2007.06063*, 2020.
- [89] R. Rahaman *et al.*, “Uncertainty quantification and deep ensembles,” *Advances in neural information processing systems*, vol. 34, pp. 20 063–20 075, 2021.
- [90] K. Borys, Y. A. Schmitt, M. Nauta, C. Seifert, N. Krämer, C. M. Friedrich, and F. Nensa, “Explainable ai in medical imaging: An overview for clinical practitioners—beyond saliency-based xai approaches,” *European journal of radiology*, p. 110 786, 2023.
- [91] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” *Advances in neural information processing systems*, vol. 30, 2017.
- [92] F. K. Gustafsson, M. Danelljan, and T. B. Schon, “Evaluating scalable bayesian deep learning methods for robust computer vision,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 318–319.
- [93] R. Szeliski, “Bayesian modeling of uncertainty in low-level vision,” *International Journal of Computer Vision*, vol. 5, no. 3, pp. 271–301, 1990.
- [94] S. J. Godsill, “On the relationship between markov chain monte carlo methods for model uncertainty,” *Journal of computational and graphical statistics*, vol. 10, no. 2, pp. 230–248, 2001.
- [95] J. Fu and J. J. Gómez-Hernández, “Uncertainty assessment and data worth in groundwater flow and mass transport modeling using a blocking markov chain monte carlo method,” *Journal of Hydrology*, vol. 364, no. 3-4, pp. 328–341, 2009.
- [96] J. Dhamala, H. J. Arevalo, J. Sapp, B. M. Horáček, K. C. Wu, N. A. Trayanova, and L. Wang, “Quantifying the uncertainty in model parameters using gaussian process-based markov chain monte carlo in cardiac electrophysiology,” *Medical image analysis*, vol. 48, pp. 43–57, 2018.
- [97] M. S. Ayhan and P. Berens, “Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks,” in *Medical Imaging with Deep Learning*, 2022.

- [98] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, “Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks,” *Neurocomputing*, vol. 338, pp. 34–45, 2019.
- [99] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, “Test-time augmentation with uncertainty estimation for deep learning-based medical image segmentation,” 2022.
- [100] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, 2014, pp. 2672–2680.
- [101] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F. Wang, “Generative adversarial networks: Introduction and outlook,” *IEEE CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 588–598, 2017.
- [102] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative adversarial networks: An overview,” *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, 2018.
- [103] L. J. Ratliff, S. Burden, and S. S. Sastry, “Characterization and computation of local nash equilibria in continuous games,” in *Allerton*, IEEE, 2013, pp. 917–924.
- [104] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, “A learning algorithm for boltzmann machines,” *Cogn. Sci.*, vol. 9, no. 1, pp. 147–169, 1985.
- [105] Z. Wang, Q. She, and T. E. Ward, “Generative adversarial networks in computer vision: A survey and taxonomy,” *ACM Comput. Surv.*, vol. 54, no. 2, 37:1–37:38, 2022.
- [106] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [107] D. Saxena and J. Cao, “Generative adversarial networks (gans): Challenges, solutions, and future directions,” *ACM Comput. Surv.*, vol. 54, no. 3, 63:1–63:42, 2022.
- [108] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. C. Courville, “Adversarially learned inference,” in *ICLR (Poster)*, OpenReview.net, 2017.
- [109] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” *arXiv preprint arXiv:1605.09782*, 2016.
- [110] H. Alqahtani, M. Kavakli-Thorne, and G. Kumar, “Applications of generative adversarial networks (gans): An updated review,” *Archives of Computational Methods in Engineering*, vol. 28, pp. 525–552, 2021.
- [111] G. Mordido, H. Yang, and C. Meinel, “Dropout-gan: Learning from a dynamic ensemble of discriminators,” *arXiv preprint arXiv:1807.11346*, 2018.
- [112] J. Schulze, P. Sperl, and K. Böttinger, “Double-adversarial activation anomaly detection: Adversarial autoencoders are anomaly generators,” in *IJCNN*, IEEE, 2022, pp. 1–8.

- [113] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, “On evaluating adversarial robustness,” *arXiv preprint arXiv:1902.06705*, 2019.
- [114] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *ICLR (Poster)*, 2014.
- [115] I. Dunn, H. Pouget, T. Melham, and D. Kroening, “Adaptive generation of unrestricted adversarial inputs,” *arXiv preprint arXiv:1905.02463*, 2019.
- [116] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [117] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in *EuroS &P*, IEEE, 2016, pp. 372–387.
- [118] N. Carlini and D. A. Wagner, “Towards evaluating the robustness of neural networks,” in *IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2017, pp. 39–57.
- [119] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, “Adversarial examples are not bugs, they are features,” in *NeurIPS*, 2019, pp. 125–136.
- [120] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, “Adversarially robust generalization requires more data,” in *NeurIPS*, 2018, pp. 5019–5031.
- [121] K. Smith-Miles and S. Bowly, “Generating new test instances by evolving in instance space,” *Comput. Oper. Res.*, vol. 63, pp. 102–113, 2015.
- [122] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis, “Towards objective measures of algorithm performance across instance space,” *Comput. Oper. Res.*, vol. 45, pp. 12–24, 2014.
- [123] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, “Model cards for model reporting,” in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 220–229.
- [124] A. Chakarov, A. Nori, S. Rajamani, S. Sen, and D. Vijaykeerthy, “Debugging machine learning tasks,” *arXiv preprint arXiv:1603.07292*, 2016.
- [125] R. Lourenço, J. Freire, and D. E. Shasha, “Debugging machine learning pipelines,” in *DEEMSIGMOD*, ACM, 2019, 3:1–3:10.
- [126] E. Schoop, F. Huang, and B. Hartmann, “UMLAUT: debugging deep learning programs using program structure and model behavior,” in *CHI*, ACM, 2021, 310:1–310:16.
- [127] R. Yousefzadeh and D. P. O’Leary, “Auditing and debugging deep learning models via flip points: Individual-level and group-level analysis,” *La Matematica*, vol. 1, no. 1, pp. 19–52, 2022.

- [128] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, “Model assertions for debugging machine learning,” in *NeurIPS MLSys Workshop*, vol. 3, 2018, p. 10.
- [129] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Y. Simard, and J. Suh, “Model-tracker: Redesigning performance analysis tools for machine learning,” in *CHI*, ACM, 2015, pp. 337–346.
- [130] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. C. Guttula, S. Mujumdar, S. Afzal, R. S. Mittal, and V. Munigala, “Overview and importance of data quality for machine learning tasks,” in *KDD*, ACM, 2020, pp. 3561–3562.
- [131] N. Gupta, S. Mujumdar, H. Patel, S. Masuda, N. Panwar, S. Bandyopadhyay, S. Mehta, S. C. Guttula, S. Afzal, R. S. Mittal, and V. Munigala, “Data quality for machine learning tasks,” in *KDD*, ACM, 2021, pp. 4040–4041.
- [132] S. Grafberger, P. Groth, J. Stoyanovich, and S. Schelter, “Data distribution debugging in machine learning pipelines,” *VLDB J.*, vol. 31, no. 5, pp. 1103–1126, 2022.
- [133] Y. Li, L. Meng, L. Chen, L. Yu, D. Wu, Y. Zhou, and B. Xu, “Training data debugging for the fairness of machine learning software,” in *ICSE*, ACM, 2022, pp. 2215–2227.
- [134] A. Sotgiu, M. Pintor, and B. Biggio, “Explainability-based debugging of machine learning for vulnerability discovery,” in *ARES*, ACM, 2022, 113:1–113:8.
- [135] A. Sotgiu, M. Pintor, and B. Biggio, “Explainability-based debugging of machine learning for vulnerability discovery,” in *ARES*, ACM, 2022, 113:1–113:8.
- [136] L. Aroyo and P. Paritosh, “Uncovering unknown unknowns in machine learning,” *Google AI Blog*. Available online at: <https://ai.googleblog.com/2021/02/uncovering-unknown-unknowns-in-machine.html>, 2021.
- [137] D. Lines and M. Van Der Wilk, “Disentangling sources of uncertainty for active exploration,” Ph.D. dissertation, Master’s thesis, Department of Engineering, University of Cambridge, 2019.
- [138] W. R. Clements, B. Van Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, “Estimating risk and uncertainty in deep reinforcement learning,” *arXiv preprint arXiv:1905.09638*, 2019.
- [139] Y. Liu, L. Feng, X. Wang, and S. Zhang, “Deepboundary: A coverage testing method of deep learning software based on decision boundary representation,” in *QRS Companion*, IEEE, 2022, pp. 166–172.
- [140] T. Trappenberg and A. D. Back, “A classification scheme for applications with ambiguous data,” in *IJCNN (6)*, IEEE Computer Society, 2000, pp. 296–301.
- [141] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal, “Deterministic neural networks with inductive biases capture epistemic and aleatoric uncertainty,” *arXiv preprint arXiv:2102.11582*, vol. 2, 2021.
- [142] R. Herbei and M. H. Wegkamp, “Classification with reject option,” *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pp. 709–721, 2006.

- [143] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in *NIPS*, Morgan Kaufmann, 1989, pp. 396–404.
- [144] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” in *NIPS*, 2016, pp. 2172–2180.
- [145] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 2642–2651.
- [146] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, “Generating images with recurrent adversarial networks,” *arXiv preprint arXiv:1602.05110*, 2016.

Appendix A

Table A.1: Overview of different GAN architectures

Criteria	Vanilla GAN[100]	WGAN[60]	InfoGAN[144]	DCGAN[40]	BiGAN[109]
Learning Network Architecture	Unsupervised Multilayer Perceptrons	Unsupervised Multilayer Perceptrons	Unsupervised Multilayer Perceptrons	Unsupervised CNNs with constraints	Unsupervised CNNs
Gradient updates	SGD with k steps for D and 1 step for G Minimize value function for G and maximize for D	SGD with k steps for D and 1 step for G Minimize value function for G and maximize for D	SGD updates to G and D Disentangled representations maximizing mutual information	SGD for both G and D Learn hierarchy of representations from parts to scenes in G and D	SGD with k steps for D and 1 step for G Minimize value function for G and E and maximize for D
Methodology					
Distributions distance	JS Divergence	EM Distance	Mutual information	JS Divergence	JS Divergence
Performance metrics	Log-Likelihood	Generative adversarial metric	Information and representation	Accuracy and error rate	Accuracy
Criteria	CGAN[106]	ACGAN[145]	CycleGAN[37]	GRAN[146]	GATeN[5]
Learning Network Architecture	Supervised Multilayer Perceptrons	Supervised Multilayer Perceptrons	Unsupervised CNNs as adversarial autoencoders	Unsupervised Recurrent convolutional networks with constraints	Supervised CNNs
Gradient updates	SGD with k steps for D and 1 step for G	SGD with k steps for D and 1 step for G	SGD with k steps for D and 1 step for G	conditional SGD for G if prediction is correct and wrong for D if prediction is wrong	SGD with k steps for D and 1 step for G
Methodology	Minimize value function for G and maximize for D conditioned on extra information	Maximize log-likelihood of source and class for D and maximize the difference of log-likelihood of class and source for G	Minimize value function for G and F and maximize for D_X and D_Y while maintaining image characteristics across domains	Generation of images by incremental updates on a "canvas"	Minimize value function and C for G and maximize for D
Distributions distance	JS Divergence	JS Divergence	JS Divergence, Cycle consistency loss & Identity loss	JS Divergence	JS Divergence
Performance metrics	Log-Likelihood	MS-SSIM & Image discriminability	Accuracy, IOU & Visual Inspection	Generative adversarial metric	FID, IS & Visual Inspection

Appendix B

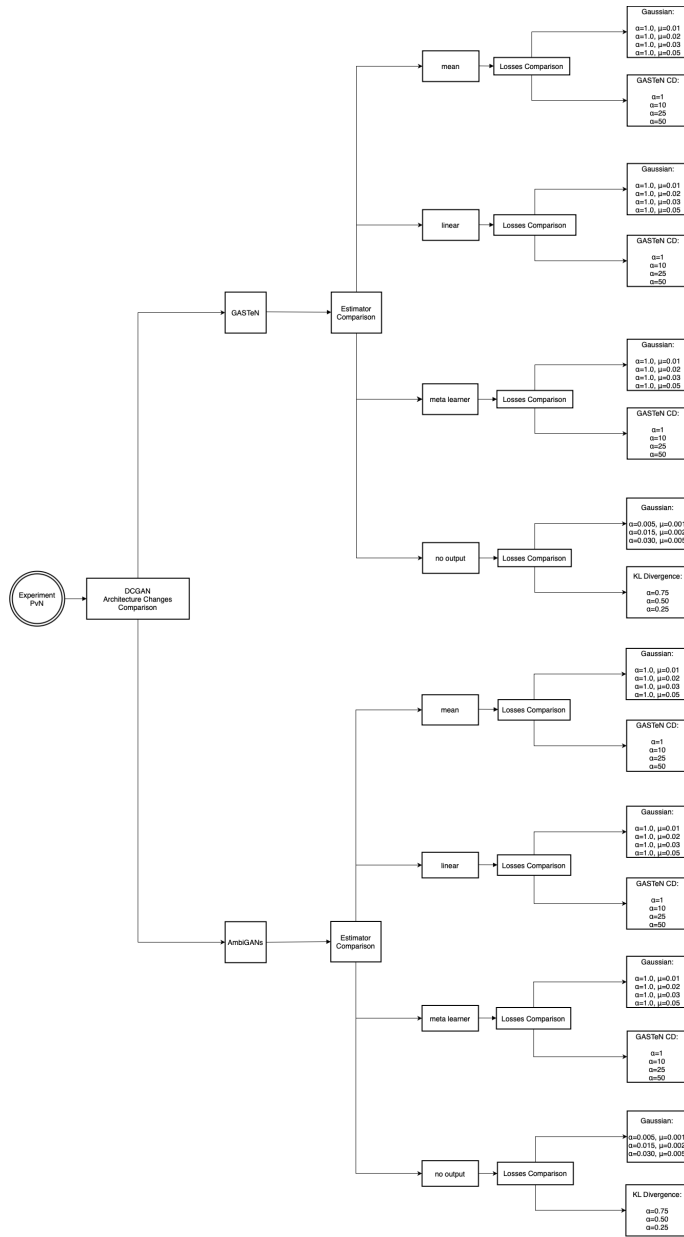


Figure B.1: Experiments diagram

Appendix C

C.1 Ambiguity Estimators Accuracy

Table C.1: Accuracy for different Ambiguity Estimators used in AmbiGAN training for image generation

Ambiguity Estimator	Dataset	Accuracy
Mean Output Combination Classifiers	MNIST: 7v1	0.993
Liner Output Combination Classifiers	MNIST: 7v1	0.994
Meta-Learner Output Combination Classifiers	MNIST: 7v1	0.994
Mean Output Combination Classifiers	MNIST: 9v8	0.975
Liner Output Combination Classifiers	MNIST: 9v8	0.973
Meta-Learner Output Combination Classifiers	MNIST: 9v8	0.982

Table C.2: Accuracy for different Ambiguity Estimators used in Hubris Evaluation

Ambiguity Estimator	Dataset	Accuracy
Mean Output Combination Classifiers	MNIST: 7v1	0.994
Mean Output Combination Classifiers	MNIST: 9v8	0.982

C.2 Pre-trained RESNET-50 Accuracy

Table C.3: Accuracy of pre-trained RESNET-50 for Hubris Evaluation

Dataset	Accuracy
MNIST: 7v1	1.000
MNIST: 9v8	1.000

Appendix D

D.1 7v1 Experiment

Table D.1: Full results for Mean Output Combination Estimator 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Mean	Gaussian	1.00	0.01	0.23	209.20
GASTeN	Mean	Gaussian	1.00	0.02	0.04	193.39
GASTeN	Mean	Gaussian	1.00	0.03	0.00	263.11
GASTeN	Mean	Gaussian	1.00	0.05	0.21	189.82
GASTeN	Mean	Confusion Distance	1	N/A	0.38	15.04
GASTeN	Mean	Confusion Distance	10	N/A	0.38	11.97
GASTeN	Mean	Confusion Distance	25	N/A	0.38	12.63
GASTeN	Mean	Confusion Distance	50	N/A	0.37	16.74
AmbiGAN	Mean	Gaussian	1.00	0.01	0.08	51.51
AmbiGAN	Mean	Gaussian	1.00	0.02	0.08	36.67
AmbiGAN	Mean	Gaussian	1.00	0.03	0.08	42.20
AmbiGAN	Mean	Gaussian	1.00	0.05	0.09	44.67
AmbiGAN	Mean	Confusion Distance	1	N/A	0.37	22.45
AmbiGAN	Mean	Confusion Distance	10	N/A	0.37	20.88
AmbiGAN	Mean	Confusion Distance	25	N/A	0.33	22.48
AmbiGAN	Mean	Confusion Distance	50	N/A	0.06	58.76

Table D.2: Full results for Linear Output Combination Estimator 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Linear	Gaussian	1.00	0.01	0.26	168.84
GASTeN	Linear	Gaussian	1.00	0.02	0.25	148.94
GASTeN	Linear	Gaussian	1.00	0.03	0.10	189.97
GASTeN	Linear	Gaussian	1.00	0.05	0.14	290.27
GASTeN	Linear	Confusion Distance	1	N/A	0.43	15.78
GASTeN	Linear	Confusion Distance	10	N/A	0.43	18.57
GASTeN	Linear	Confusion Distance	25	N/A	0.43	13.63
GASTeN	Linear	Confusion Distance	50	N/A	0.42	11.59
AmbiGAN	Linear	Gaussian	1.00	0.01	0.23	41.62
AmbiGAN	Linear	Gaussian	1.00	0.02	0.16	62.14
AmbiGAN	Linear	Gaussian	1.00	0.03	0.14	46.45
AmbiGAN	Linear	Gaussian	1.00	0.05	0.24	45.05
AmbiGAN	Linear	Confusion Distance	1	N/A	0.42	19.49
AmbiGAN	Linear	Confusion Distance	10	N/A	0.42	21.02
AmbiGAN	Linear	Confusion Distance	25	N/A	0.41	21.36
AmbiGAN	Linear	Confusion Distance	50	N/A	0.20	46.71

Table D.3: Full results for Meta-Learner Output Combination Estimator 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Meta-Learner	Gaussian	1.00	0.01	0.49	11.83
GASTeN	Meta-Learner	Gaussian	1.00	0.02	0.49	11.53
GASTeN	Meta-Learner	Gaussian	1.00	0.03	0.49	15.03
GASTeN	Meta-Learner	Gaussian	1.00	0.05	0.49	17.70
GASTeN	Meta-Learner	Confusion Distance	1	N/A	0.49	13.40
GASTeN	Meta-Learner	Confusion Distance	10	N/A	0.49	10.99
GASTeN	Meta-Learner	Confusion Distance	25	N/A	0.49	12.92
GASTeN	Meta-Learner	Confusion Distance	50	N/A	0.49	12.66
AmbiGAN	Meta-Learner	Gaussian	1.00	0.01	0.47	21.77
AmbiGAN	Meta-Learner	Gaussian	1.00	0.02	0.48	21.49
AmbiGAN	Meta-Learner	Gaussian	1.00	0.03	0.47	21.52
AmbiGAN	Meta-Learner	Gaussian	1.00	0.05	0.47	22.02
AmbiGAN	Meta-Learner	Confusion Distance	1	N/A	0.49	21.67
AmbiGAN	Meta-Learner	Confusion Distance	10	N/A	0.49	22.23
AmbiGAN	Meta-Learner	Confusion Distance	25	N/A	0.49	21.92
AmbiGAN	Meta-Learner	Confusion Distance	50	N/A	0.48	24.87

Table D.4: Full results for No Output Combination Estimator 7v1

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	No-Output	Gaussian	0.005	0.001	0.37	11.12
GASTeN	No-Output	Gaussian	0.015	0.002	0.37	13.66
GASTeN	No-Output	Gaussian	0.03	0.005	0.37	10.90
GASTeN	No-Output	KL-Div	0.25	N/A	0.37	21.16
GASTeN	No-Output	KL-Div	0.50	N/A	0.09	342.32
GASTeN	No-Output	KL-Div	0.75	N/A	0.08	243.49
AmbiGAN	No-Output	Gaussian	0.005	0.001	0.22	49.20
AmbiGAN	No-Output	Gaussian	0.015	0.002	0.07	125.26
AmbiGAN	No-Output	Gaussian	0.03	0.005	0.11	68.06
AmbiGAN	No-Output	KL-Div	0.25	N/A	0.18	54.47
AmbiGAN	No-Output	KL-Div	0.50	N/A	0.09	122.42
AmbiGAN	No-Output	KL-Div	0.75	N/A	0.07	122.56

D.2 9v8 Experiments

Table D.5: Full results for Mean Output Combination Estimator 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Mean	Gaussian	1.00	0.01	0.16	118.29
GASTeN	Mean	Gaussian	1.00	0.02	0.19	50.04
GASTeN	Mean	Gaussian	1.00	0.03	0.18	90.74
GASTeN	Mean	Gaussian	1.00	0.05	0.14	124.35
GASTeN	Mean	Confusion Distance	1	N/A	0.32	11.85
GASTeN	Mean	Confusion Distance	10	N/A	0.31	10.76
GASTeN	Mean	Confusion Distance	25	N/A	0.30	17.02
GASTeN	Mean	Confusion Distance	50	N/A	0.01	188.79
AmbiGAN	Mean	Gaussian	1.00	0.01	0.05	24.08
AmbiGAN	Mean	Gaussian	1.00	0.02	0.05	24.79
AmbiGAN	Mean	Gaussian	1.00	0.03	0.05	22.87
AmbiGAN	Mean	Gaussian	1.00	0.05	0.05	24.10
AmbiGAN	Mean	Confusion Distance	1	N/A	0.29	21.41
AmbiGAN	Mean	Confusion Distance	10	N/A	0.21	21.91
AmbiGAN	Mean	Confusion Distance	25	N/A	0.08	24.04
AmbiGAN	Mean	Confusion Distance	50	N/A	0.05	38.78

Table D.6: Full results for Linear Output Combination Estimator 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Linear	Gaussian	1.00	0.01	0.31	36.97
GASTeN	Linear	Gaussian	1.00	0.02	0.32	20.18
GASTeN	Linear	Gaussian	1.00	0.03	0.32	24.85
GASTeN	Linear	Gaussian	1.00	0.05	0.31	42.48
GASTeN	Linear	Confusion Distance	1	N/A	0.39	11.98
GASTeN	Linear	Confusion Distance	10	N/A	0.39	10.88
GASTeN	Linear	Confusion Distance	25	N/A	0.38	9.92
GASTeN	Linear	Confusion Distance	50	N/A	0.36	19.99
AmbiGAN	Linear	Gaussian	1.00	0.01	0.10	25.02
AmbiGAN	Linear	Gaussian	1.00	0.02	0.10	28.83
AmbiGAN	Linear	Gaussian	1.00	0.03	0.11	31.24
AmbiGAN	Linear	Gaussian	1.00	0.05	0.10	30.52
AmbiGAN	Linear	Confusion Distance	1	N/A	0.37	21.98
AmbiGAN	Linear	Confusion Distance	10	N/A	0.31	21.15
AmbiGAN	Linear	Confusion Distance	25	N/A	0.12	32.40
AmbiGAN	Linear	Confusion Distance	50	N/A	0.10	27.10

Table D.7: Full results for Meta-Learner Output Combination Estimator 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	Meta-Learner	Gaussian	1.00	0.01	0.46	14.24
GASTeN	Meta-Learner	Gaussian	1.00	0.02	0.46	17.47
GASTeN	Meta-Learner	Gaussian	1.00	0.03	0.46	11.74
GASTeN	Meta-Learner	Gaussian	1.00	0.05	0.26	21.82
GASTeN	Meta-Learner	Confusion Distance	1	N/A	0.48	14.22
GASTeN	Meta-Learner	Confusion Distance	10	N/A	0.48	13.26
GASTeN	Meta-Learner	Confusion Distance	25	N/A	0.48	15.33
GASTeN	Meta-Learner	Confusion Distance	50	N/A	0.47	13.67
AmbiGAN	Meta-Learner	Gaussian	1.00	0.01	0.30	25.12
AmbiGAN	Meta-Learner	Gaussian	1.00	0.02	0.30	22.65
AmbiGAN	Meta-Learner	Gaussian	1.00	0.03	0.30	25.22
AmbiGAN	Meta-Learner	Gaussian	1.00	0.05	0.29	29.72
AmbiGAN	Meta-Learner	Confusion Distance	1	N/A	0.47	18.98
AmbiGAN	Meta-Learner	Confusion Distance	10	N/A	0.45	20.01
AmbiGAN	Meta-Learner	Confusion Distance	25	N/A	0.39	25.51
AmbiGAN	Meta-Learner	Confusion Distance	50	N/A	0.21	34.29

Table D.8: Full results for No Output Combination Estimator 9v8

Architecture	Classifier Output	Loss	Loss hyperparams		ACD	FID
			Weight	Variance		
GASTeN	No-Output	Gaussian	0.005	0.001	0.01	392.241
GASTeN	No-Output	Gaussian	0.015	0.002	0.30	26.72
GASTeN	No-Output	Gaussian	0.03	0.005	0.30	12.24
GASTeN	No-Output	KL-Div	0.25	N/A	0.30	19.28
GASTeN	No-Output	KL-Div	0.50	N/A	0.27	55.23
GASTeN	No-Output	KL-Div	0.75	N/A	0.02	320.83
AmbiGAN	No-Output	Gaussian	0.005	0.001	0.08	36.66
AmbiGAN	No-Output	Gaussian	0.015	0.002	0.06	40.00
AmbiGAN	No-Output	Gaussian	0.03	0.005	0.07	32.64
AmbiGAN	No-Output	KL-Div	0.25	N/A	0.10	25.41
AmbiGAN	No-Output	KL-Div	0.50	N/A	0.08	30.14
AmbiGAN	No-Output	KL-Div	0.75	N/A	0.06	38.56

Appendix E

E.1 7v1 experiment

E.1.1 Mean Output Combination Estimator

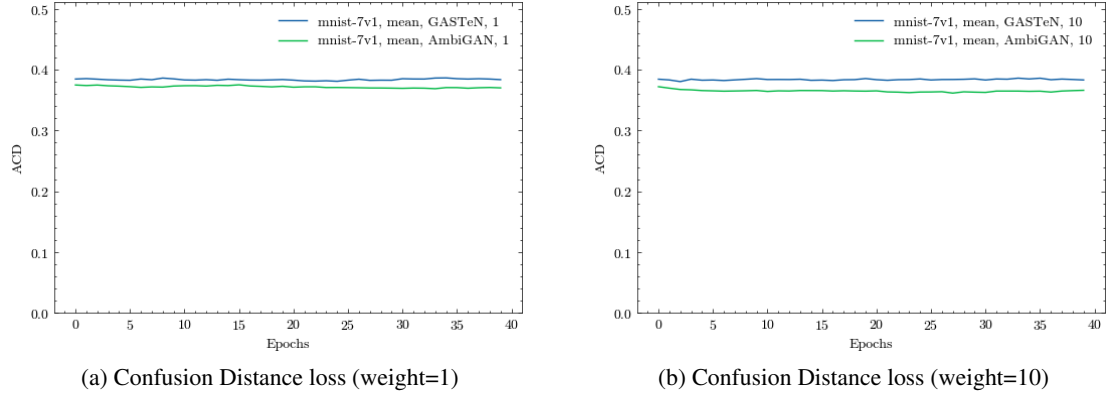


Figure E.1: Comparison of ACD during training, Mean Estimator, CD loss weights 1 and 10

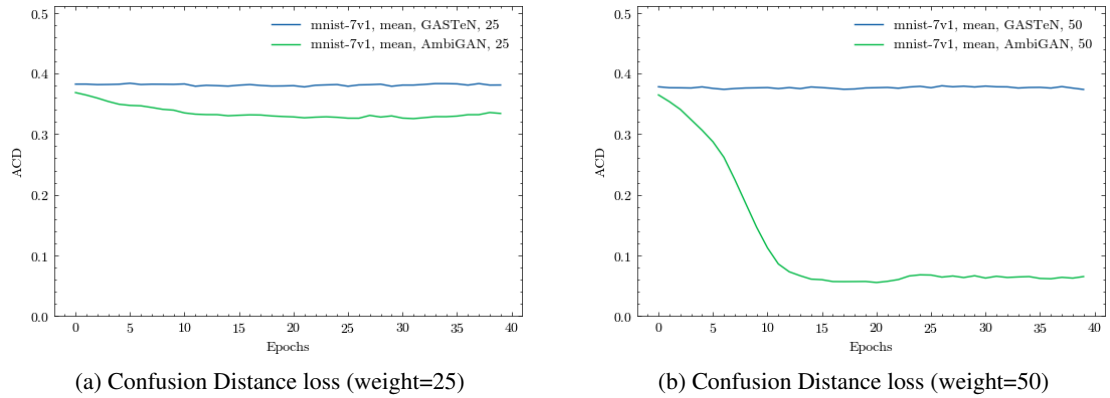


Figure E.2: Comparison of ACD during training, Mean Estimator, CD loss weights 25 and 50

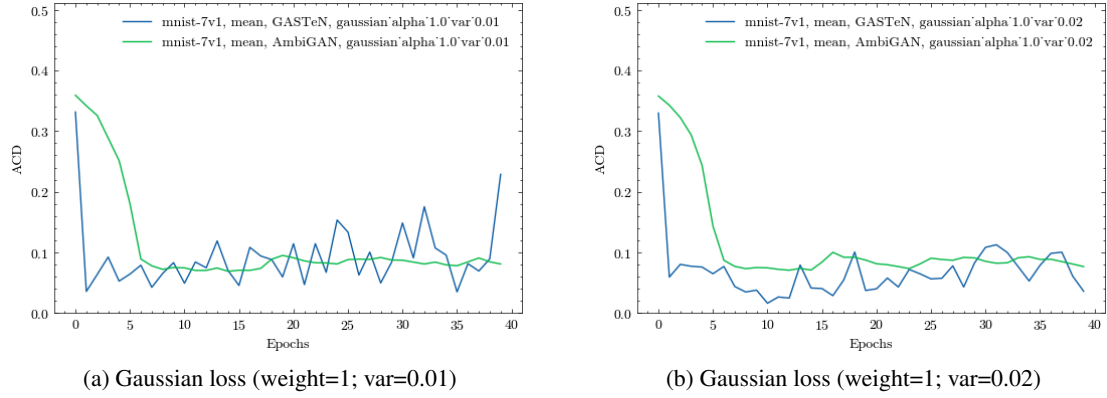


Figure E.3: Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02

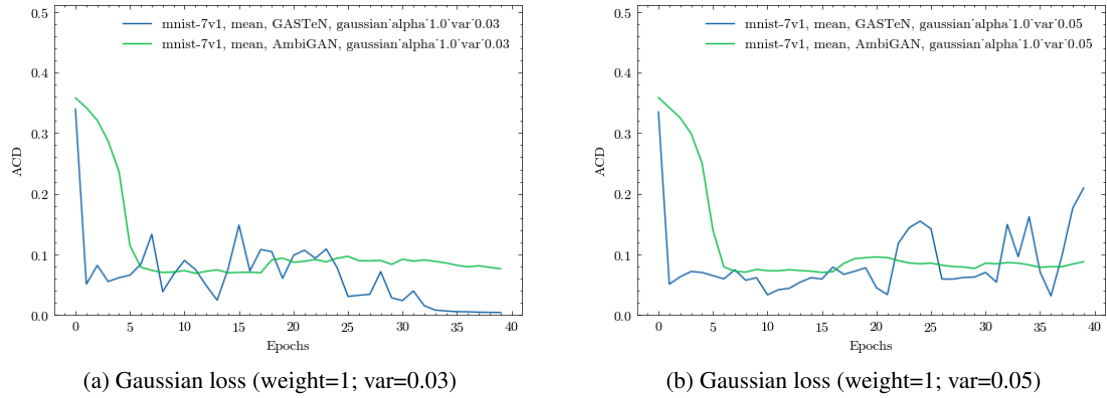


Figure E.4: Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05

E.1.2 Linear Output Combination Estimator

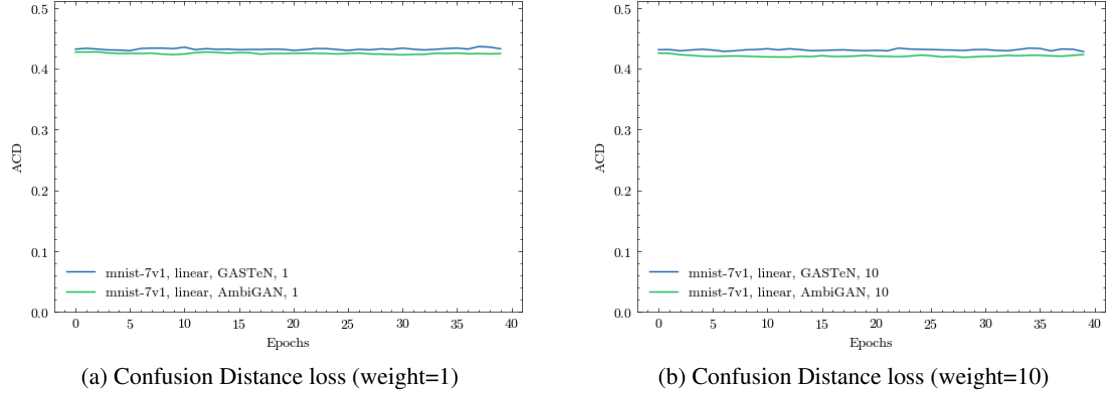


Figure E.5: Comparison of ACD during training, Linear Estimator, CD loss weights 1 and 10

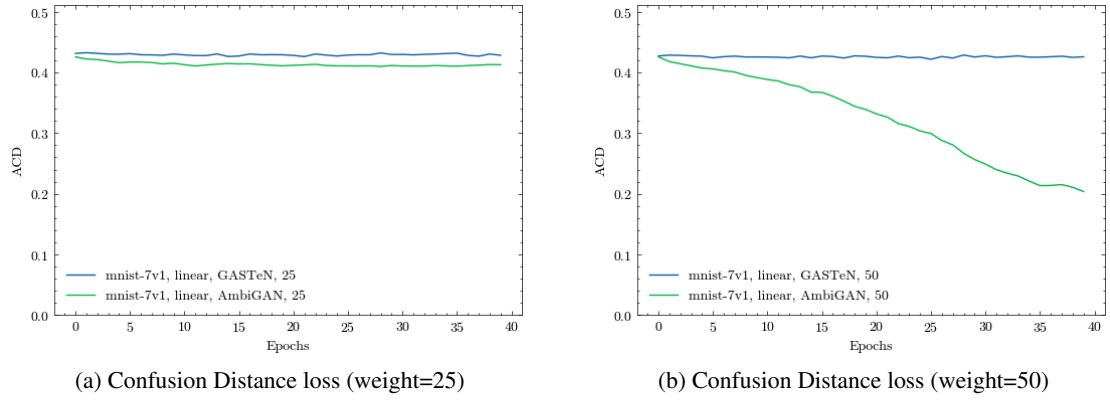


Figure E.6: Comparison of ACD during training, Linear Estimator, CD loss weights 25 and 50

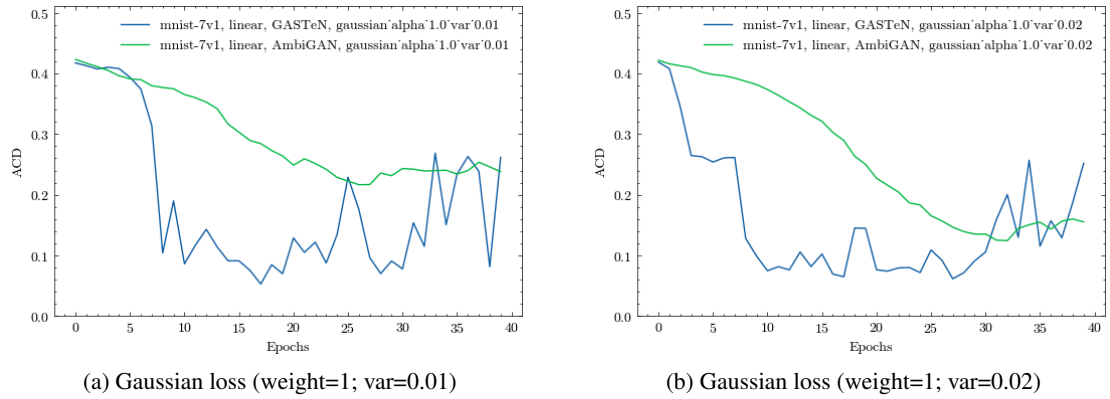


Figure E.7: Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02

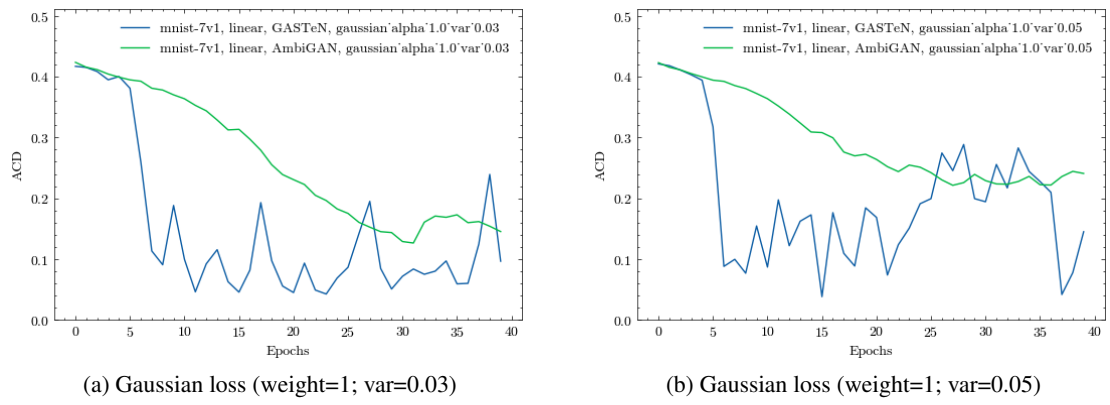


Figure E.8: Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05

E.1.3 Meta-Learner Output Combination Estimator

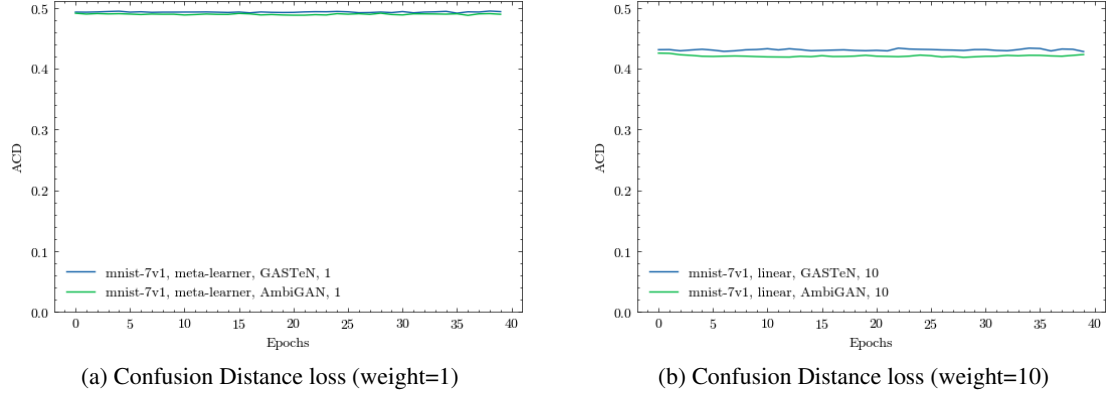


Figure E.9: Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 1 and 10

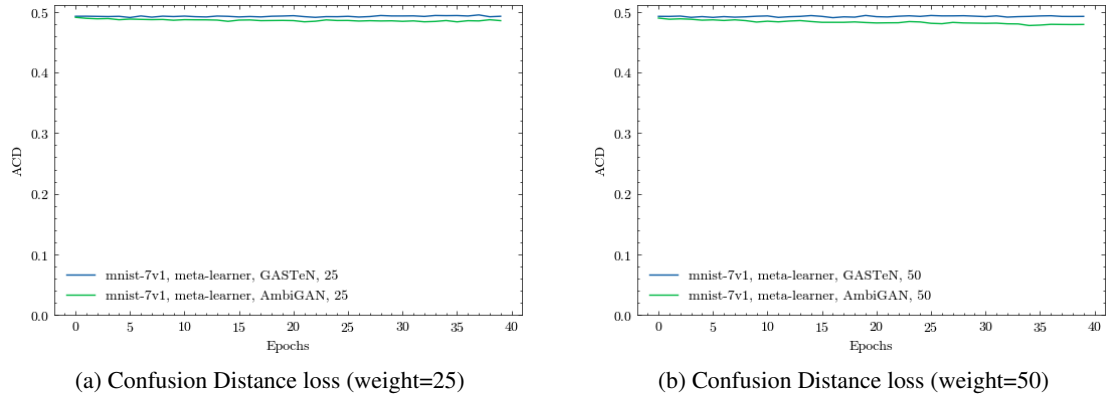


Figure E.10: Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 25 and 50

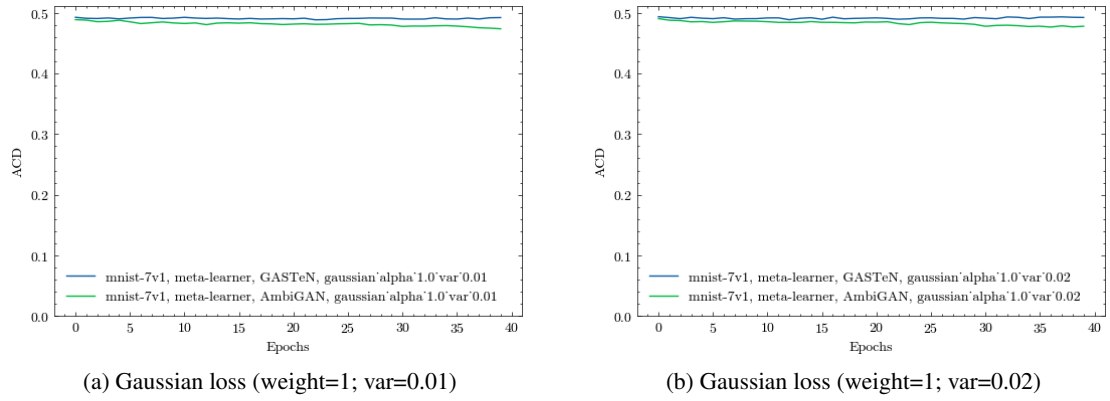


Figure E.11: Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02

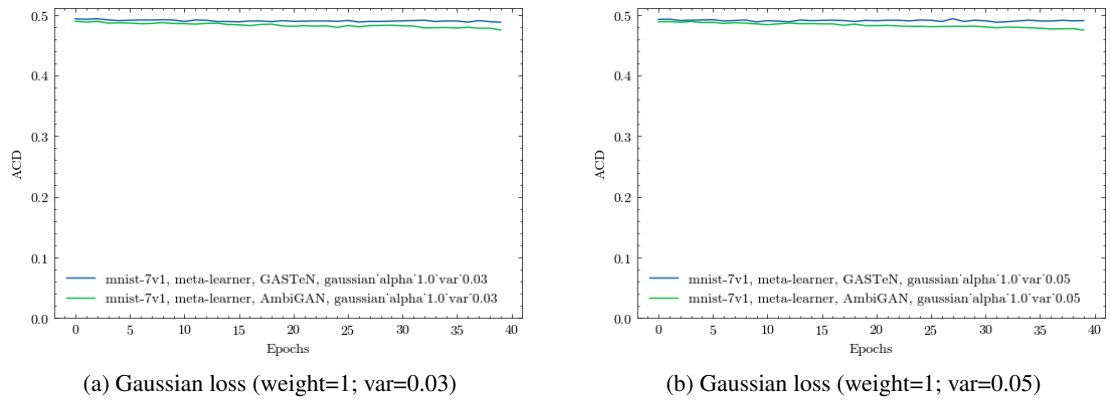


Figure E.12: Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05

E.1.4 No Output Combination Estimator

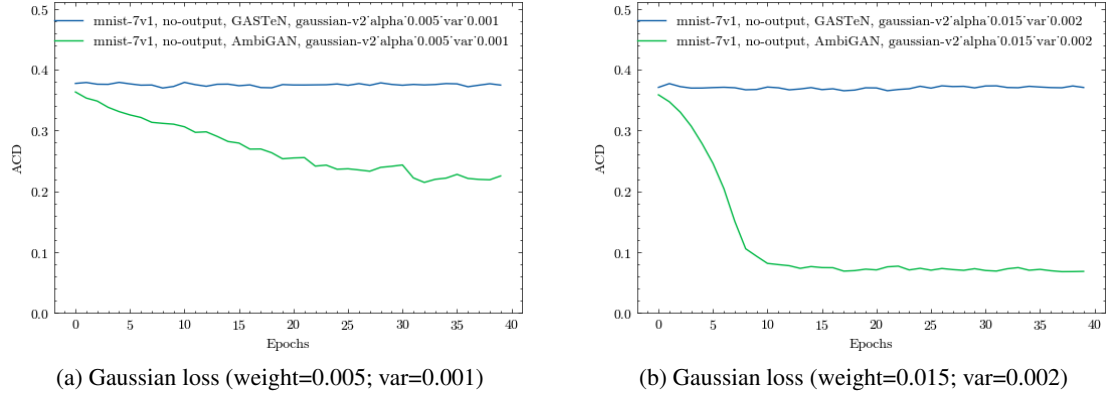


Figure E.13: Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002

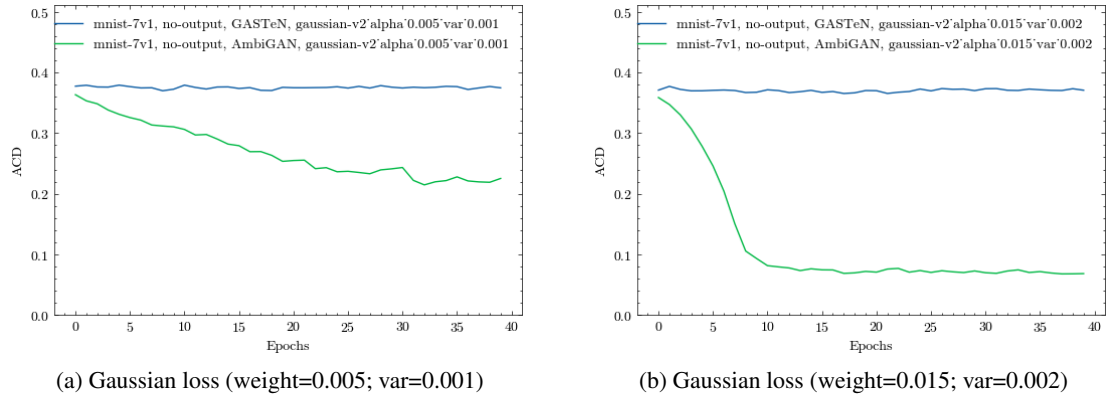


Figure E.14: Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002

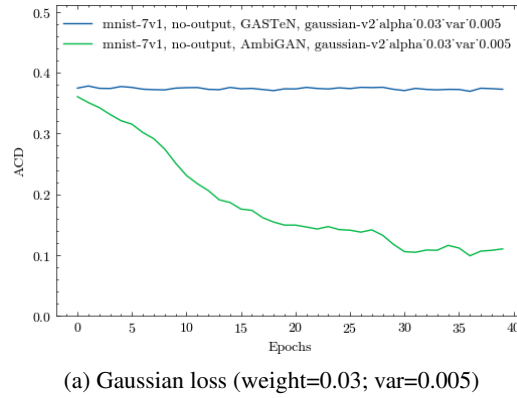


Figure E.15: Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.03 and vars 0.005

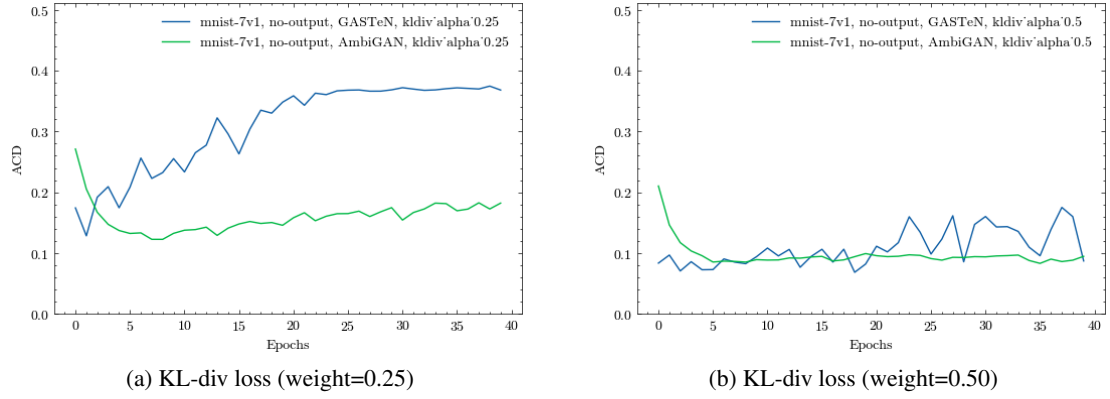


Figure E.16: Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.25 and 0.50

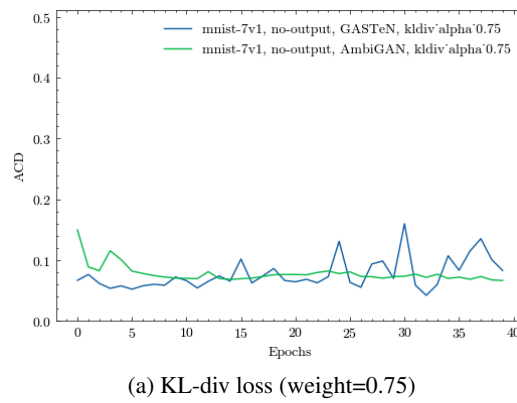


Figure E.17: Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.75

E.2 9v8 Experiment

E.2.1 Mean Output Combination Estimator

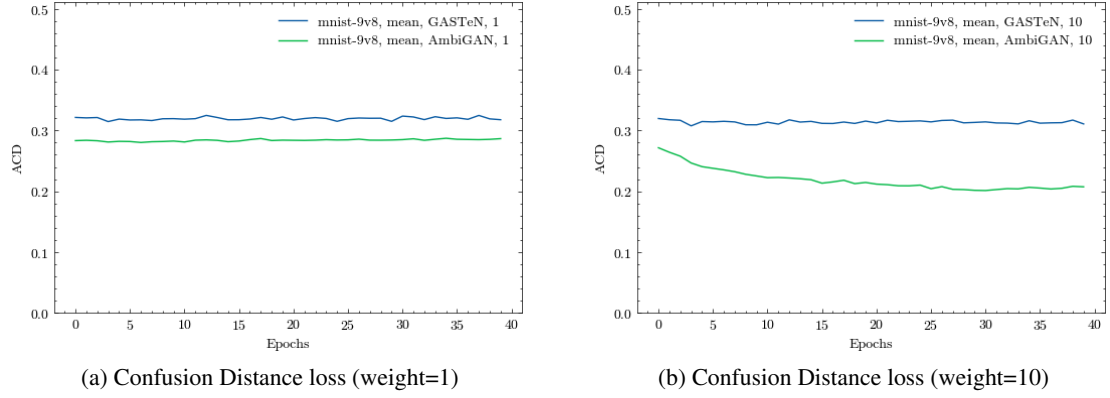


Figure E.18: Comparison of ACD during training, Mean Estimator, CD loss weights 1 and 10

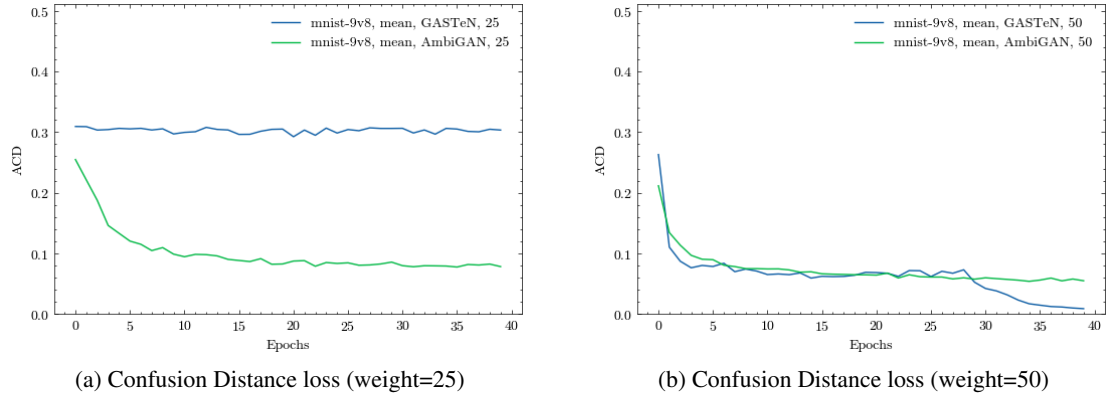


Figure E.19: Comparison of ACD during training, Mean Estimator, CD loss weights 25 and 50

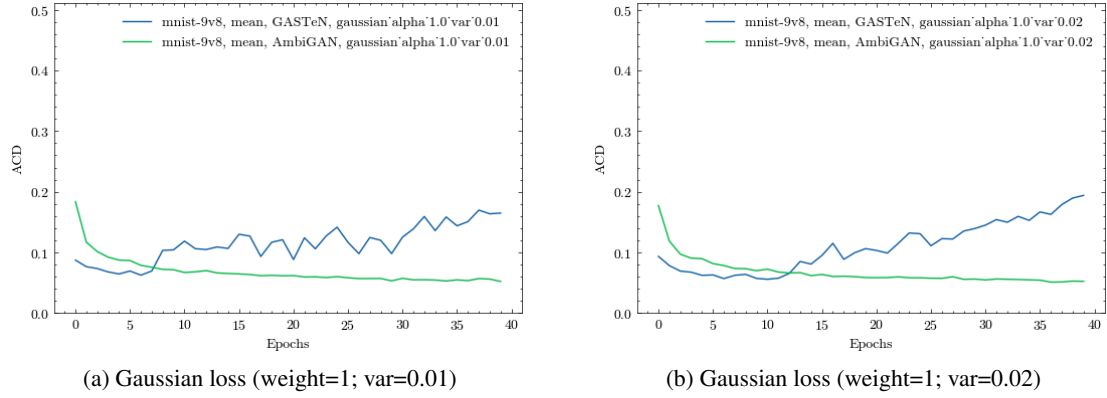


Figure E.20: Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02

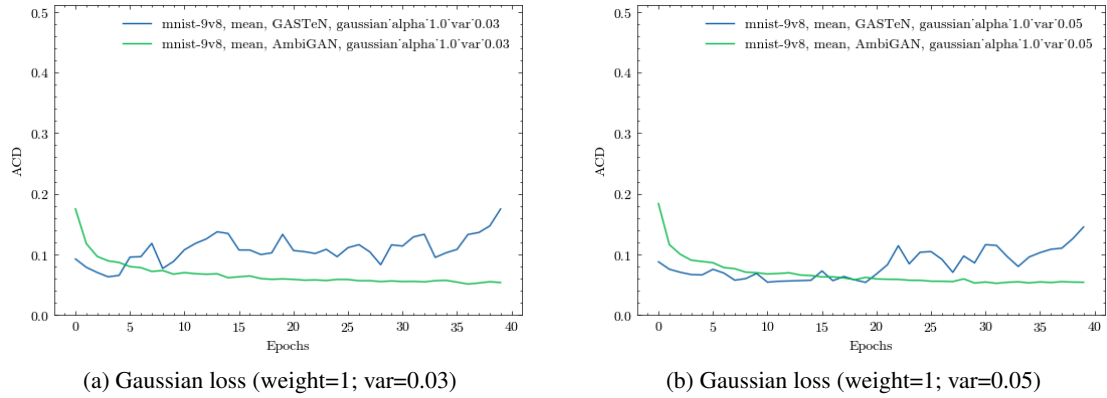


Figure E.21: Comparison of ACD during training, Mean Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05

E.2.2 Linear Output Combination Estimator

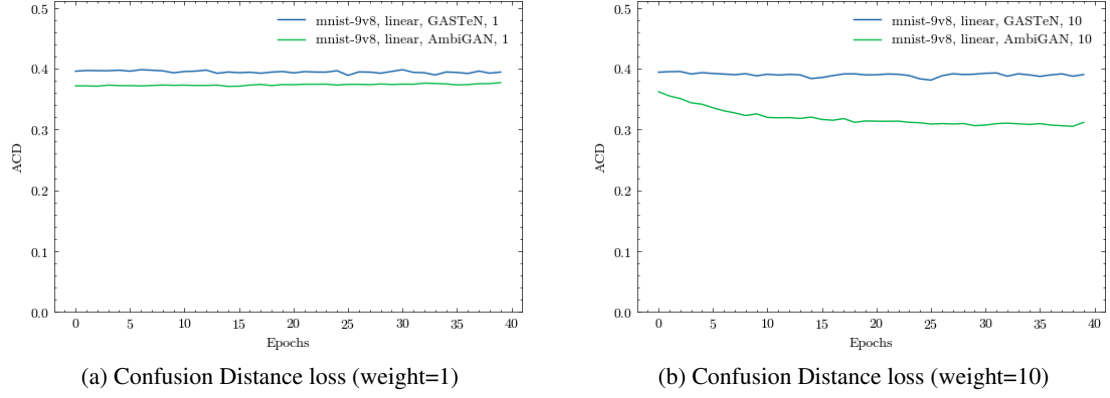


Figure E.22: Comparison of ACD during training, Linear Estimator, CD loss weights 1 and 10

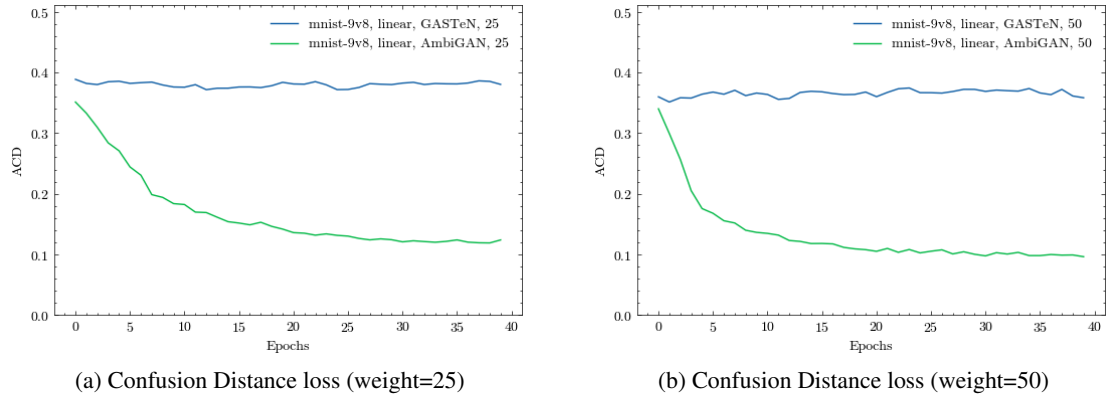


Figure E.23: Comparison of ACD during training, Linear Estimator, CD loss weights 25 and 50

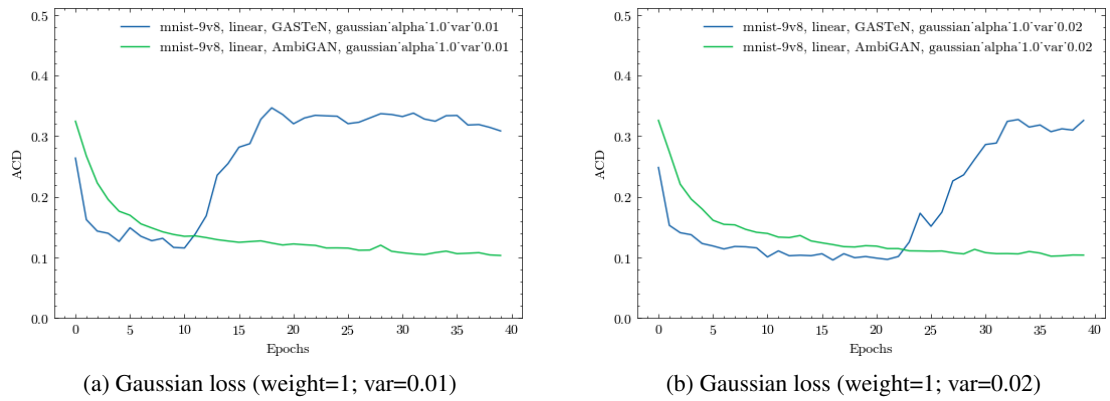


Figure E.24: Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02

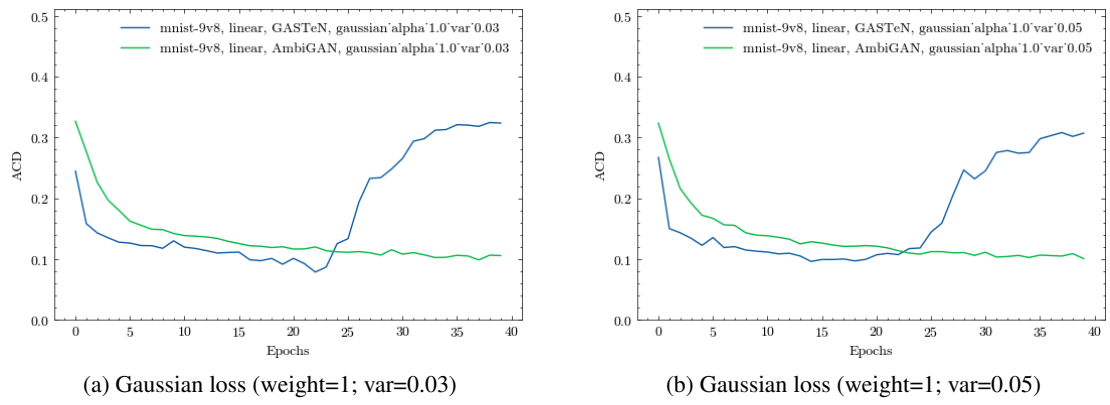


Figure E.25: Comparison of ACD during training, Linear Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05

E.2.3 Meta-Learner Output Combination Estimator

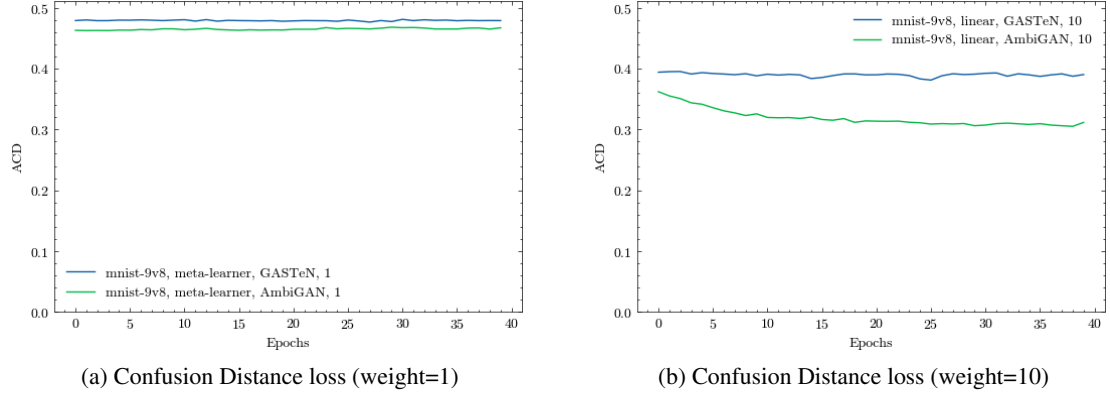


Figure E.26: Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 1 and 10

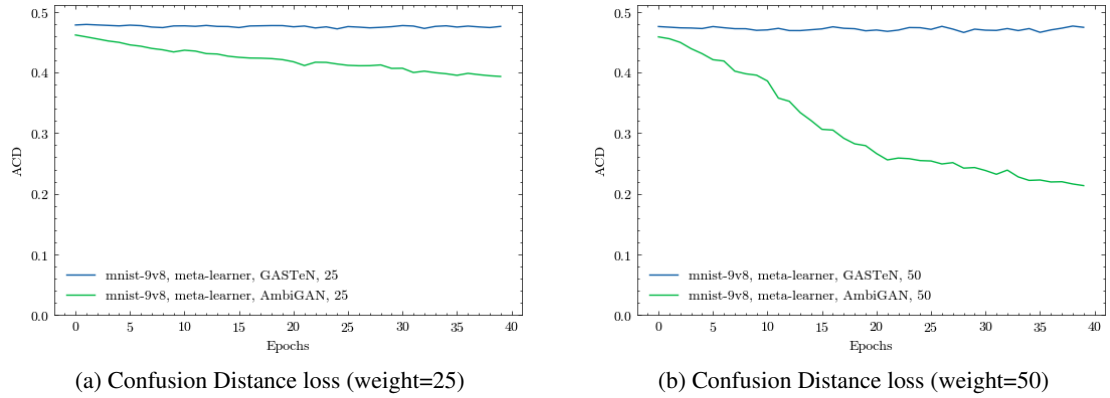


Figure E.27: Comparison of ACD during training, Meta-Learner Estimator, CD loss weights 25 and 50

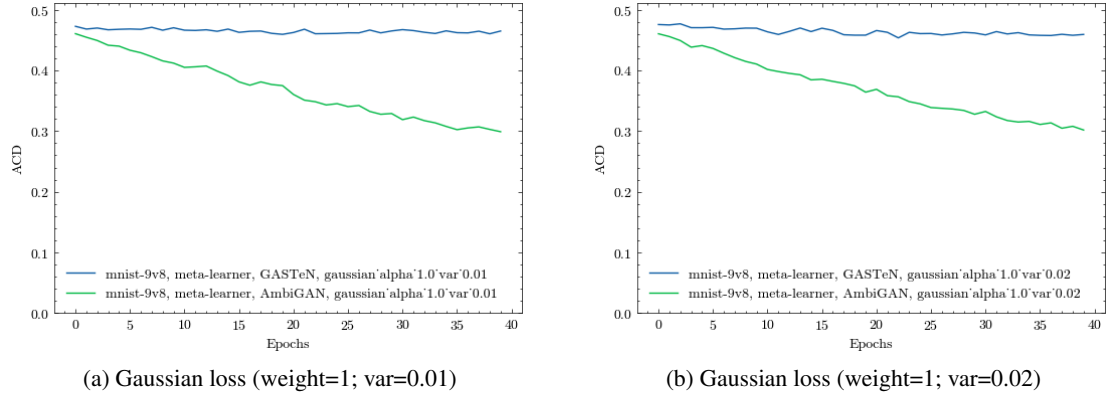


Figure E.28: Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.01 and 0.02

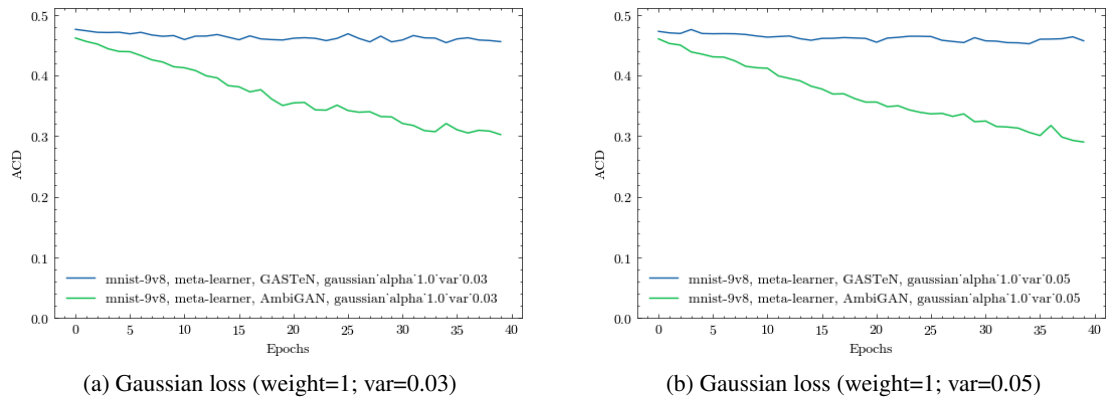


Figure E.29: Comparison of ACD during training, Meta-Learner Estimator, Gaussian loss weight 1 and vars 0.03 and 0.05

E.2.4 No Output Combination Estimator

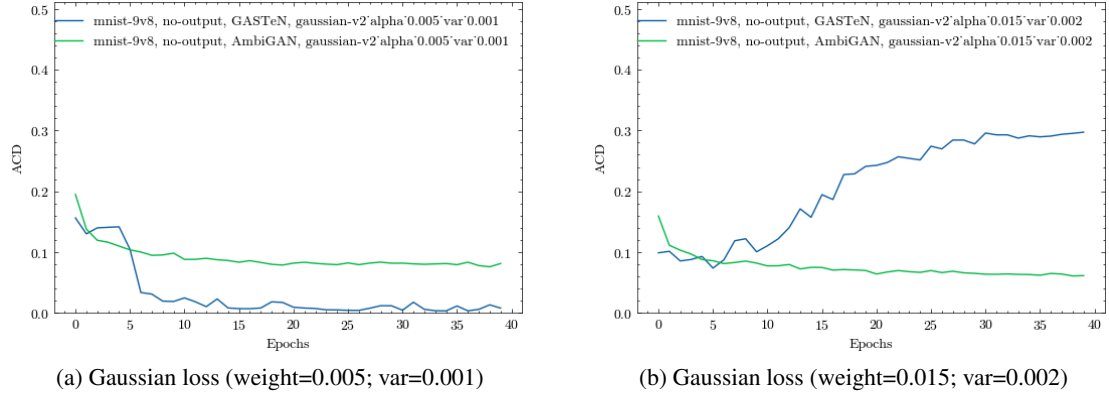


Figure E.30: Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002

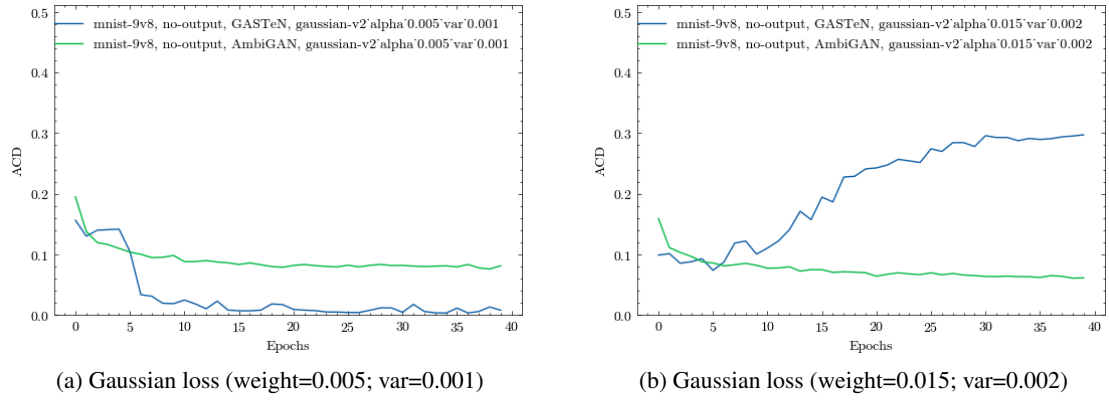


Figure E.31: Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.005 and vars 0.001 and 0.002

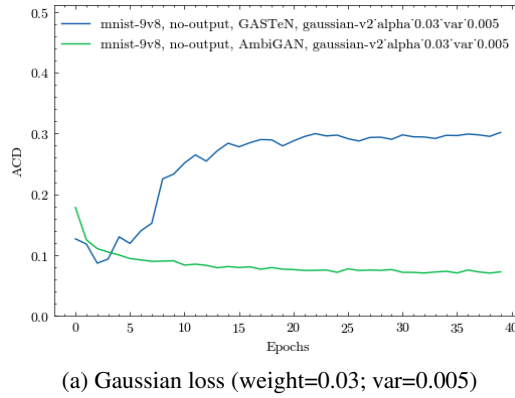


Figure E.32: Comparison of ACD during training, No Output Estimator, Gaussian loss weights 0.03 and vars 0.005

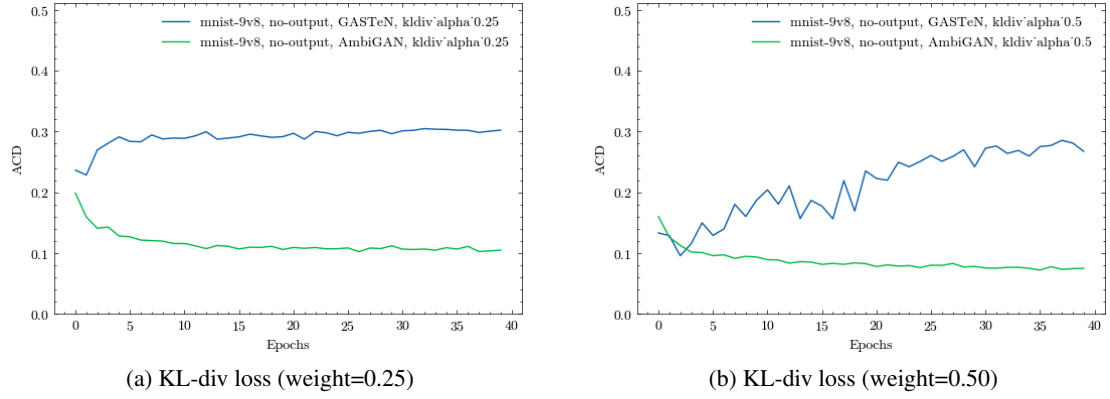


Figure E.33: Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.25 and 0.50

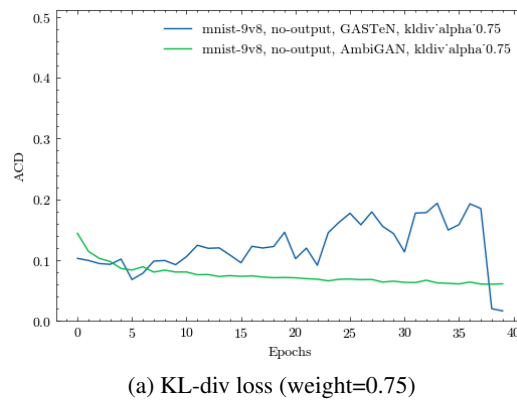


Figure E.34: Comparison of ACD during training, No Output Estimator, KL-div loss weights 0.75

Appendix F

F.1 7v1 Experiment

F.1.1 Mean Output Combination Estimator

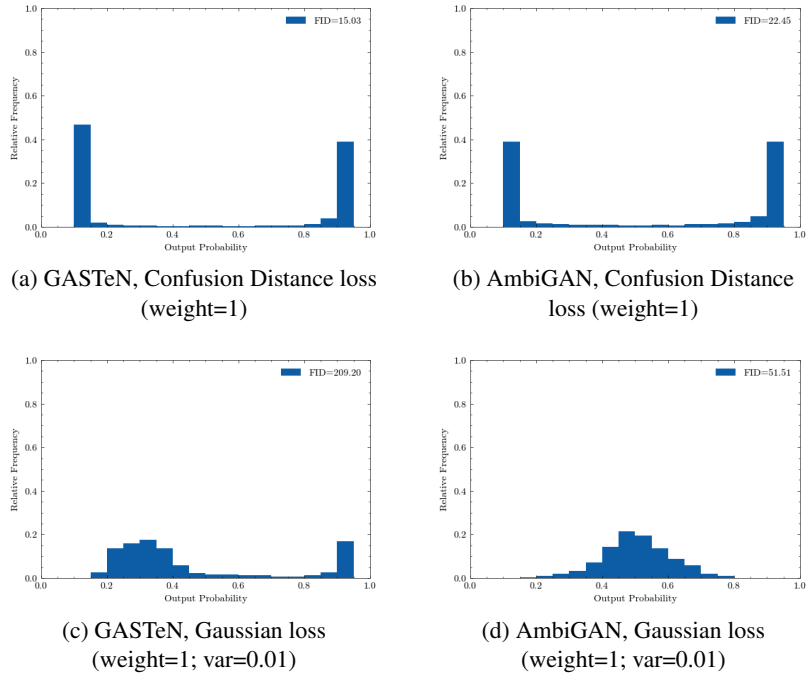


Figure F.1: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01

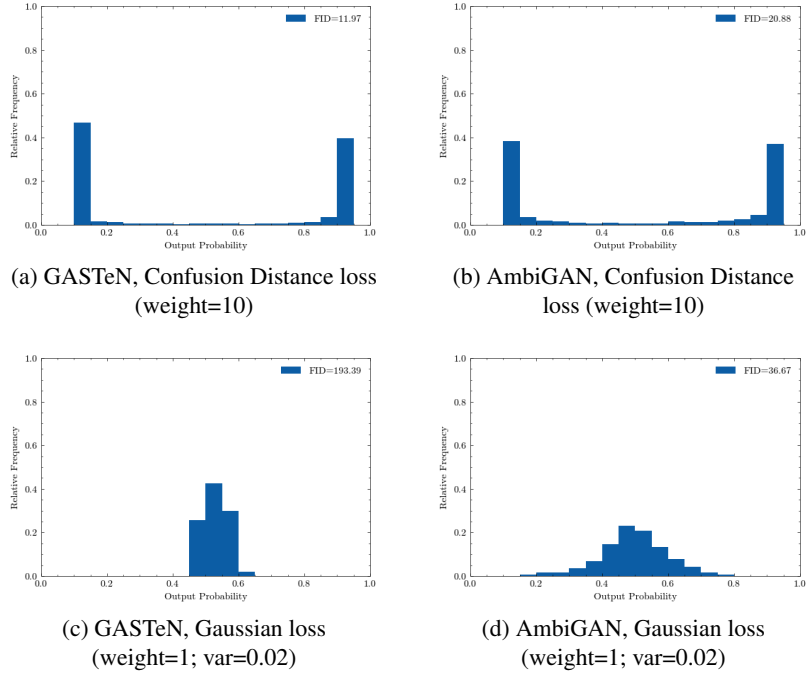


Figure F.2: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02

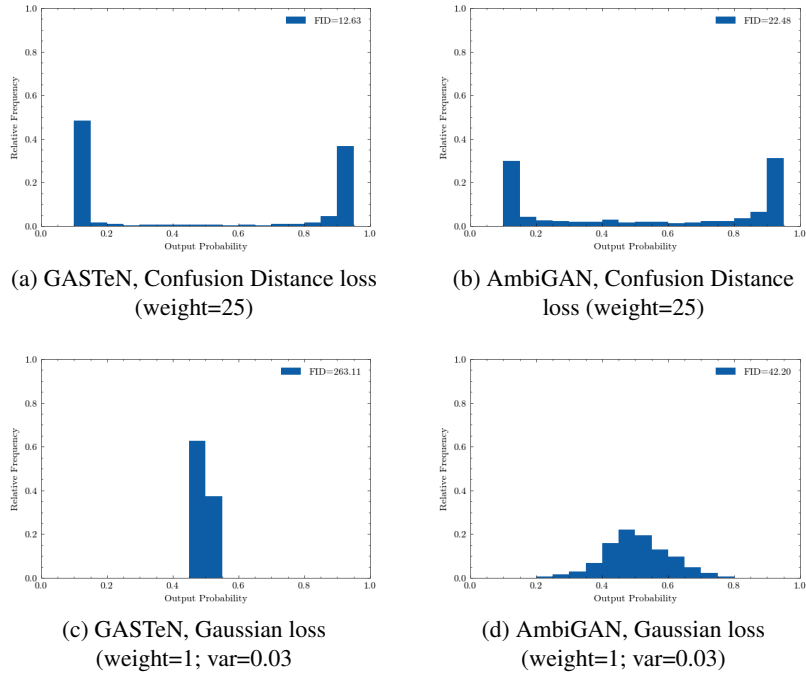


Figure F.3: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03

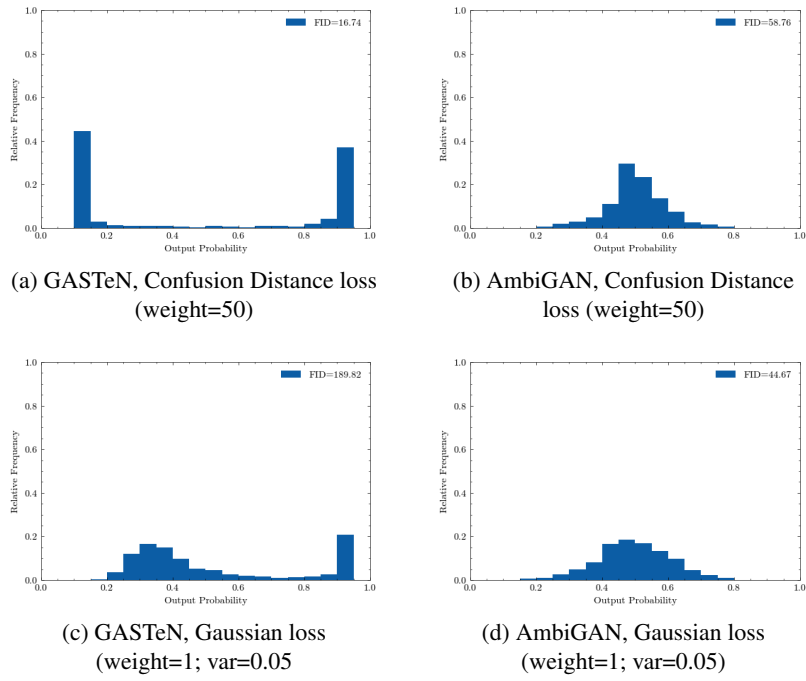


Figure F.4: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05

F.1.2 Linear Output Combination Estimator

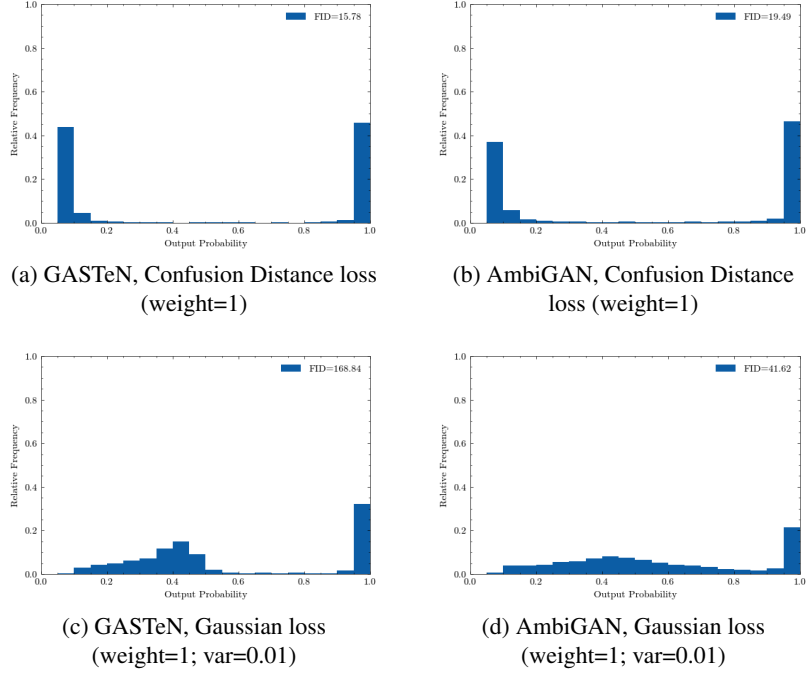


Figure F.5: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01

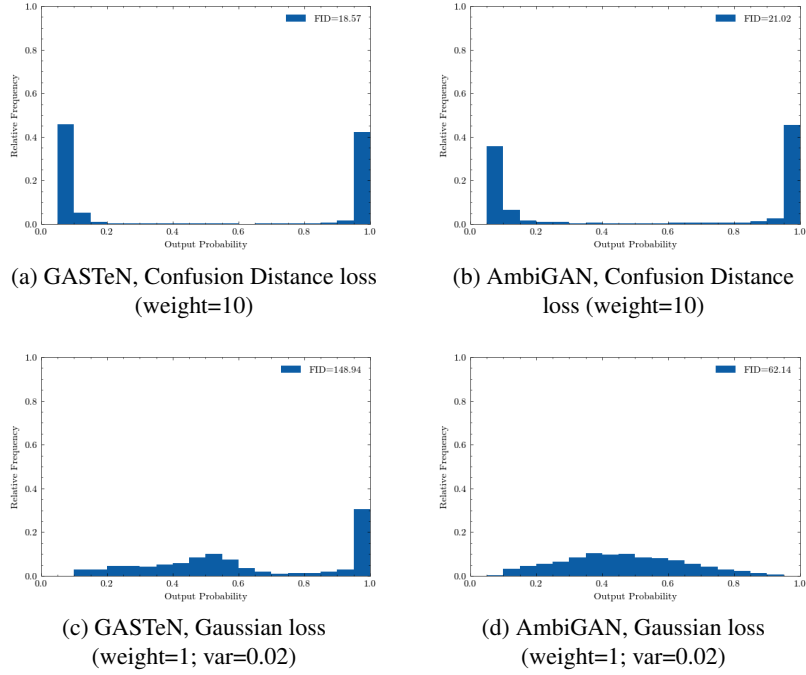


Figure F.6: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02

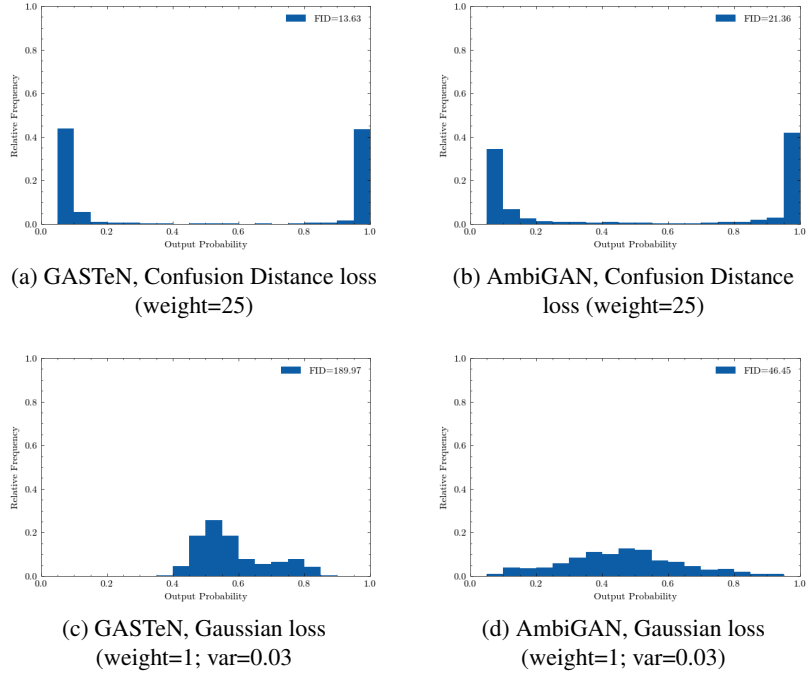


Figure F.7: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03

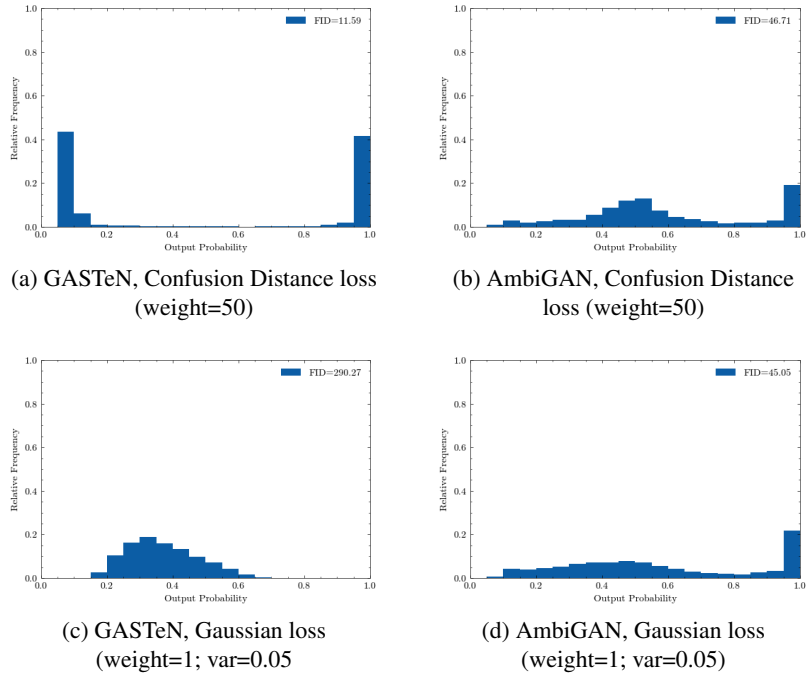


Figure F.8: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05

F.1.3 Meta-Learner Output Combination Estimator

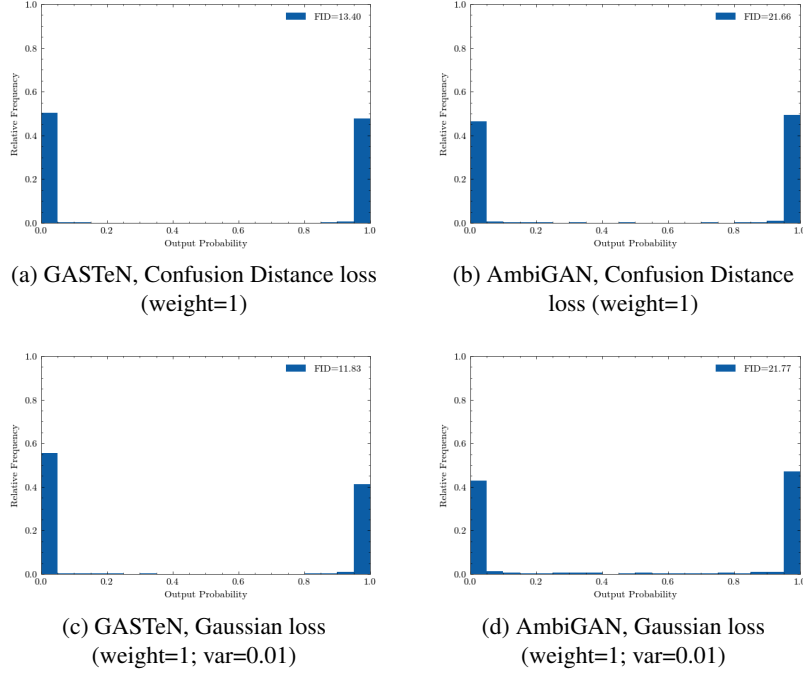


Figure F.9: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01

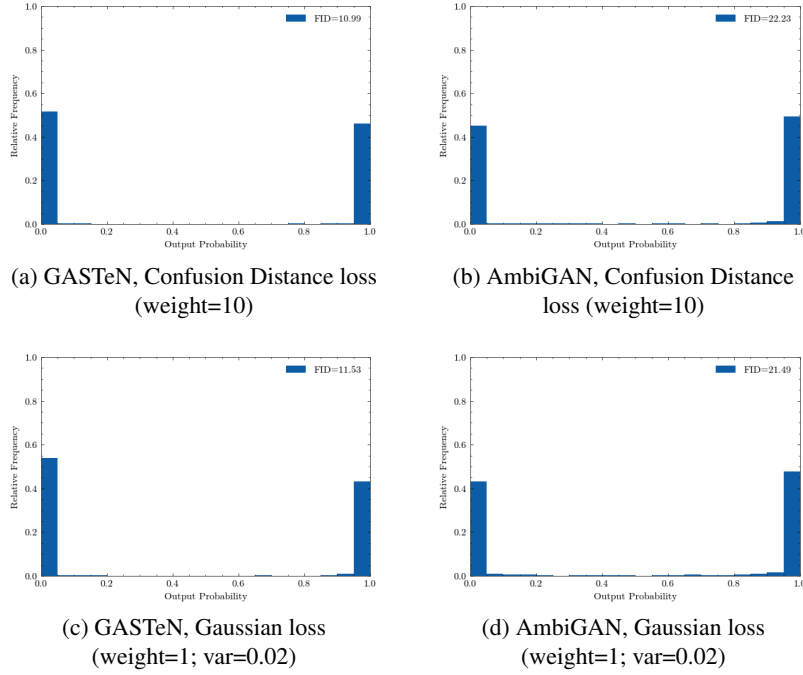


Figure F.10: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02

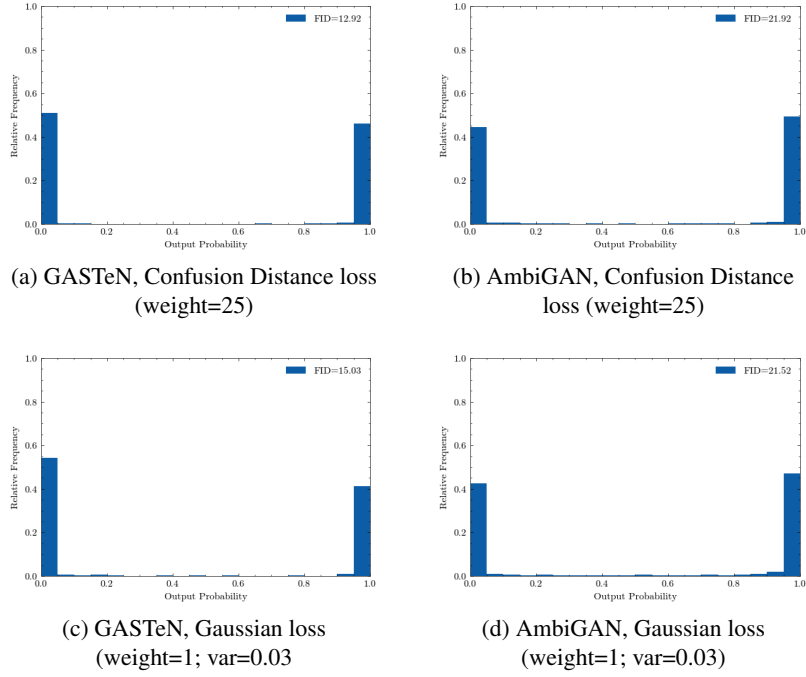


Figure F.11: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03

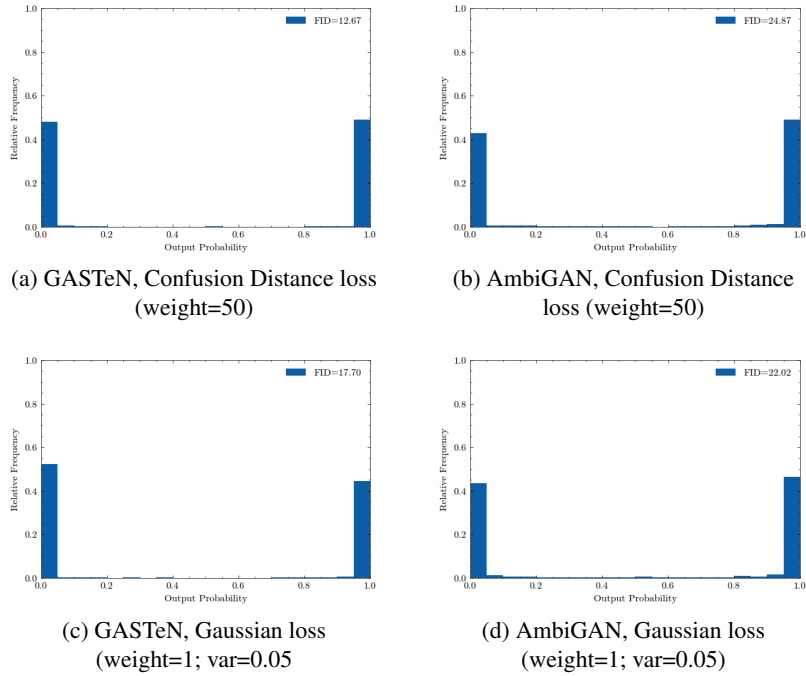


Figure F.12: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05

F.1.4 No Output Combination Estimator

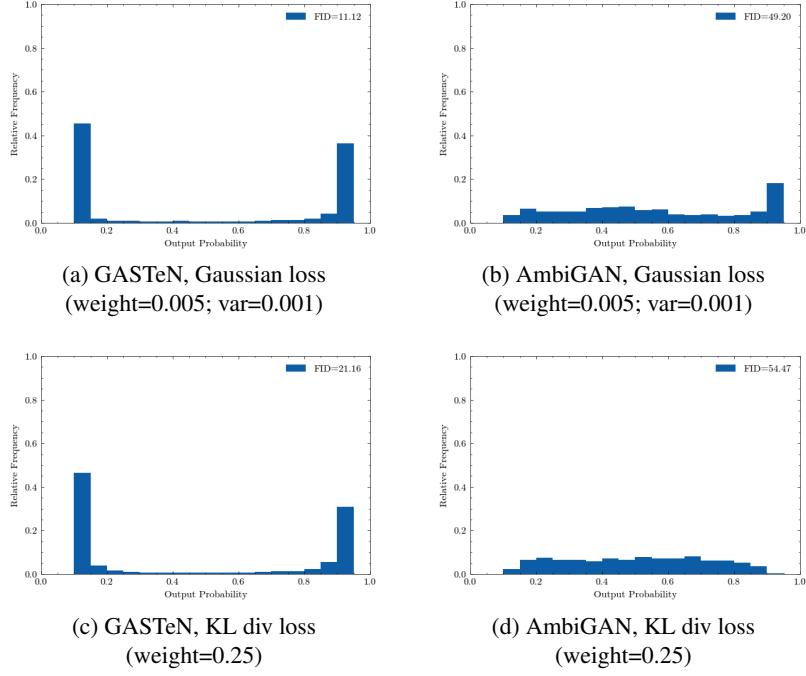


Figure F.13: Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.005 and var 0.001 and KL div loss weight 0.25

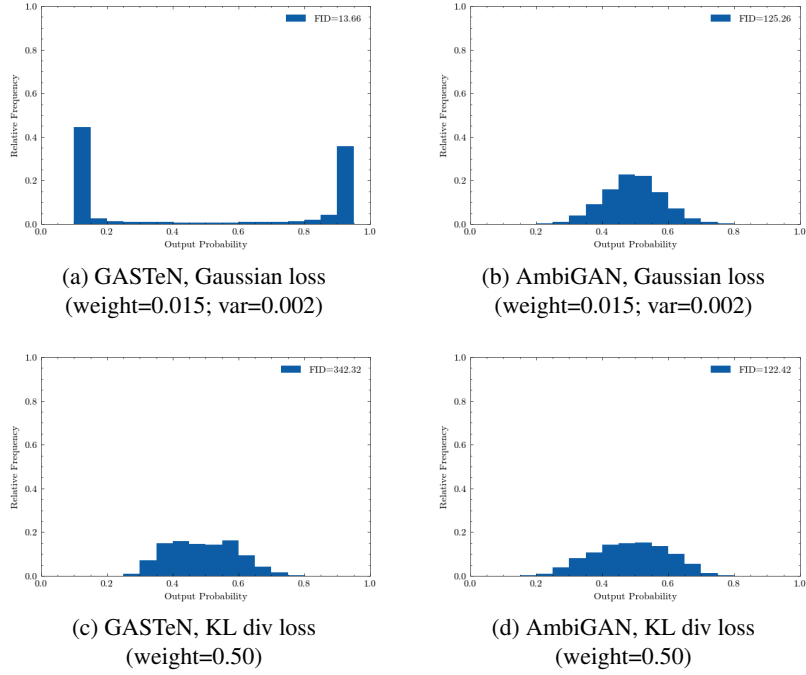


Figure F.14: Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.015 and var 0.002 and KL div loss weight 0.50

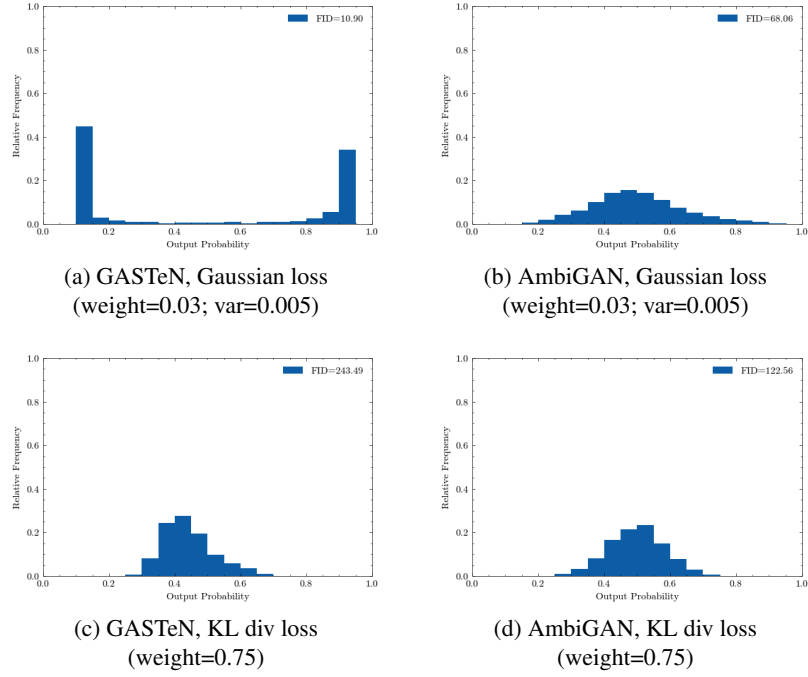


Figure F.15: Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight 0.03 and var 0.005 and KL div loss weight 0.75

F.2 9v8 Experiment

F.2.1 Mean Output Combination Estimator

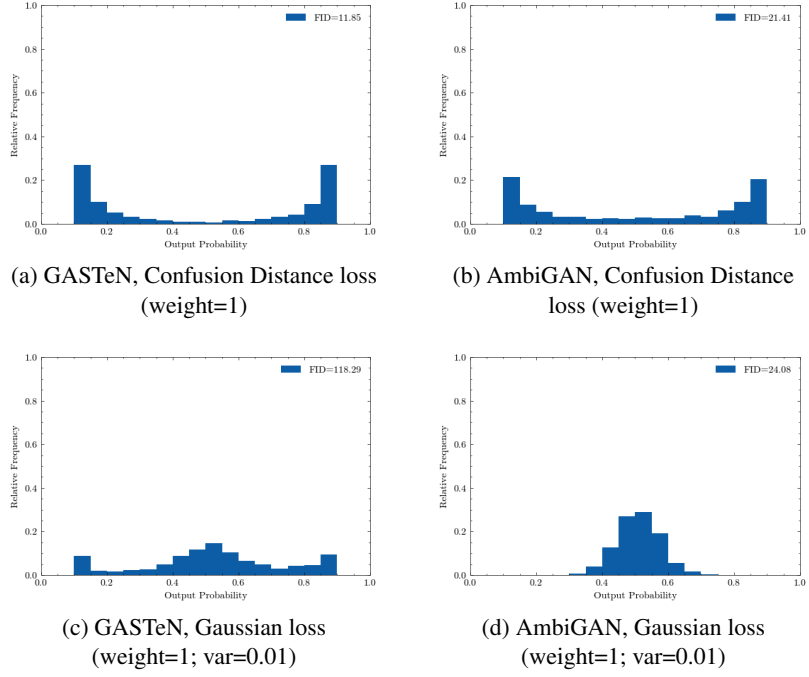


Figure F.16: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01

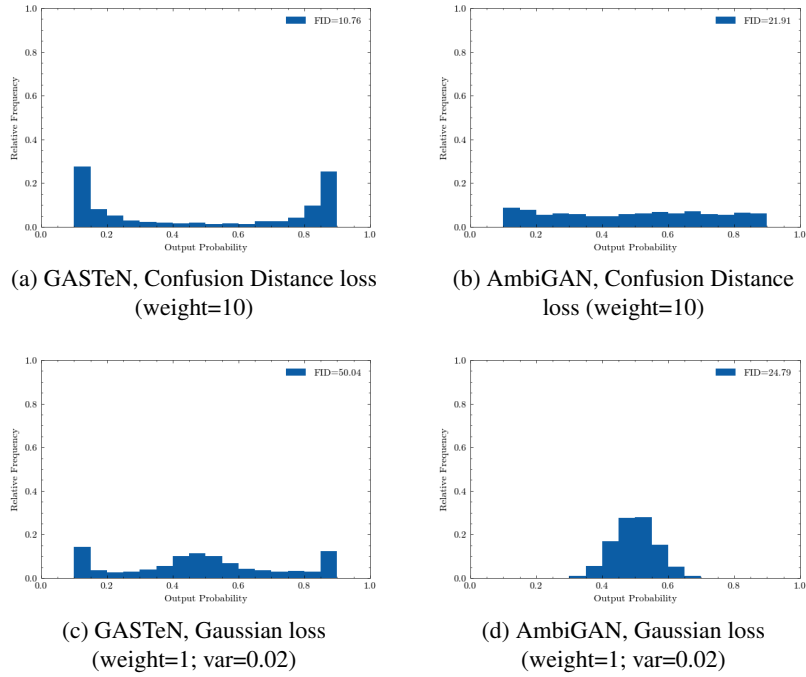


Figure F.17: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02

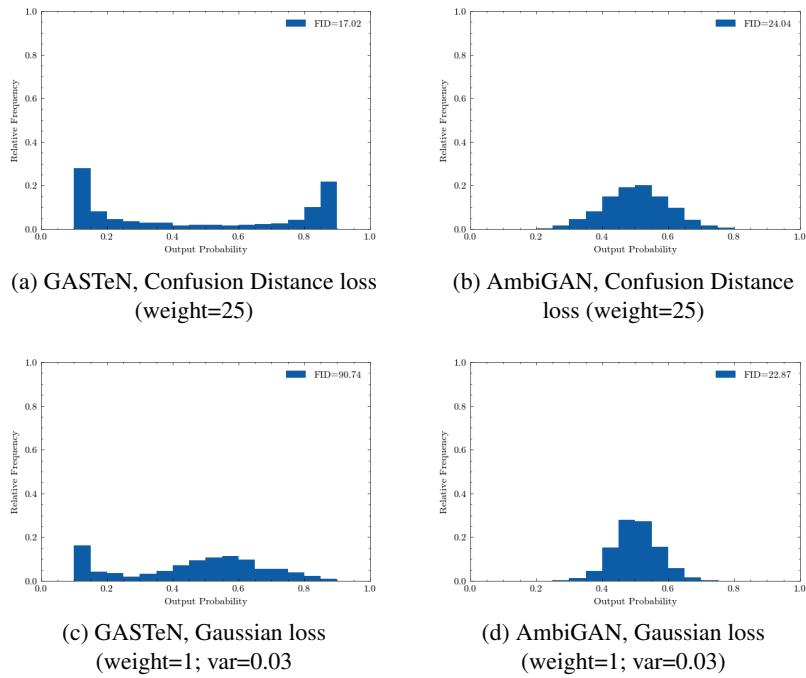


Figure F.18: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03

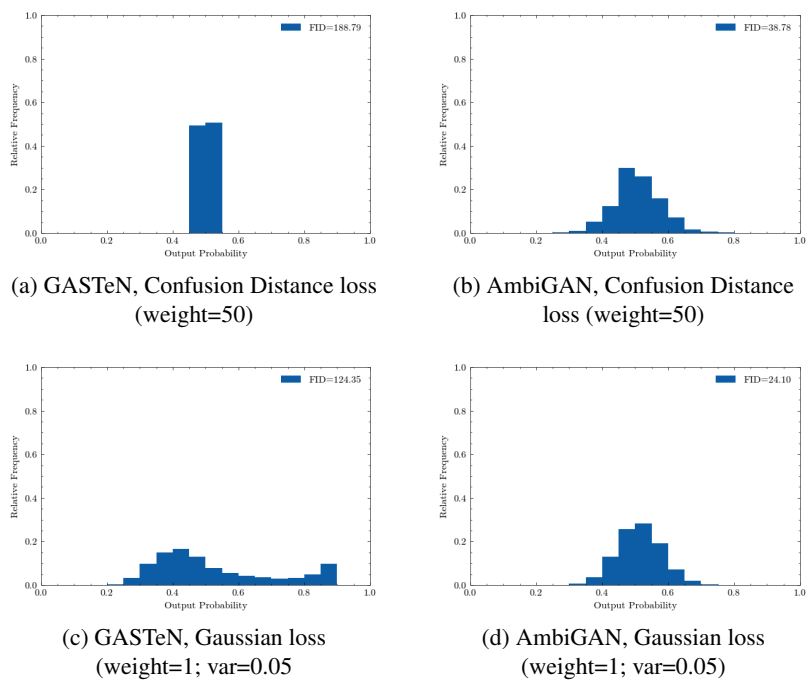


Figure F.19: Histogram comparison GASTeN and AmbiGAN architecture, Mean Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05

F.2.2 Linear Output Combination Estimator

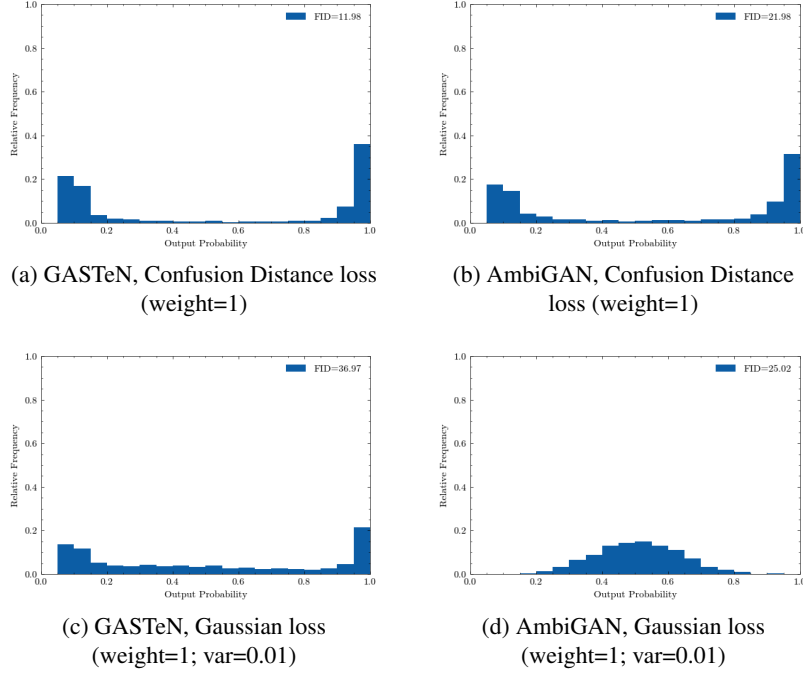


Figure F.20: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01

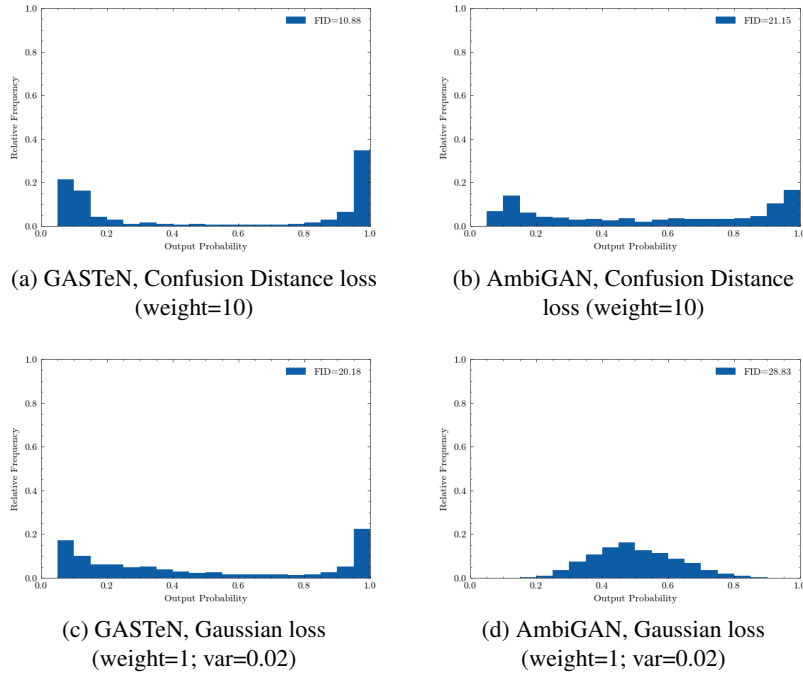


Figure F.21: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02

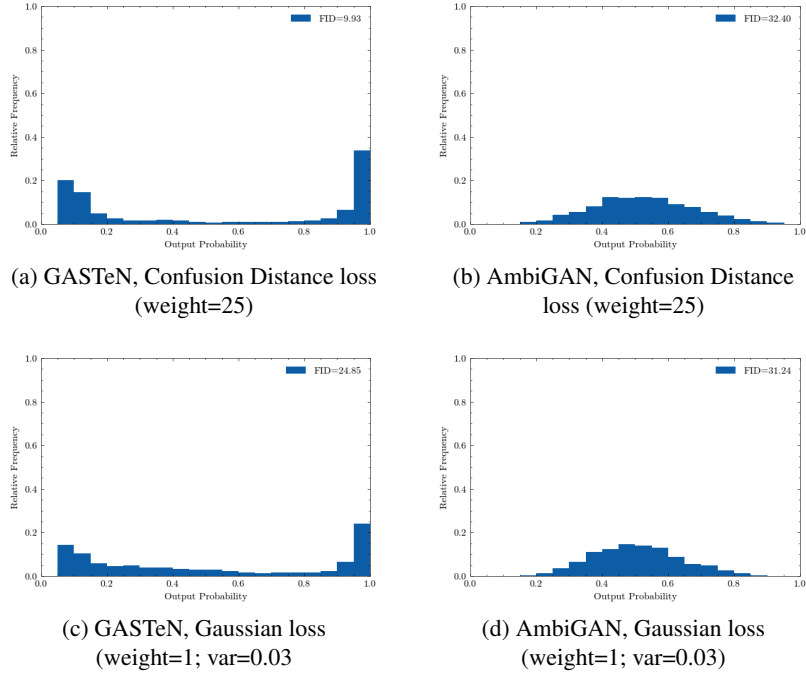


Figure F.22: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03

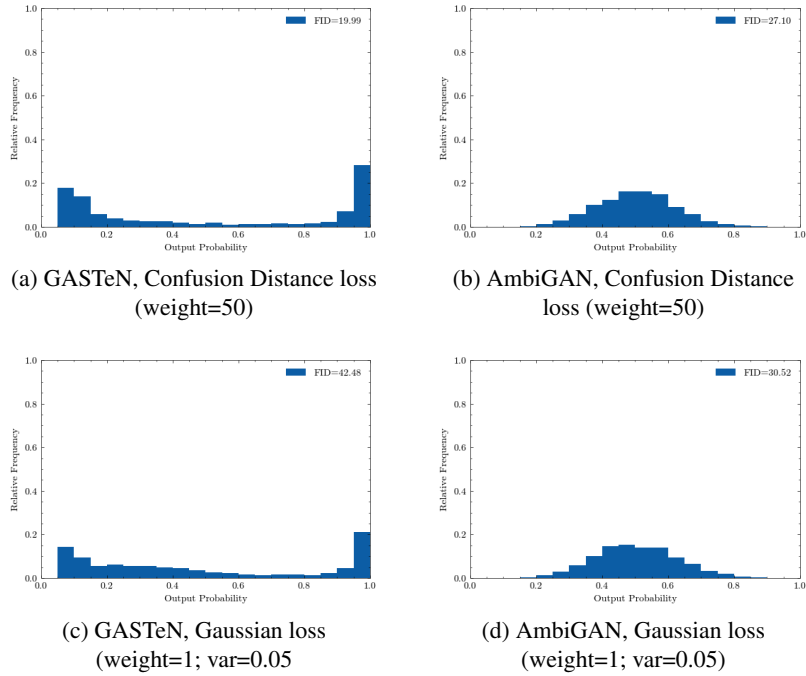


Figure F.23: Histogram comparison GASTeN and AmbiGAN architecture, Linear Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05

F.2.3 Meta-Learner Output Combination Estimator

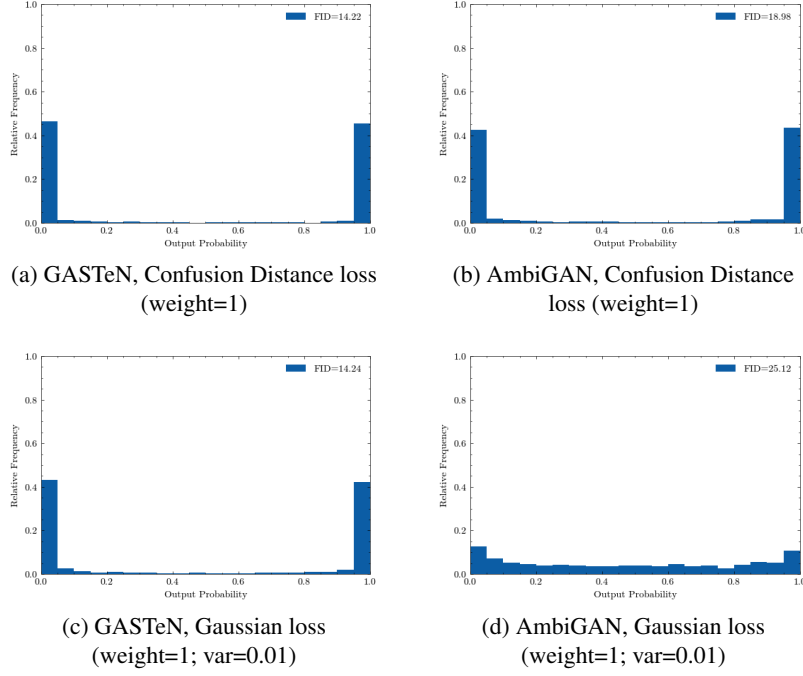


Figure F.24: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 1 and Gaussian loss weight 1 and var 0.01

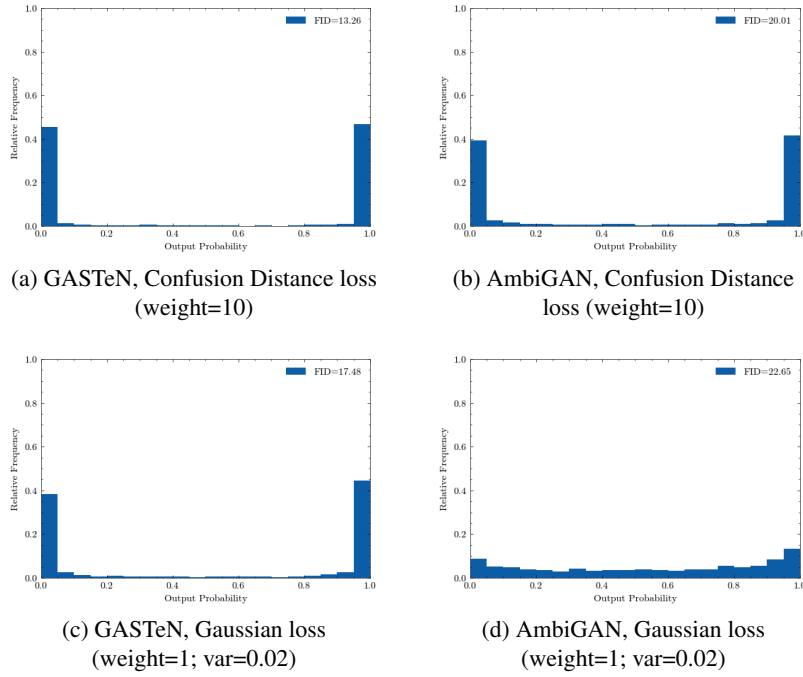


Figure F.25: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 10 and Gaussian loss weight 1 and var 0.02

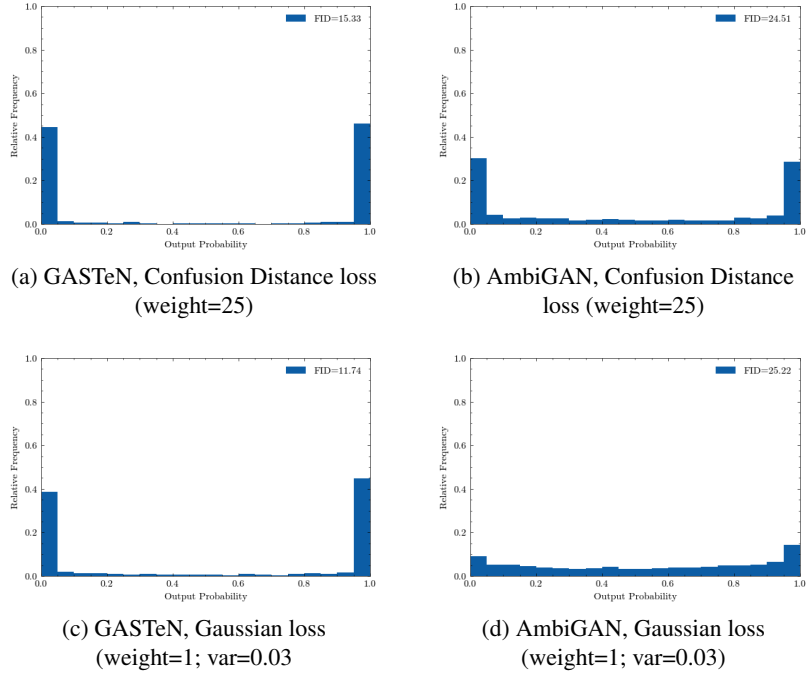


Figure F.26: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 25 and Gaussian loss weight 1 and var 0.03

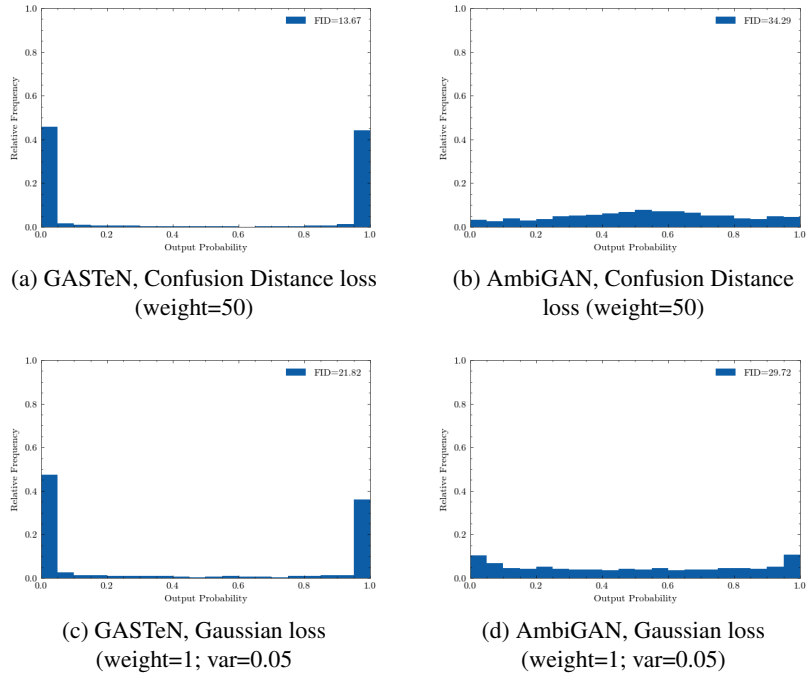


Figure F.27: Histogram comparison GASTeN and AmbiGAN architecture, Meta-Learner Estimator, CD loss weight 50 and Gaussian loss weight 1 and var 0.05

F.2.4 No Output Combination Estimator

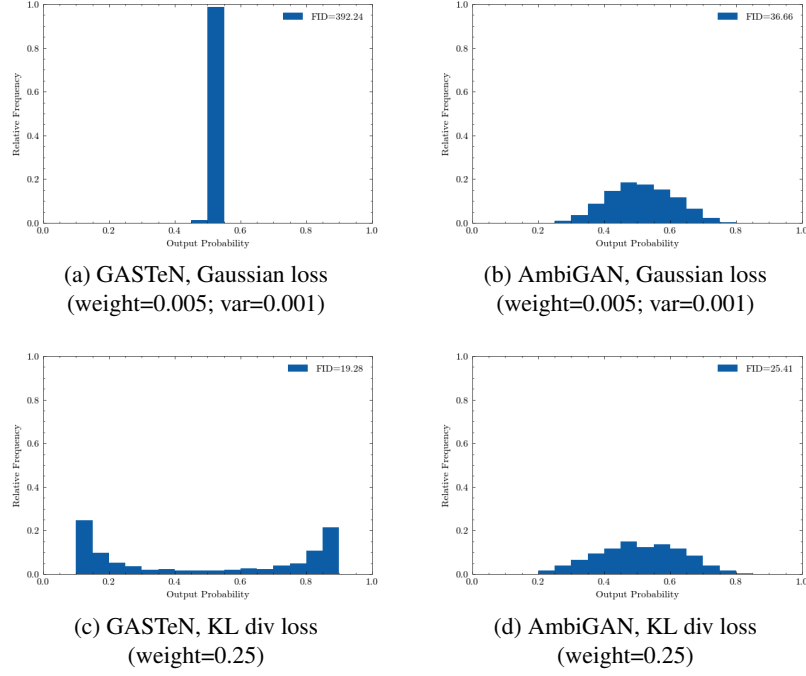


Figure F.28: Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.005 and var 0.001 and KL div loss weight 0.25

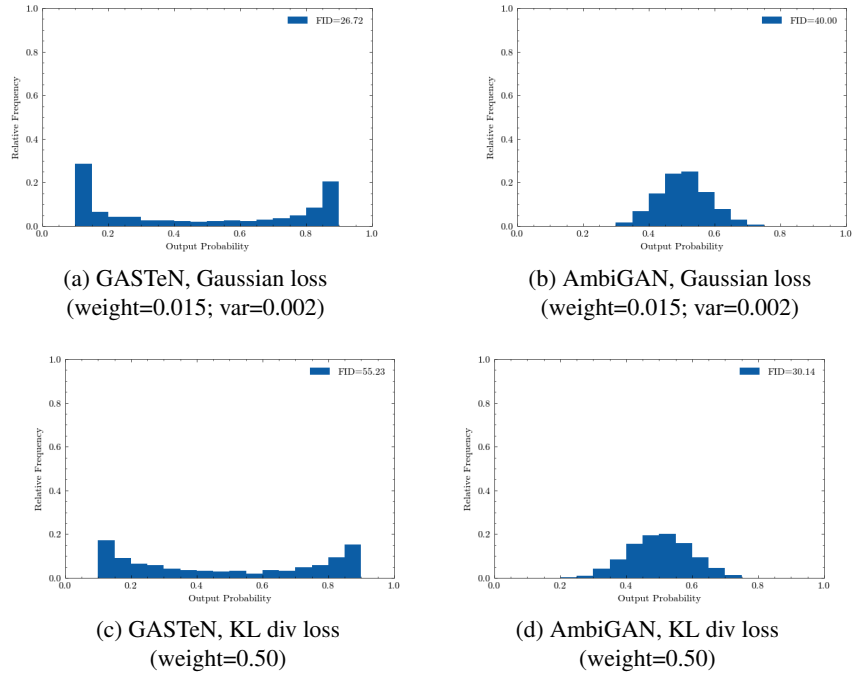


Figure F.29: Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight weight 0.015 and var 0.002 and KL div loss weight 0.50

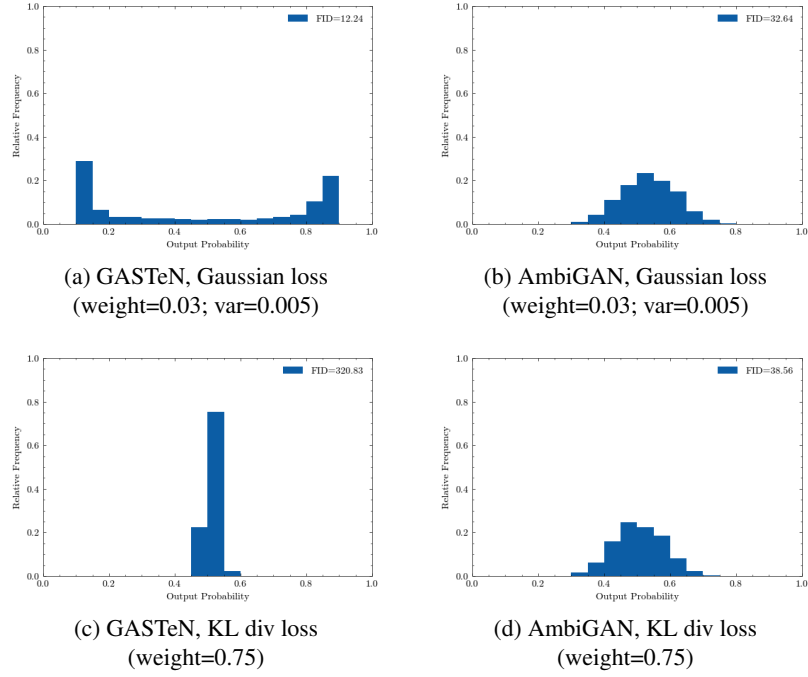
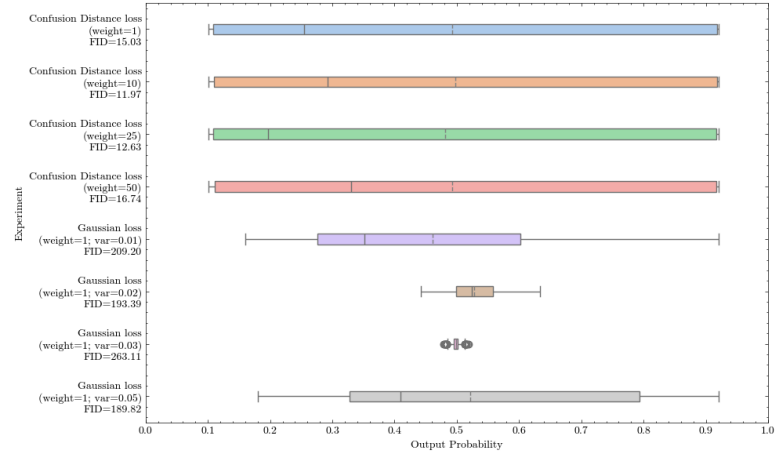


Figure F.30: Histogram comparison GASTeN and AmbiGAN architecture, No Output Estimator, Gaussian loss weight 0.03 and var 0.005 and KL div loss weight 0.75

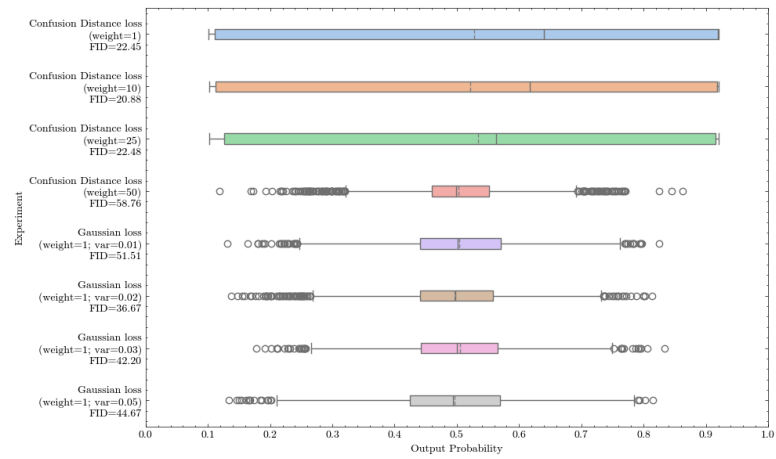
Appendix G

G.1 7v1 Experiment

G.1.1 Mean Output Combination Estimator



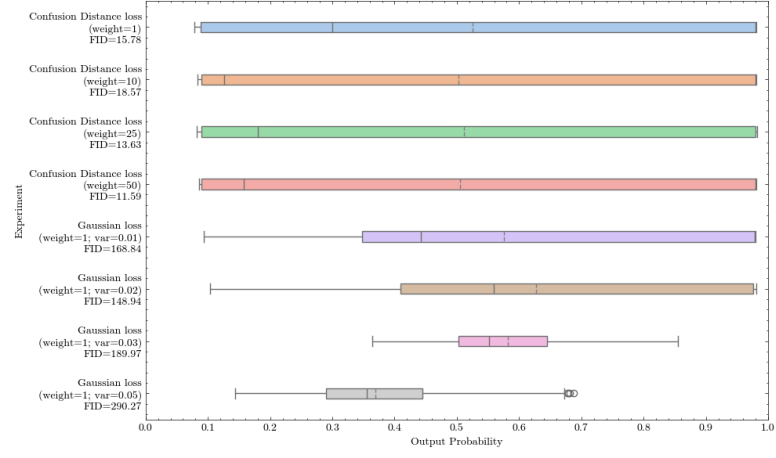
(a) GASTeN Architecture



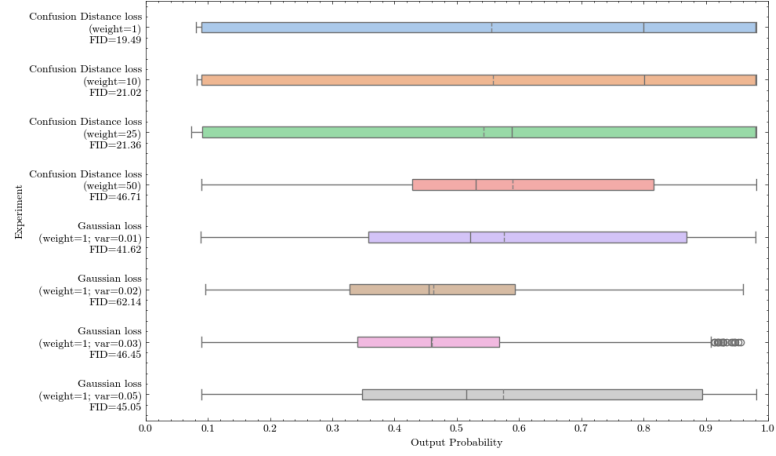
(b) AmbiGAN Architecture

Figure G.1: Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, Mean Estimator, all hyperparams variations

G.1.2 Linear Output Combination Estimator



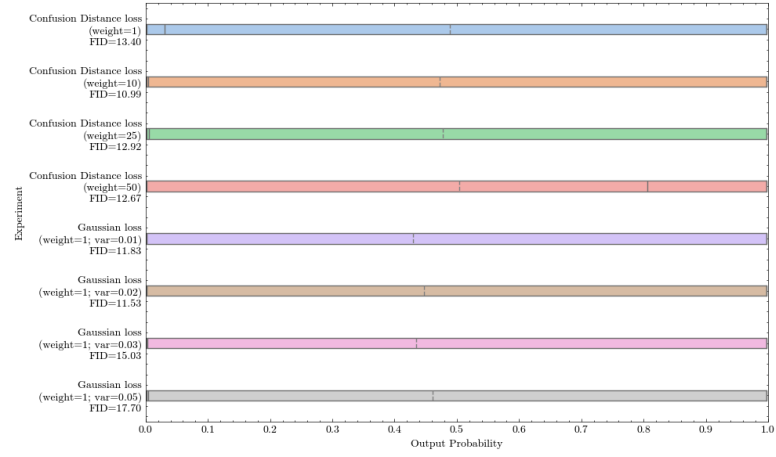
(a) GASTeN Architecture



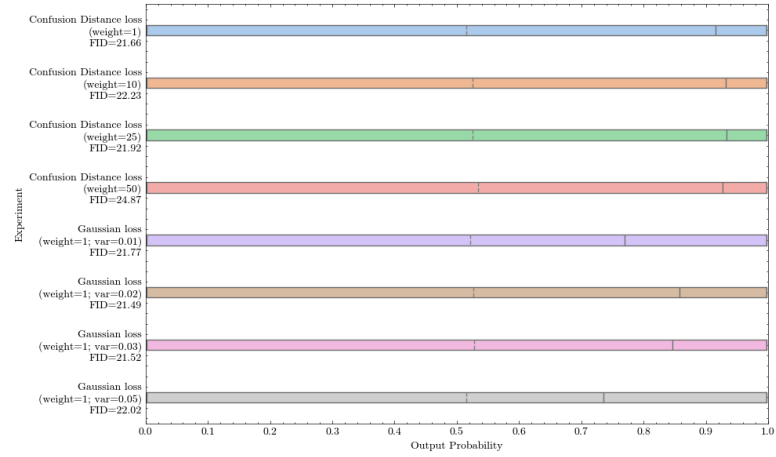
(b) AmbiGAN Architecture

Figure G.2: Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, Linear Estimator, all hyperparams variations

G.1.3 Meta-Learner Output Combination Estimator



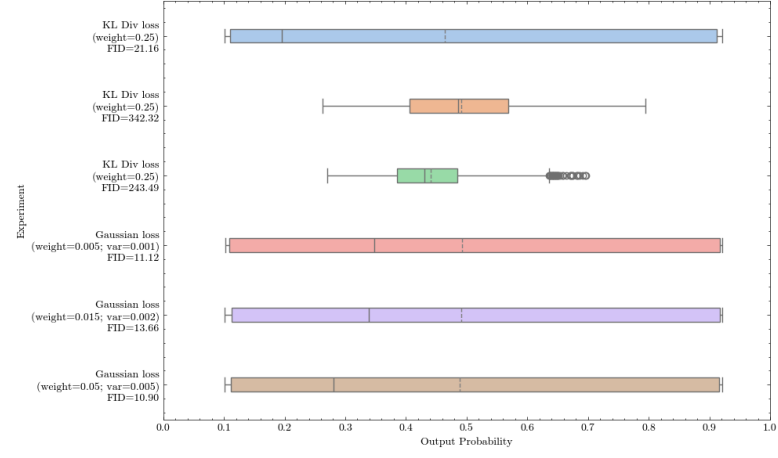
(a) GASTeN Architecture



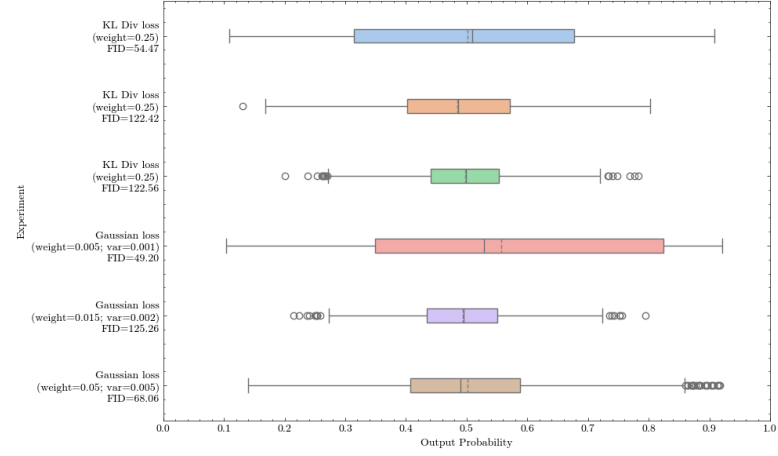
(b) AmbiGAN Architecture

Figure G.3: Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, Meta-Learner Estimator, all hyperparams variations

G.1.4 No Output Combination Estimator



(a) GASTeN Architecture

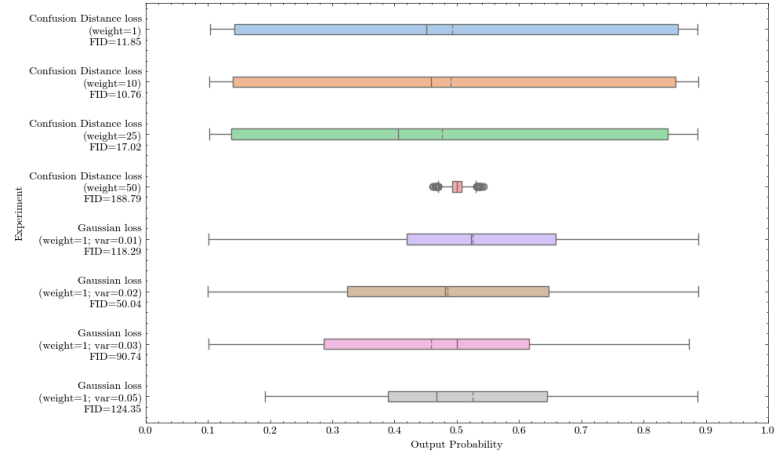


(b) AmbiGAN Architecture

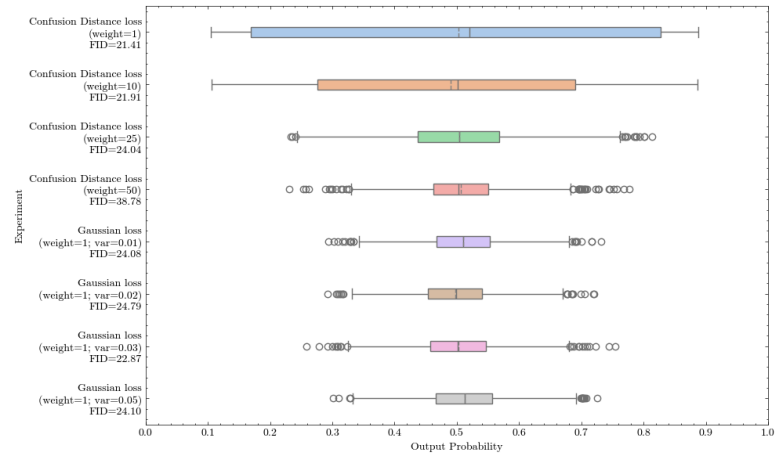
Figure G.4: Boxplot comparison 7v1, GASTeN and AmbiGAN architecture, No Output Estimator, all hyperparams variations

G.2 9v8 Experiment

G.2.1 Mean Output Combination Estimator



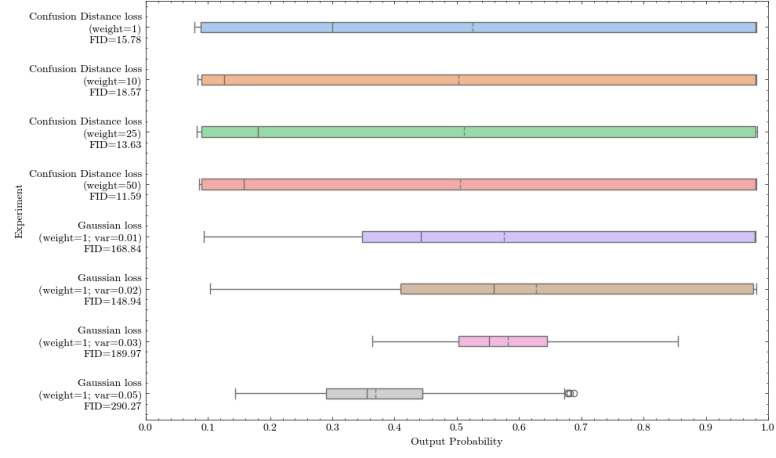
(a) GASTeN Architecture



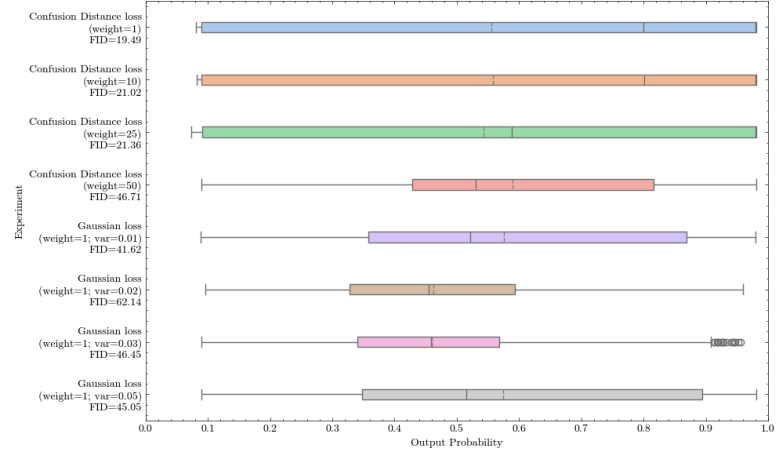
(b) AmbiGAN Architecture

Figure G.5: Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, Mean Estimator, all hyperparams variations

G.2.2 Linear Output Combination Estimator



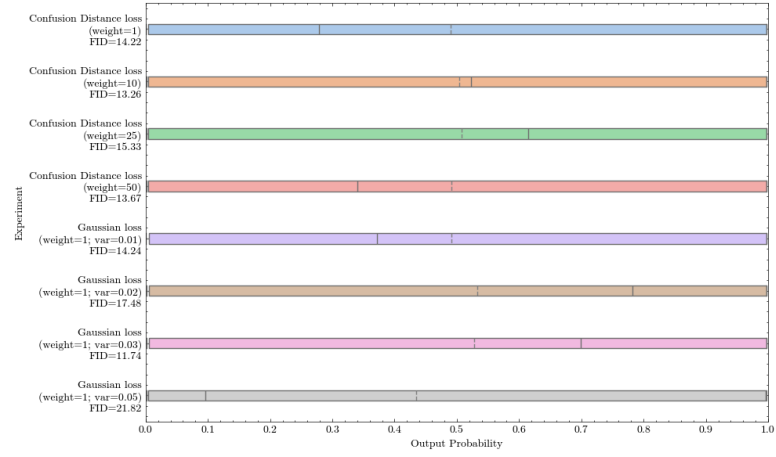
(a) GASTeN Architecture



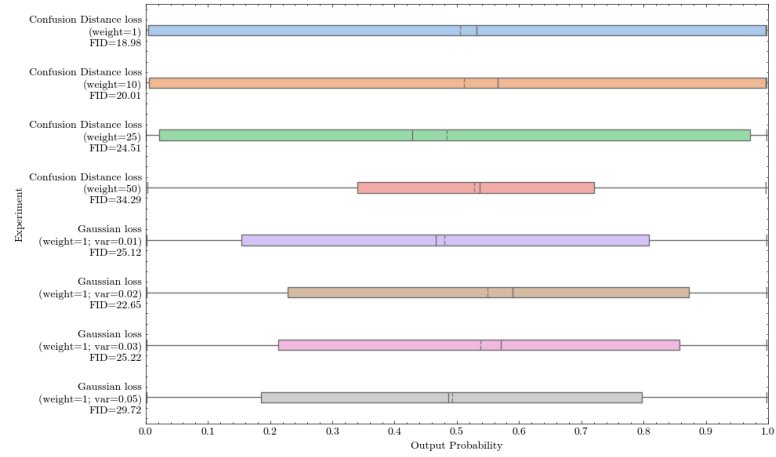
(b) AmbiGAN Architecture

Figure G.6: Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, Linear Estimator, all hyperparams variations

G.2.3 Meta-Learner Output Combination Estimator



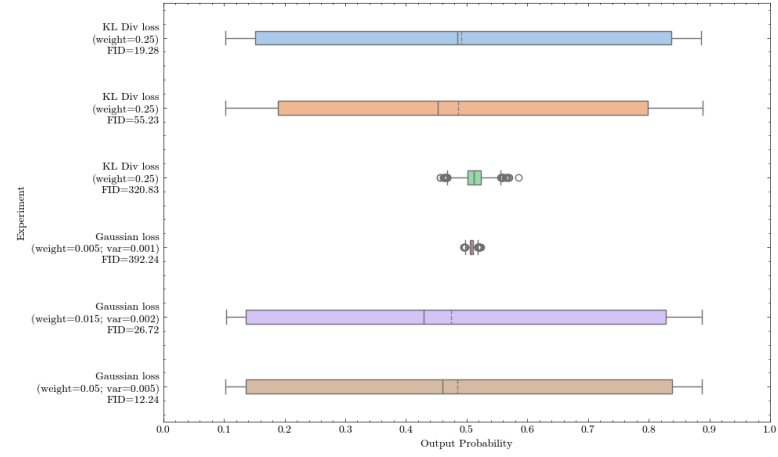
(a) GASTeN Architecture



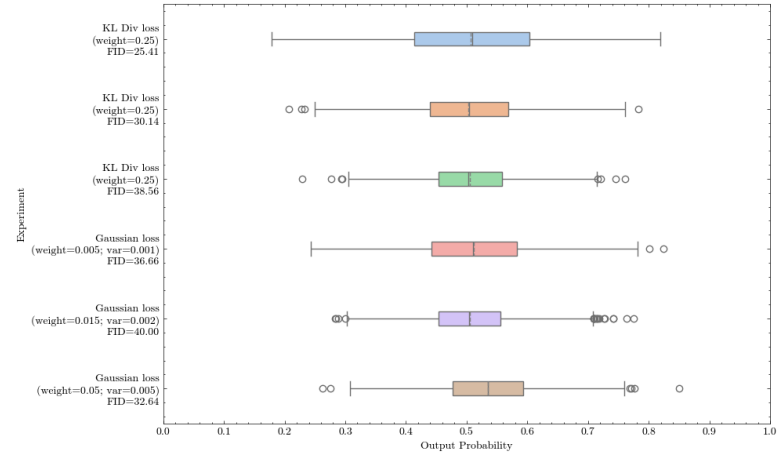
(b) AmbiGAN Architecture

Figure G.7: Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, Meta-Learner Estimator, all hyperparams variations

G.2.4 No Output Combination Estimator



(a) GASTeN Architecture



(b) AmbiGAN Architecture

Figure G.8: Boxplot comparison 9v8, GASTeN and AmbiGAN architecture, No Output Estimator, all hyperparams variations

Appendix H

H.1 7v1 Experiment

Table H.1: RESNET-50's Hubris evaluation for 7v1 for all types of output

Dataset	AmbiGAN type	ACD	FID	Hubris _A ^a	Hubris _R ^b
MNIST [143]	-	0.500	3.409	0.972	0.053
AmbiguousMNIST [141]	-	0.497	68.955	0.971	0.259
AmbiGAN-MNIST	Mean Output Combination	0.483	52.796	0.966	0.891
AmbiGAN-MNIST	Linear Output Combination	0.490	42.232	0.969	0.495
AmbiGAN-MNIST	Meta-Learner Output Combination	0.497	21.983	0.971	0.138
AmbiGAN-MNIST	No Output Combination (KL-div)	0.487	55.809	0.967	0.684
AmbiGAN-MNIST	No Output Combination (Gaussian)	0.484	69.4043	0.966	0.868

^a Absolute Hubris: Hubris(ref = 0.50)

^b Relative Hubris: Hubris(ref = Estimator Output Distribution)

H.2 9v8 Experiment

Table H.2: RESNET-50's Hubris evaluation for 9v8 for all types of output

Dataset	AmbiGAN type	ACD	FID	Hubris _A ^a	Hubris _R ^b
MNIST [143]	-	0.500	3.295	0.972	0.177
AmbiguousMNIST [141]	-	0.495	31.570	0.971	0.536
AmbiGAN-MNIST	Mean Output Combination	0.493	23.573	0.970	0.913
AmbiGAN-MNIST	Linear Output Combination	0.495	31.015	0.970	0.954
AmbiGAN-MNIST	Meta-Learner Output Combination	0.498	25.032	0.969	0.895
AmbiGAN-MNIST	No Output Combination (KL-div)	0.494	25.734	0.970	0.872
AmbiGAN-MNIST	No Output Combination (Gaussian)	0.492	32.349	0.970	0.904

^a Absolute Hubris: Hubris(ref = 0.50)

^b Relative Hubris: Hubris(ref = Estimator Output Distribution)