FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Aquaculture Fish Quality Control Using Synthetic Data

Hugo Miguel Miranda Barros

MSC IN BIOENGINEERING - BIOMEDICAL ENGINEERING

Supervisor: Pedro Miguel Vendas da Costa, MSc

Co-supervisor: Jaime dos Santos Cardoso, PhD

June 28, 2020

Aquaculture Fish Quality Control Using Synthetic Data

Hugo Miguel Miranda Barros

MSC IN BIOENGINEERING - BIOMEDICAL ENGINEERING

June 28, 2020

Resumo

A aquicultura é um dos meios de produção de recursos alimentares animais com menor impacto ambiental e, por esse motivo, é uma alternativa ao consumo de carne com base em indústrias mais poluentes. No entanto, os custos associados à aquicultura necessitam de ser reduzidos para que seja viável a sua aplicação em larga escala. Um sistema que monitorize automaticamente tanques de aquicultura, detetando peixes não saudáveis ou mortos, e que estime o tamanho e peso do animal, ajudaria a automatizar as operações envolvidas na aquicultura. Para além disso, uma ferramenta capaz de estimar o tamanho do peixe pode ser bastante útil para adquirir dados sobre as espécies pescadas, para perceber o limite da embarcação, para fins estatísticos ou ainda para melhorar e expeditar o processo de ordenação e separação em terra.

Nesta dissertação, é esperado o desenvolvimento de um ou mais modelos computacionais que consigam identificar o peixe na imagem, medir algumas das suas características e contar o número de peixes em cada imagem. No entanto, é difícil obter imagens submarinas anotadas com informação pertinente, nomeadamente máscaras de segmentação e/ou distâncias. Assim, faz parte do trabalho uma aquisição de imagens geradas por um simulador submarino que imita o ambiente real. Procura-se, ainda, testar o modelo resultante em imagens reais, havendo o desafio adicional de transferir o conhecimento obtido no ambiente virtual para o ambiente real.

Para alcançar os objectivos previstos, elaborou-se um modelo de segmentação de peixes que visa isolar o peixe, utilizando *bounding boxes* como anotação para segmentação, com uma função de custo adaptada, em imagens reais. Mais ainda, desenvolveu-se modelos para adaptação de domínio e aprendizagem por multi-tarefa (*multitasking*), nomeadamente predição simultânea de profundidade e segmentação, de forma a aproveitar os dados obtidos através do simulador. Foi também realizada a contagem de peixes por imagem utilizando operações morfológicas.

Em suma, o trabalho desenvolvido utilizou técnicas de aprendizagem computacional que se focaram na predição de máscaras de segmentação e medição de distâncias, armando-se de técnicas de adaptação de domínio e de *multitasking*. Para os modelos de segmentação sem adaptação de domínio o melhor resultado obtido foi de 0.64 e 0.37 *IoU* para validação e teste, respectivamente, correspondente ao modelo de predição com *bounding boxes* como anotação. Já usando adaptação de domínio o melhor resultado foi 0.69 para a validação e 0.29 para o teste correspondente a uma adaptação da rede *cycle-GAN* com rede de segmentação integrada no treino e usando multitask learning. O mesmo modelo revelou também o melhor resultado para a contagem de peixes no conjunto de validação, com uma *accuracy* de 0.87. Quanto ao conjunto de teste, nesta categoria, o melhor modelo foi para o modelo de predição com *bounding boxes* como anotação, com uma *accuracy* de 0.58.

Abstract

Aquaculture is one of the methods of producing animal food resources with less environmental impact, being an alternative to meat whose industry is more polluting. However, aquaculture costs need to be reduced if it is to be used at scale. A system that can automatically monitor an aquaculture farm by detecting unhealthy or dead fish and estimating the size and weight of the fish helps to scale aquaculture operations. In addition, a tool capable of estimating the size of the fish can be very useful in the fishing industry to acquire data on the species caught, to understand the vessel's limit, for statistical purposes or to improve and expedite the sorting and separation process on land.

In this dissertation, it is expected to develop one or more computational models that can identify the fish in the image, measure some of its characteristics and count the number of fish in each image. However, it is difficult to obtain underwater images annotated with relevant information, namely segmentation masks and/or distances. Thus, part of the work is an acquisition of images generated by an underwater simulator that mimics the real environment. It is necessary to test the resulting model in real images, with the additional challenge of transferring the knowledge obtained in the virtual environment to the real environment.

To achieve the expected results, a fish segmentation model was developed to isolate the fish, using bounding boxes as an annotation for segmentation, with an adapted loss function, in real images. Additionally, it was developed several models for domain adaptation and multitask learning , namely simultaneous depth prediction and segmentation, in order to take advantage of the data obtained through the simulator. Fish counting in each image was also performed using morphological operations.

In short, the work developed used computational learning techniques that focused on the prediction of segmentation masks and distance measurement, using domain adaptation and multitasking techniques. For segmentation models without domain adaptation, the best result was 0.64 and 0.37 IoU for validation and testing, respectively, corresponding to the prediction model with bounding boxes as annotations. Using domain adaptation, the best result was 0.69 for validation and 0.29 for the test, corresponding to an adaptation of the cycle-GAN network with a segmentation network integrated in the training and multitask learning. The same model also revealed the best result for fish counting in the validation set, with an accuracy of 0.87. As for the test set, in this category, the best result was observed in the prediction model with bounding boxes as an annotation, with an accuracy of 0.58.

Agradecimentos

Primeiramente, agradeço ao professor Jaime Cardoso e ao Pedro Costa por toda a disponibilidade e orientação ao longo destes meses. De seguida, ao Filipe Marques e Filipa Castro pelo acompanhamento e ajuda no ambiente de trabalho. Aos meus pais deixo uma nota de gratidão imensuravél por todo o suporte emocional e financeiro ao longo destes anos e, finalmente, à Inês Torres pelo apoio incondicional.

Hugo Barros

"Quando o sol se põe e diz adeus às montanhas,
ele mostra os seus raios mais bonitos para que não se esqueçam dele e esperem o seu regresso no dia seguinte"

Avô da Heidi

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AP	Average Precision
AUV	Autonomous Underwater Vehicle
ROV	Remotely Operated Vehicle
SfM	Structure from Motion
BatchNorm	Batch Normalisation
CNN	Convolutional Neural Network
FC	Fully Connected
ReLU	Rectified Linear Unit
SLAM	Simultaneous Localisation and Mapping
RoR	Residual networks Of Residual networks
GAN	Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
CRF	Conditional Random Field
Rel	Mean Relative Error
$\log_{10}$	Mean of Logarithmic Error
RMSE	Root Mean Squared Error
SSIM	Structure Similarity
LiDAR	Light Detection and Ranging
Sonar	Sound Navigation and Ranging
Radar	Radio Detection and Ranging
MAE	Mean Absolute Error
MSE	Mean Square Error
IoU	Intersection Over Union
NN	Nearest Neighbour
BCE	Binary Cross Entropy

Chapter 1

Introduction

1.1 Context and Motivation

The aquaculture production in China is 60% of the world's total production while the European Union's production amounts to 1.53%. Still, this value represents 1.25 tonnes of commercialised fish and molluscs corresponding to a 5 billion euros industry [1]. With a tendency to grow, the aquaculture production is expected to surpass the fisheries supply in the next years.

Aquaculture, however, deals with enormous amounts of fish, making it difficult to monitor at large scale, increasing the number of deaths and consequently the costs. With the intention of lowering the cost in order to obtain an accessible and diverse source of food to people, a system capable of monitoring the ever growing farms, could manifest as a reliable tool to decrease the production costs. Calculating the weight and size of the fish could also help to monitor the overall health and to predict a suitable time for the fish commercialisation.

To achieve the the supra-mentioned goal a simple, cost-effective and versatile method is necessary. With the advancements of machine learning and deep learning techniques, it becomes possible to obtain relevant results using a normal underwater video camera, without any additional sensor. For this particular case, it could be used for monitoring and calculating morphological features of the fish, such as the body length, the body width and weight.

Machine learning, specifically deep learning, normally require large amount of annotated data yet, in the underwater environment, the acquisition of supervised data is costly and time-consuming, which leads to a reduced number of existing datasets. Therefore, it is proposed to make use of synthetic data from a simulator with underwater scenes to increase the amount of annotated data. To use images from a simulator it is necessary a transfer knowledge technique to make use of the information retrieved in the simulated environment to the real one.

Concluding, the identification of the fish for its measurements can be achieved with machine learning techniques capable of object identification and segmentation. Domain adaptation techniques can be used to expand the existing small size datasets. And all this can be achieved by

[1] https://ec.europa.eu/fisheries/cfp/aquaculture/facts_en. (Access: 19-06.2020)

camera based monitoring systems, developed with machine and deep learning techniques resulting in a potentially reduction of the associated costs.

1.2 Objectives and Contributions

The main goal of the dissertation is to develop a monitoring system, making use of synthetic data and extrapolate to real world data in order to achieve effective monitoring of aquaculture fish. In the end, the main goals are:

- develop a segmentation prediction model to identify and isolate the fish;
- create a domain adaptation method to allow transfer knowledge from the trained models to real world images;
- use depth map information in the simulated datasets in multitask setting;
- extract fish numbers from each image;

After reviewing the related works the project was put in motion resulting in the following contributions;

- annotation of 569 segmentation masks of an existent dataset;
- development of a supervised model for fish segmentation leveraging existing annotated data in the form of bounding box to improve segmentation results;
- study of the effects of simulated data for real prediction making use of different methods;
- assessment on the effect of adding data augmentation and multitask learning.

1.3 Structure

Excluding the introduction, this report has a chapter committed to literature review (chapter 2) where a light introduction to Deep Learning approaches is made (section 2.1) to facilitate further reading, followed by fish and object segmentation - section 2.3 - domain adaptation research works - section 2.4 - multitask learning - section 2.5- and depth prediction related literature - section 2.6

Chapter 3 presents the explanation of the framework implemented to address the issues at hand, including the detailed exposition of each step of the pipeline. The segmentation model is described in section 3.3 and domain adaptation techniques in section 3.4. In chapter 4 the results of the employed methods are exposed, organised and discussed.

Finally, in chapter 5 the last comments are written in order to critically assess the overall outcomes of the work.

The appendix A also has information on the state-of-the-art results in table format for depth prediction methods and segmentation methods.

Chapter 2

Literature Review

2.1 Background on Machine Learning

Machine learning makes use of algorithms that make inferences from data to learn new tasks. The algorithms use existing data, extract features from it and construct a mathematical model capable of performing a given task.

2.1.1 Nearest Neighbour

Nearest Neighbour (NN), or more specifically K-Nearest Neighbour Classifier (Peterson, 2009) makes predictions based on the known classification of its neighbours. This is a classification or regression algorithm. A new unknown data point is assigned to a class according to the distance between the new sample and all samples in the training set. It selects the k nearest samples, where k is the number of neighbours considered in the selection step and their neighbours will define the class of the new data point

2.1.2 Unsupervised Nearest Neighbour

There exists, however, an unsupervised version of NN, one that it is limited to find the nearest neighbours.

To find a nearest-neighbour, it is possible to compute all pairwise distances but it might not be very efficient. This is why there are smarter ways that use specific data structures, for instance Ball Tree, that sorts data points into nested sets. Basically, in an unsupervised NN the data is stored in a structure based on the structuring algorithm. Afterwards, it is possible to retrieve the closes neighbours given a new data point.

2.2 Background on Deep Learning

Deep learning is a branch of machine learning composed of a variety of deep artificial neural networks. The composition is based on several interconnected layers that learn high level features

through a combination of simple operations that can be updated by simply changing its weights (coefficients). The features are extracted automatically in each layer of the network. To guide the network in extracting the best features it is necessary a cost/loss function which will evaluate the output of the network with the chosen metric and constraining the network by updating the weights. The basics for Fully Connected Networks (FC) and Convolutional Neural Networks as well as some basic architectures and models are presented hereinafter.

2.2.1 Fully Connected Networks

A fully connected network is a network constituted of several layers of neurons represented in a vector form. It can be also called Multilayer Perceptron (MLP).

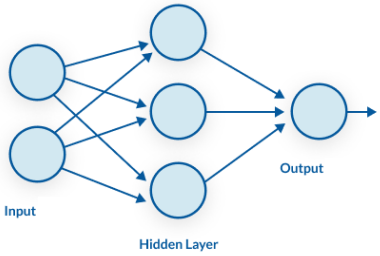

Figure 2.1: The first FC layer maps the input to the hidden layer and the second FC maps the hidden layer to the output.[1]

In figure 2.1, an example of a simple and classic structure for a MLP is visible . The output, y, of the FC layer is a linear combination between the input, f, of the layer and its weights, w. h represents an activation function which is a non-linear transformation of the input better explained in the section bellow.

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j h_j(\mathbf{x})\right) \tag{2.1}$$

2.2.2 Convolutional Neural Networks

CNNs are other type of network often used to handle image data as it is expressed in matrix form. In a CNN the fundamental parts are convolutional layers, activations layers and pooling layers. Often, FC layers are part of CNN's. Figure 2.2 shows a typical architecture (LeNet-5 from Lecun et al. (1998)).

The convolutional layers are matrix convolutions between the input ($height1 \times width1 \times depth$) and kernel (filter) with a specific dimensions, $height2 \times width2 \times depth$. Each filter is convolved across the width and height of the input volume and computes the dot product between

[1] https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer&-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/. (Access: 21-01-2020)

the elements of the filter and the input. The output dimensions depends on several factors and can be expressed in the equation:

$$output\_height = \frac{height1 - height2 + 2padding}{stride} + 1, \tag{2.2}$$

where padding represents an addition of pixels around the input and stride controls the shifting of the kernel around the input. The calculation is valid for height and width.

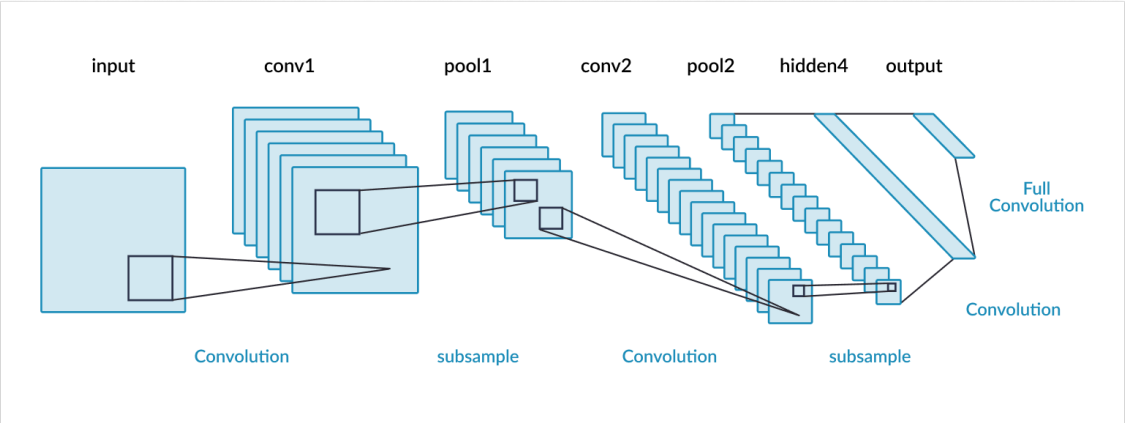

Figure 2.2: Convolutional Neural Network.[2]. The convolutional layers are followed by pooling layer for downsampling (or subsampling). The hidden layer represents a fully connected layer with two dimensions similar to the one in figure 2.1

The pooling layer is a non linear operation specific for downsampling (or subsambling, as in the image 2.2). The two most common types of pooling (figure 2.3) are average and max pooling. Max pooling is normally preferred for its noise reduction and translation invariance effects.

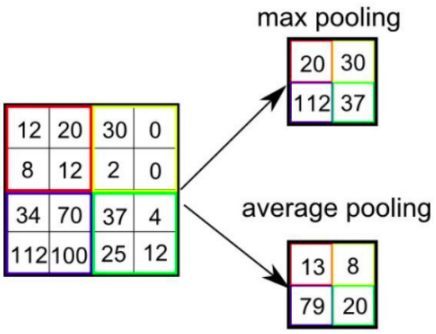

Figure 2.3: Types of pooling in a pooling layer with a 2×2 kernel. Image retrieved from[3].

[2] https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/. (Access: 22-01-2020)

[3] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural&-networks-the-eli5-way-3bd2b1164a53. (Access: 22-01-2020)

The last important layer is the **activation layer**, responsible to apply a non-linear transformation on the output of the previous layer, which is necessary otherwise the composition of multiple linear transformations (Conv Layer or FC layer) reduces to a single linear transformation. Activation functions allow the network to learn more complex non-linear decision boundaries. Common activations functions are ReLu -Rectified Linear Unit- and sigmoid, represented in figure 2.4.

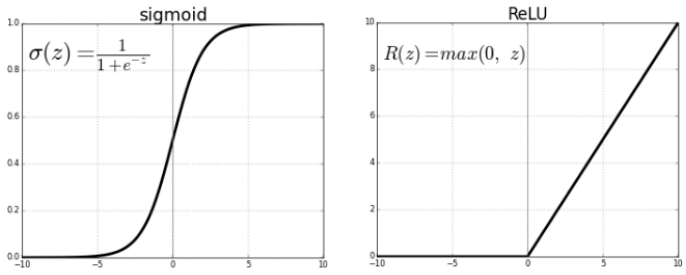

Figure 2.4: ReLu and Sigmoid activation functions. Image retrieved from[4].

2.2.3 Batch Normalisation

Batch Normalisation (BatchNorm) is a technique that normalises activations along the neural network layers, it mitigates the amplification of small alterations in the parameters, preventing high gradient values. In summary, BatchNorm normalises the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. The latter, however, introduces a shift/scale on the previous activations function which leads to imperfect weights to the succeeding layer. This problem is culminated by the introduction of two additional weight parameters that allow the denormalisation (using, for instance, Stochastic Gradient Descent for weight update [5]) by taking into consideration and altering the values of theses two weights in each activations.

Th normalisation, $\overline{x_i}$, is calculated using the batch mean, μ_B, and the batch standard deviation, σ_B^2.

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{2.3}$$

The normalisation is followed by the scaling and shifting operation, where γ and β are learnable parameters.

$$y_i = \gamma\overline{x_i} + \gamma \tag{2.4}$$

[4] https://medium.com/@himanshuxd/activation-functions-sigmoid-relu&-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e. (Access: 22-01-2020)

[5] https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31 (Access: 22-01-2020)

By normalising the input layer through activations scaling and adjustment, it enables the use of higher learning rates as it mitigates the explosion or vanishing of the gradients. Furthermore, according to the original paper (Ioffe and Szegedy, 2015) Batch Normalisation also has regularisation effects, having a positive role in preventing overfitting.

2.2.4 Dropout

A dropout layer in machine learning is implemented by randomly ignoring neurons in the network layers, meaning that the neural network will be incomplete in each training iteration. The goal of dropout is the regularisation of the model, preventing overfitting and allowing generalisation. By masking a percentage of the neurons in each training iteration the model will be less likely to form dependencies between certain neurons and hopefully avoid overfitting.

2.2.5 CNN Architectures

There are countless architectures developed over the years. Some of the first architecture include LeNet-5 (Lecun et al., 1998), VGG (Simonyan and Zisserman, 2014) or AlexNet (Krizhevsky et al., 2012) that constitute the basis for following developments. The architectures are normally specific to different tasks and should be adapted according to the problem. Some of the architectures researched are described in the following section.

2.2.5.1 Residual Networks

This architecture is extensively used since its publication and it is intended to solve the problem of degradation in deep neural networks. For that reason, the skip connections (Figure 2.5) are formally presented in the work of He et al. (2015) .

Taking into consideration a set of convolution layers, with x as input and resulting in $Y(x)$, this set of layers can benefit with the presence of a skip connection, or shortcut connections allowing to learn a residual function resulting in $Y(x) = F(x)+x$ which is different from the original set of layers.

The difference (or residual) of a neural network work can be denoted as $F(x) = Output\text{-}Input = Y(x) - x$. The Skip connection presented in figure 2.5, where $F(x) = W_2[ReLu(W_1x)]$, can be accomplished by simply adding to $F(x)$ the initial object x in order to obtain $Y(x) = F(x) + x$ being x the identity function. The goal of the CNN is to learn the true output, $Y(x)$, however as $Y(x) = F(x) + x$ then, objectively what the residual block is trying to learn is only the residual $F(x)$ as the identity function is provided.

This makes the learning process easier as the identity function can be difficult to learn on very deep neural networks as the deeper the network the more apparent may become the curse of dimensionality [6], the vanishing gradient [7] and accuracy saturation [8]. However with the introduction of shortcuts, the identity function is preserved and, simultaneously, the network tries to learn the residual mapping *F(x, W_i)*. According to the original work of He et al. (2015) by applying several residual blocks in cascade, the accuracy saturation observed in deep plain neural networks is mitigated while the computational power is not increased. To conclude, ResNet presents itself as a fine alternative to plain neural networks combating some of its main problems and therefore increasing the overall performance.

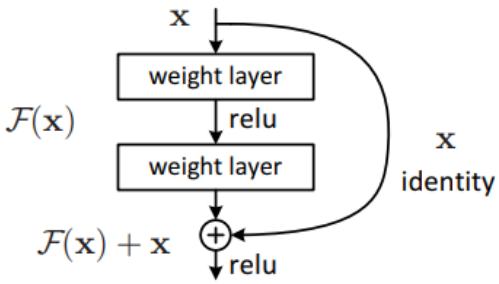

Figure 2.5: Residual network building block from He et al. (2015).

2.2.5.2 Residual Networks of Residual Networks

Residual Networks of Residual Networks (RoR) proposes a solution built on top of standard Residual Networks.

As represented in Figure 2.6 and as the name suggests, RoR consists in two added dimensions of skip connections. Said dimensions exist not only in the residual block but also across a group of residual blocks and repeated at a higher level across a group of "groups of residual blocks". The additional shortcut may be advantageous as the layers in upper blocks can also propagate information to layers in lower blocks. Zhang et al. (2017) presents a RoR composed of groups with 3 residual blocks (1st dimension of skip connections) with a skip connection involving them (2nd dimension of skip connections) and finally a third dimension of skip connections that envelops all residual blocks. The architecture is compared with similar depth ResNet and achieved better results in several image classification tasks.

[6] https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e. (Access:15-12-2019)

[7] https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484. (Access:15-12-2019)

[8] https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec. (Access:15-12-2019)

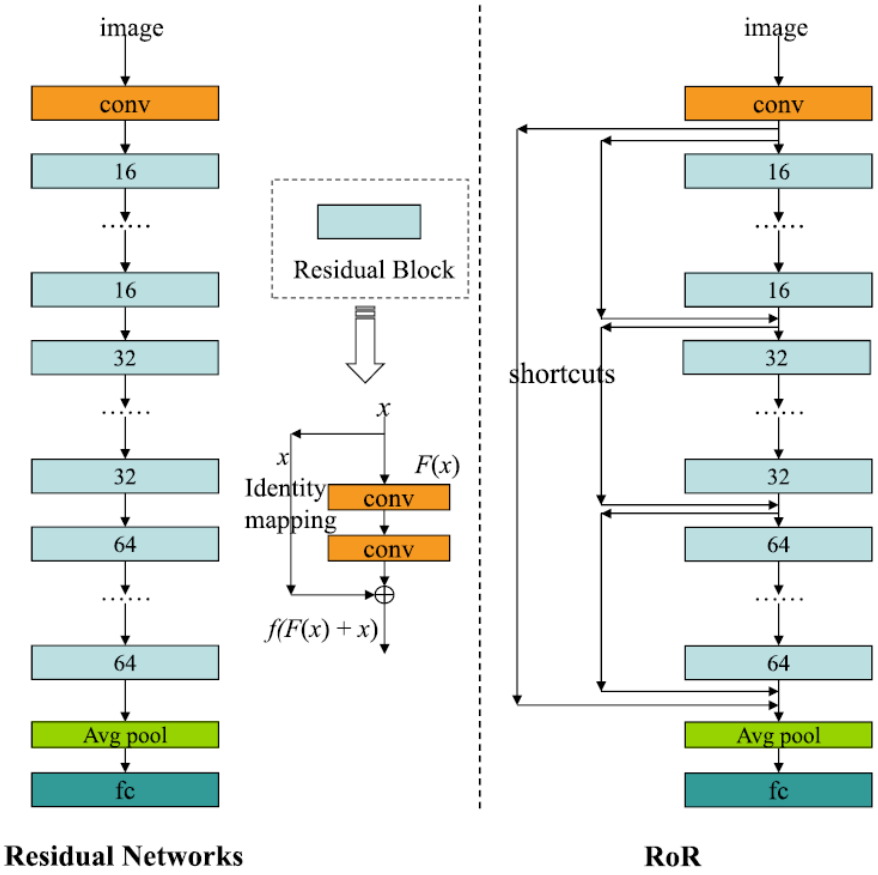

Figure 2.6: Comparison between ResNet and RoR (Zhang et al., 2017).

2.2.5.3 U-net

The U-net architecture takes its name from the obvious U-shaped format illustrated in Figure 2.7.

The characteristically symmetric shape is formed by two separate but interconnected paths: the contracting path and the expansion path.

• **Contracting path**: it is constituted by several blocks of 3×3 convolutional layers and a 2×2 max pooling layer (stride 2) for downsampling, which allows the extraction of more advanced features but it also reduces the size of the feature maps.

• **Expansion path**: constituted by blocks with an Up-convolution layer, a concatenation operation and a 3×3 convolution. The latter path serves as to restore the original shape and the concatenation operation (marked as grey arrows in Figure 2.7) is extremely important to provide the localisation information from the contraction path to the expansion path.

Originally U-net architecture was developed for image segmentation tasks, however, it has also been used for regression tasks (Yao et al., 2018).

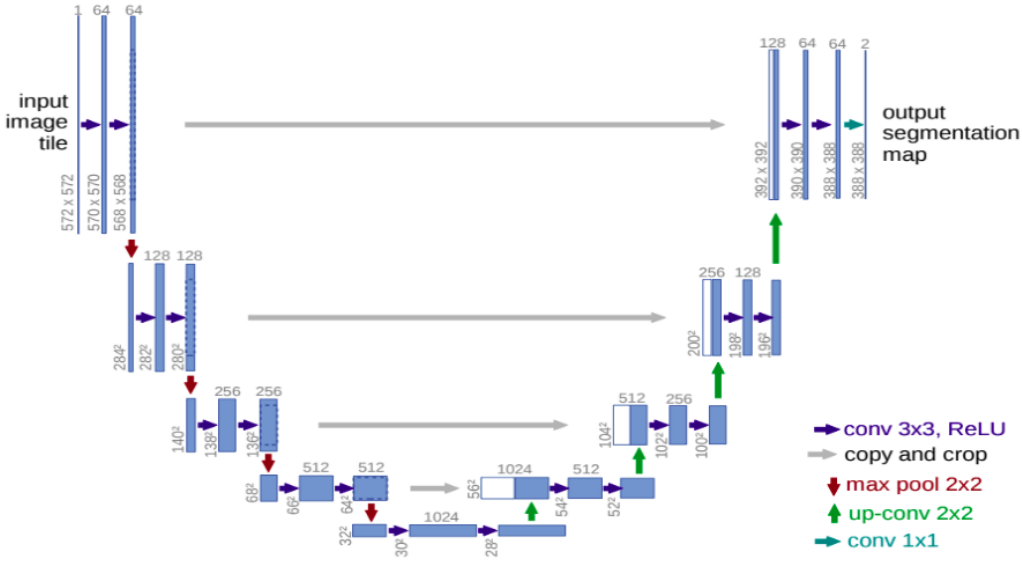

Figure 2.7: U-Net architecture (Zhang et al., 2015).

2.2.5.4 ResUNet

ResUNet (Zhang et al., 2018), as the name suggest, is an assimilation of the residual learning in the U-net architecture (Figure 2.8). This is achieved adding a residual connection between every convolution of the U-net architecture. The architecture grants advantages of both U-net and residual networks. In the residual block however, they institute atrous convolutions with different dilation values that are added in the end of the block, as perceived in figure 2.8 (right).

2.2.5.5 Generative Adversarial Network

Generative Adversarial Network (GAN) is described by Goodfellow et al. (2014) as an estimation of a generative model resorting to an adversarial process. GAN is constituted by two models: the Generative, G, and the Discriminative, D. The generative model intends to generate images as authentic as possible when compared with the training images, in order to induce miss-classification in the Discriminator. Contrarily, the Discriminator aims to distinguish between images created by G and real training images, classifying with respects to its origin. The objective is attained when G is capable of generating perfect falsification and the discriminator deems all the classifications as 50% probability of being either generated or real.

Considering x to be an image, the Discriminator, $D(x)$, outputs the (scalar) probability of x's origins and tries to maximise said probability, $log[D(x)]$. Moreover considering z to be a latent space vector sampled from a standard normal distribution, $G(z)$ represents the generator model which maps the latent vector z to data-space. G intends to estimate the distribution of the training data, P_{data}, so it may generate falsification images based on the estimated distribution. Briefly, G attempts to minimise the probability of a correct prediction by D, $log[1\text{-}D(G(x))]$.

The function describing the loss is represented by equation 2.5.

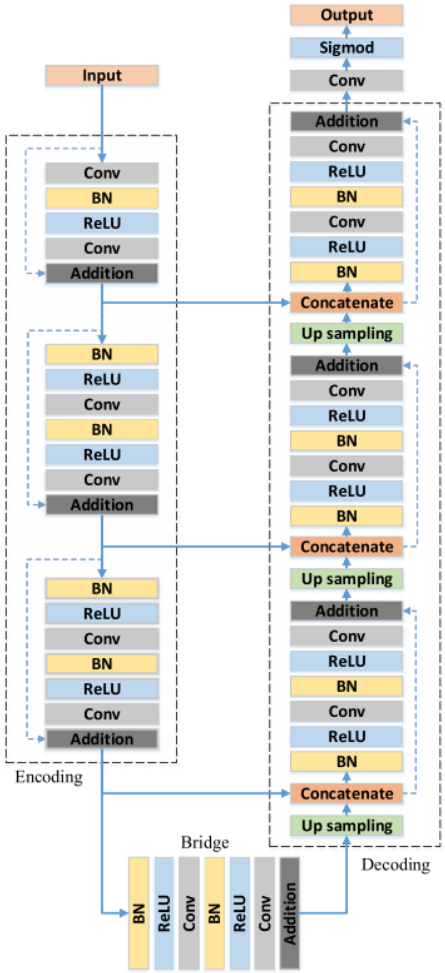

Figure 2.8: ResUNet architecture (Zhang et al., 2018).

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]. \tag{2.5}$$

Another significant mention is the Deep Convolutional Generative Adversarial Network (DCGAN). It is a variation from normal GAN but uses convolutional and transposed convolution layers in the discriminator and generator, respectively as stated originally by Alec et al. (2015). GANs and DCGANs have been widely used in unsupervised tasks and proved to be an effective tool for domain adaptation challenges as discussed herein in section 2.4.

2.2.6 Data Augmentation

In computer vision problems, specially using deep learning techniques, the generalisation is something to take in consideration. Having a small sample set prevents learning of a broad spectrum of

features and leads to unwanted overfitting to the training set. Data augmentation makes use of relevant modification to the input data, as a way to enlarge the dataset and introduce more variability, preventing overfitting and improving generalisation.

Some modification can be basic with predefined variables, such as translations, rotations, scaling, colour filters, etc. Others may be learned by the model (Shorten and Khoshgoftaar, 2019).

2.3 Segmentation

To perform the isolation of individual and small groups of fish it is necessary to retrieve the shape of the animal. By detecting and identifying the shape of the fish it becomes possible to calculate its morphological features. This step is crucial to the advancement of the work. Object segmentation has a long history and hundreds of published work.

Some of the architectures often used for segmentation are U-net (Zhang et al., 2015), ResNet (He et al., 2015) and its many variations, and more recently ResUNet, previously mentioned before. The V-net (Milletari et al., 2016) was also used extensively, being very similar to U-net and variations of SegNet (Badrinarayanan et al., 2017) are often used as they are simple encoder-decoder architectures. However, to reduce the spectrum, only more recent works, and works related to fish segmentation, will be presented.

Segmentation tasks can be divided in instance segmentation or semantic segmentation depending if the segmentation isolates individuals or classes of objects, respectively. In this section some base knowledge related to segmentation state-of-the-art is presented, including the most prominent datasets, evaluations metrics, generalised segmentation works and specific works related with fish segmentation. Table A.1 and table A.2 have a few compiled results for the coco instance segmentation competition.

2.3.1 Datasets

There are several datasets for image segmentation. The following section presents some of the most prominent.

2.3.1.1 Cityscapes

Figure 2.9: Images from cityscapes dataset.

Cityscapes was first introduced by Cordts et al. (2016) and has been mainly used for semantic segmentation tasks however many depth related works have used the stereo images of this dataset to train unsupervised depth estimation models. The images were recorded while driving in 50 cities during several months in three different seasons (summer, spring and fall). The dataset has 25000 image pairs with 2048×1024 and baseline around 22cm. Figure 2.9 represents an example of this dataset.

2.3.1.2 PASCAL VOC 2012

This dataset (Everingham et al.) has 20 classes with 11,530 images containing 27,450 annotated objects and 6,929 segmentation masks. The classes are very different, including bikes, dogs, people, etc. Figure 2.10 has an example of the dataset.

Figure 2.10: Images from PASCAL VOC 2012 dataset.

2.3.1.3 COCO Dataset

The Microsoft Common Objects in Context dataset (Lin et al., 2014) has 91 common object categories with a total of 2,500,000 labelled instances in 328,000 images. It is also a general dataset with very different categories some of which can be seen in figure 2.11.

Figure 2.11: Images from COCO dataset.

2.3.2 Evaluation Metrics

For segmentation, some of the most used metrics are the Jaccard index, or Intersection Over Union (IoU), and the dice coefficient:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

$$Dice(A,B) = \frac{2|A \cap B|}{|A| + |B|} \tag{2.7}$$

IoU is the coefficient between the overlapping pixels in the mask and prediction and the total number of pixels. Dice corresponds to the double of the overlapping pixels divided by the sum of the pixels in both prediction and ground truth mask.

The Average Precision (AP) or the area under the precision-recall curve. The predictions are sorted according to the confidence level after which the class is determined as positive or negative (normally more than 0.5 IoU s positive but other threshold values can be used). After that, the precision recall curve is calculated taking into consideration the number of prediction including and above itself. For instance for the third highest prediction the number of objects is 3 and the precision-recall is calculated with those 3 points. Finally, all its left is to calculate the area under the curve.

For a limit of 0.5 in the IoU metric, the pixels with more than 0.5 will be considered as mask in the output and the AP is calculated based on that (AP50). However there is the mAP which is the mean AP for several threshold values (i.e. for COCO dataset, AP is calculated for [0.5,0.55,...,0.95].

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.8}$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.9}$$

2.3.3 Instance segmentation

Fast R-CNN (Girshick, 2015) in a network that performs object detection and classification of the objects. The network takes as input the images and a set of object proposals. Then, for each object proposal a region of interest (RoI) is extracted via pooling layer producing a fixed-length feature vector from the feature map, which is then is used to obtain a class and a bounding box. (He et al., 2017) developed Mask R-CNN which is an extension of Fast R-CNN (Girshick, 2015), however, in addition to the two outputs (bounding box and class) it is introduced a third output from instance segmentation. In the Mask R-CNN there are two RoI operations: the equivalent to Fast R-CNN and other for the segmentation channel.

Liu et al. (2018) based their work in Mask R-CNN architecture, however it uses a feature pyramid network with lateral connections that propagate semantically strong features from low level layers to top layers, enhancing all features. They also developed adaptive feature pooling in order to aggregate features from all feature levels. Additionally an FC layer is introduced in the mask prediction branch, which, the authors claim, have the ability to adapt to different spatial locations and are able to differentiate instances, improving overall results. Although this work main objective is image classification it outperformed many works in instance and semantic segmentation and currently has one of the best scores in the COCO dataset.

Du et al. (2020) followed a different approach and constructed a scale-permuted network which permutes the layers based on ResNet-FPN. FPN or Feature Pyramid Network combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections.

The authors claim that scale-decreased backbone throws away the spatial information by down-sampling and, therefore, a model that changes the feature maps dimensions at any point in the architecture can preserve those features. In the model there are also cross-scale connections, seen in figure 2.12, where different scales are re-sampled and fused together which contributes to the preservation of high-resolution features as well.

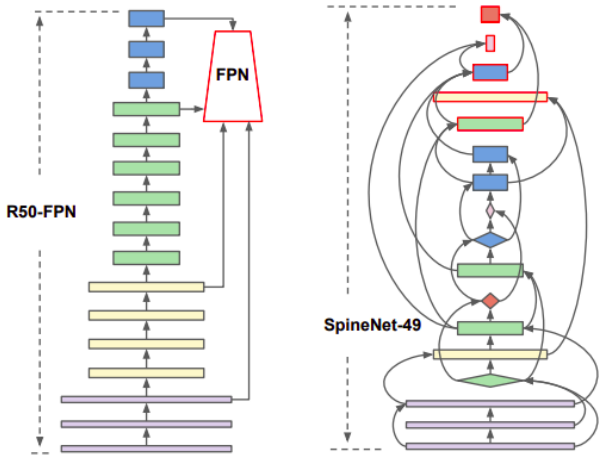

Figure 2.12: SpineNet compared with a ResNet. From Du et al. (2020).

Another ResNet variant is described in the work of Zhang et al. (2020). They introduce a "Split-Attention" block that splits the input feature map into R x K groups along the channel dimension and applies separate convolutions to each groups. The feature representation for each block is then obtained by a weighted combination of the representations of its splits. The blocks can be easily stacked and showed slight improvements when compared with tradition ResNet models.

Liu et al. (2019) introduced CBNet which is a backbone architecture characterised by the assembly of several backbones or paths. The structure is composed of a main path and assistance paths connected by feeding the output features of the previous backbone as part of input features to the succeeding backbone. The idea is better understood in figure 2.13.

2.3.4 Semantic segmentation

Long et al. (2015) were responsible for one of the first successful works using neural network in a end-to-end fashion for semantic segmentation. Adapting their architecture from classification networks such as Alexnet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2014) and

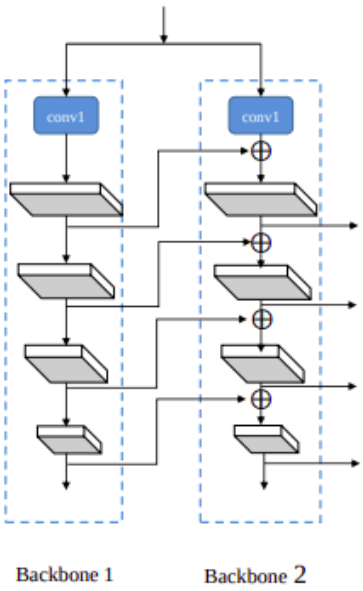

Figure 2.13: CBNet backbone architecture using two paths. The number of backbones (or paths) may be bigger. From Liu et al. (2019).

GoogLeNet (Szegedy et al., 2015), they were able to construct a neural network with deconvolution layers and obtain a pixel dense prediction (segmentation). They also used skip connection to combine coarse, high layer information with fine, low layer information.

Noh et al. (2015) released their paper along Long et al. (2015) and they propose a similar solution. Using VGG as a starting point they create a deconvolutional network that is a mirror of the convolutional part. While the previous work used strided deconvolutions, up-sampling directly in the convolution, this work uses "unpooling" layers for up-sampling. The output is, as expected, a mask with the same resolution as the input.

Since 2015, however, the developments have been increasing in the last years with more complex techniques. Following the advancements, RefineNet (Lin et al., 2017) was developed using multi paths for multi levels, posteriorly joining the low-resolution semantic features of the higher levels with fine-grained low-level features allowing, that way, a refinement of the features.

More recently, Fu et al. (2019), used a Resnet backbone and a dual attention model for segmentation. The model, in figure 2.14, divides it self in two parts: a Position Attention Module and a Channel Attention Module. The first is responsible for enhancing spacial representation and relations of local features, improving intra-class consistency. The channel attention model is similar however it is focused in the relations between channels, modelling inter-dependencies between channels.

Ding et al. (2019) also explores the relation between pixels of the same class. They proposed a neural network with paired convolution that learns relations between pixels and aggregates the

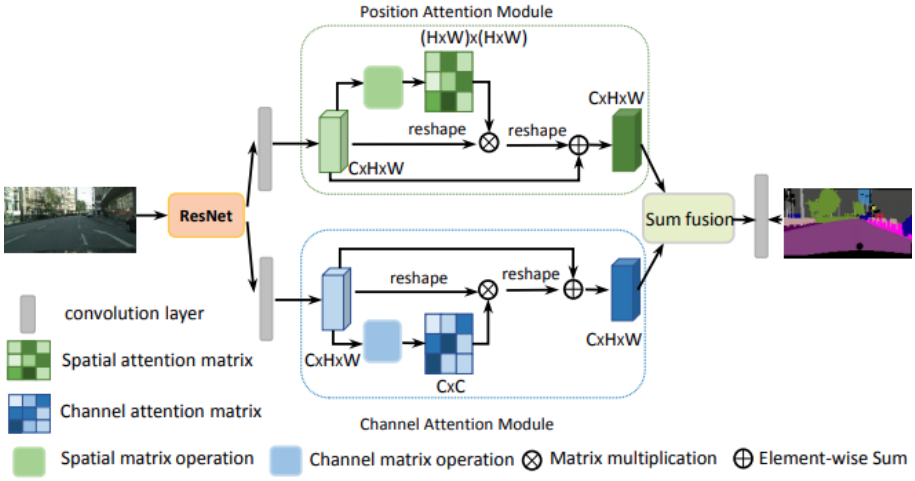

Figure 2.14: Dual Attention Network architecture using two modules. From Fu et al. (2019).

context information of a pixel from its semantic-correlated region. By now, it is clear that the relations between pixels are important to form a segmentation for each class, which makes sense as different object of the same class have common features. By exploiting that relation it can be achieved a superior prediction. In that same spirit, Yuan et al. (2019) also exploited relation between each pixel and each object region using aggregation techniques.

The ResUnet architecture in section 2.2.5.4 is also used in semantic segmentation problems as well as its descendants such as ResUNet-a (Diakogiannis et al., 2020) that modifies slightly the ResUNet architecture.

2.3.5 Fish Segmentation

Recurring to classical methods Yao et al. (2013) used the k-mean algorithm for fish segmentation and Wang et al. (2019b) used a contour-based segmentation. First, a segmentation method using background subtraction is applied, constituting the coarsest level for the entire contour alignment. At the finer level, rather than aligning the entire contour, they aligned two adjacent contour segments iteratively, which can recover the local structure of the fish body. They did not use deep learning techniques.

Prados et al. (2017) used a stereo camera setup to perform fish segmentation and measurement also in a classic manner with no Deep Learning methods. They start by estimating the scene background by considering fish-free images. This is followed by the elimination of non-uniform illumination which is achieved by converting the images to the HSV (Hue, Saturation, Value) colour space. The median of several background images is calculated and then the ratio between the S channel of the background images and the images with fish is calculated, resulting in a initial perception of the location of the fish, followed by a blob analyses. The S channel is used because it is invariant to illumination changes.

Yu et al. (2020) used a dataset with single fish in each image and captured above water, which do not replicate the conditions of the work at hands. However, their approach is similar to Wang et al. (2019a) with a pre-processing stage and then a CNN (Mask R-CNN) for segmentation of the fish and individual parts of the fish.

Garcia et al. (2019) performed fish segmentation using a dataset with stereo images and they implemented the pre-processing stage of Prados et al. (2017) to remove non-uniform light, followed by a Mask R-CNN (He et al., 2017) to segment the fish and separate them form the background. To conclude, the image is scaled to the original high dimensional resolution which results in loss of accuracy, leading to a segmentation refinement stage obtained through the calculation and threshold of the highest values of the gradients of the V channel on the original images (from the three channel image - HSV). As a final step, the length of the fish is estimated resorting to morphological skeletonization (multiple morphological operations to obtain the skeleton) and the initial and end point of each fish is obtained for the left and right image of the stereo pair and used to compute the distances of the segments connecting them using epipolar geometry, thanks to the calibration of the stereo system. They achieved an IoU of 0.80 for single fish and 0.61 for overlapping fish.

2.4 Domain Adaptation

Domain adaptation is a sub-discipline of machine learning and craves for the transformation of a source domain space to a target domain space which allows flexibility in training. For the particular case at hands the goal is to find a model that transforms synthetic underwater images to real underwater images (or vice-versa). Achieving the supra-described task it becomes possible to learn features, from synthetic data and extrapolate the knowledge to real world data. Based on this assumption some state-of-the-art research papers in the field of image domain adaptation are described in this section.

One of the most popular, though simple, domain adaptation methods was presented by Ganin and Lempitsky (2015). In figure 2.15 it is noticeable a simple feed forward CNN for feature extraction followed by Fully Connected layers for classification. It would be a classical classification problem was not for the existence of the domain classification segment.

Given input samples X_s from a source domain and the correspondent labels Y_s the goal is to find the labels for the input samples of a target domain, X_t. The authors attempt to perform classification and domain adaptation at the same time. The first segment produces a feature space, f, represented in figure 2.15. The second (blue) performs classification based on the inputs of the source domain, minimising the label prediction loss. The third part (pink) aims to make the features, f, domain-invariant. The domain classifier is trained to minimise the domain classification loss, and due to the gradient reversal layer, the feature network is trained to maximise the domain loss (by having the feature map as similar between domains as possible). The domain invariant

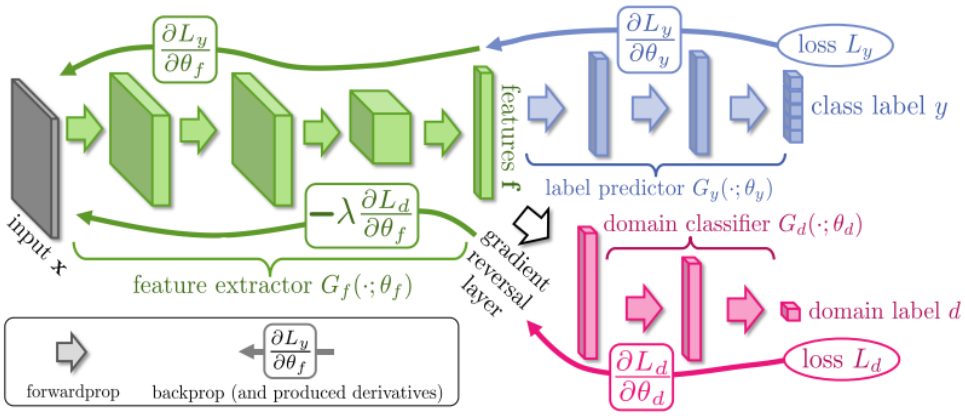

Figure 2.15: Architecture from Ganin and Lempitsky (2015) containing a feature extractor (green), a classifier (blue) and a binary domain classifier (pink) that tries to differentiate between two domain distributions.

feature space, f, is the first important step for domain adaptation, because it allows extractions of similar features from different domains. This methodology is simple and can be implemented in any feed forward network.

Tommasi and Caputo (2013) used another methodology to decrease the distance of the distributions between domains. In the presence of a class labelled and unlabelled domains, the authors use the Nearest Neighbour (NN) algorithm to find local distances between classes and tried to match those with the most similar images in the other domain. The intuition is that local differences may be present in two different domains and by finding similar images in both domains, the differences may be deduced from the labelled domain.

Sener et al. (2016b,a) followed a similar approach taking the NN inference into learning, by enforcing equal labels in the unsupervised domain compared to similar images in the supervised domain. They go further to ensure cycle consistency, meaning that if the unsupervised estimated labels are used to estimated the labels from the supervised data points, those labels must be equals to the ground truth labels.

Zhu et al. (2017) wrote one of the most prominent works in the area using cycle consistency and adversarial learning for image-to-image translation (CycleGAN). This work became the backbone for many further domain adaptation developments. To better explain the concept a figure from the paper of Atapour-Abarghouei and Breckon (2018) is presented, as it summarises the principle of the CycleGAN.

In figure 2.16 it is possible to visualise the transformation of A to B' on the generator $G_{A>B}$ proceeded by a domain classification on the discriminator, D_B, where the adversarial loss is applied. The same happens on the opposite spectrum using the generator $G_{B>A}$ and the discriminator D_A. The generated images A', B' are then introduced to the respective generators to attempt to

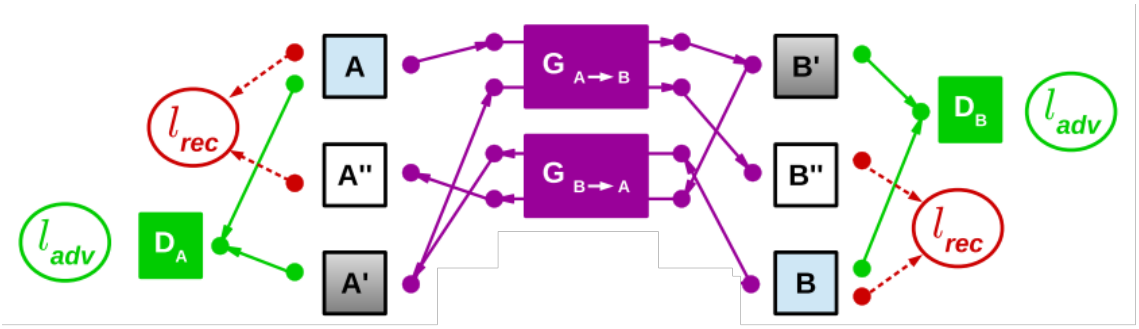

Figure 2.16: Approach from Atapour-Abarghouei and Breckon (2018). A represent one of the domains's RGB images, B are RGB images from other domain. $A'andB'$ are generated images and A'',B'' are cyclically regenerated images. G and D are the generators an discriminators, respectively.

reconstruct the original input A and B, with reconstructions being A'' and B'', respectively. This is achieved using a consistency loss that constrains A'' to be similar to A and the same thing for B'' and B. The consistency loss or reconstruction loss, l_{rec}

$$\mathscr{L}_{\text{cyc}}(G_{A>B}, G_{B>A}) = E_{A\sim p_{data}(A)}\left[\|A'' - A\|_1\right] + E_{B\sim p_{data}(B)}\left[\|B'' - B\|_1\right]. \tag{2.10}$$

In addition, a identity loss is applied to regularise the generators. This loss forces images of one domain, X, when passed through the generator that transforms Y to X, to not suffer changes. Zhu et al. (2017) approach allows for a controlled image-to-image translation. The discriminators try to differentiate between fake and true images images while the generator tries to fool the discriminator and both learn together, while the cycle of the images keeps constrains by imposing reconstruction of the images.

Hoffman et al. (2017) followed the previous work adding only a semantic loss. The authors trained a model in the source domain (domain with labels) and, as part of the training the original images and the domain translated images should be similar, enforcing semantic consistency.

Atapour-Abarghouei and Breckon (2018) used cycle consistency to learn depth and additionally to the supra described, there is a third generator, $G_{B>C}$, whose goal is to generate depth maps from synthetic images. The generated depth maps are compared to the real ones and are fed to a discriminator, D_C, resulting in two losses for training $G_{B>C}$. The objective is to achieve a working sequence where A (simulated domain) is transformed to B' (real domain) and, posteriorly, to C', obtaining the depth map.

Murez et al. (2018) performed domain adaptation between synthetic and real world images. The paper starts by pointing the main aspects in domain adaptation, already mentioned before. The first is to achieve a domain agnostic feature extraction in the form of a latent space, Z, that will allow to extract features on both synthetic and real images equally. The second is cycle

consistency. And the third is the domain specific reconstruction, meaning that the latent space Z should be able to reconstruct back to the original domains. Practically, the architecture is very similar to the supra-explained but with differences in the loss function. Besides the cycle and adversarial loss, there is a reconstruction loss and a translation loss (that allows translation from the source domain to the space Z and further to the target domain and vice-versa).

Mou et al. (2019) tackled depth and pose estimation in the same model, resorting to style transference from labelled synthetic images. The paper reports the use of four segments: a domain transfer module, a segment for depth estimation, other for pose and a fourth to predict a moving mask used to smooth the loss when faced with moving objects, similar to the implementation of Godard et al. (2019). For domain adaptation they use a standard adversarial loss (GAN loss), a temporal consistency loss (to maintain geometric congruence) and a stabilisation loss.

Isola et al. (2017) tackled image-to-image translation with conditional GAN's. They use a U-Net based architecture, and for the discriminator they introduce PatchGAN that penalises structure at the scale of image patches, meaning that the discriminator loss is applied to patches instead of a single classification and fully connected layer are avoided. This particular part of the work can be integrated easily in similar domain adaptation developments.

Finally, the work of Gupta and Mitra (2019) is noteworthy for being conducted with underwater images which is the focus of the present research. With cycle consistency as basis, there is a generator G where a RGB underwater image, X, is translated to a RGB+Depth channel above water image, Y, and a generator, F, that converts above water RGB+depth images in RGB underwater images. The generators are constrained by adversarial loss and cycle loss and it is introduced the structure similarity (SSIM) loss that maintains similarity between the original and the translated images and between the translated image and the reconstructed image. Finally a sparsity of gradient loss is introduced to reduce unwanted textures in the depth maps (artefacts originated from the different textures of the different domains).

2.5 Multitask Learning

Multitask learning multitask learning is a research area whose goal is to use a single backbone model to perform multiple tasks. The methodology exploits task specific features that may achieve a mutual improvement when shared. Other advantage is the efficiency, as is its better to extract more information from a single model. The generalisation of the problem is also a positive point because the model tends to discard specific features of a task or sample that could otherwise provoke overfitting.

Several works perform multitask on segmentation, detection and classification as is the case of the supra-mention papers of He et al. (2017); Huang et al. (2019); Liu et al. (2018); Du et al. (2020). For the problem at hands the classification is unnecessary and the detection is similar

but less complete compared with segmentation. However, depth prediction could provide valuable information, for instance measurements and improved boundaries based on depth. Depth prediction and segmentation task could also share inter task features that could benefit each other. Therefore, various compositions related to multitask, particularly performing depth prediction and segmentation are reviewed bellow.

Nekrasov et al. (2019) used an encoder and a decoder with skip connections between called Chained Residual Pooling, composed of 2 max pooling and 2 convolutions. The output of the network is separated in two paths according to the task performed (depth prediction or segmentation).

Lin et al. (2019) compared two architectures, one with a shared feature extraction network that is separated in two paths for the individual tasks. After the extraction block the commonly extracted features are applied in different tasks. The method is simple and easy to train as most parameters are shared by both tasks. The other architecture has a feature extraction block that is used for both tasks and a Global Depth Net used only for depth prediction. The feature extraction block is concatenated with the Global Depth Net to introduce the extracted object information from the segmentation task into the depth task.

In a similar way to the first architecture used by Lin et al. (2019), Bischke et al. (2019) and Zou et al. (2020), employed a encoder-decoder architecture. However, while Bischke et al. (2019) only separated the paths for each tasks in the last layer, Zou et al. (2020) featured completely separated decoders for each task. Additionally, the latter has three fold multi tasking, including depth prediction, segmentation and boundary prediction, claiming that it would improve the boundaries in the other tasks as well. The decoder is shared by the three tasks while the encoder is divided for the tasks.

The research of Harb and Knöbelreiter (2019) makes use of stereo image pairs and passes both images (left and right) through a feature extraction block, obtaining several intermediate feature maps and semantic segmentation prediction. The feature maps extracted are then aggregated and regularised in a CNN block that produces a disparity map. The previous formed segmentation and the disparity map are jointly introduced to a decoder that performs a refinement in both predictions.

2.6 Depth Prediction

Some of the most notorious depth prediction works, namely Saxena et al. (2006, 2008b,a), were performed resorting to Markov Random Fields (Li, 1994) preceded by the extraction of hand-engineered features. However, with the rise of CNN, its use for depth prediction became widespread. Supported by the fact that CNN based approaches display the best results as the state-of-the-art, the following research projects are, consequently, related with deep learning methods. An overview of the results regarding depth prediction on KITTI dataset is available in table A.3 in the appendix section A.2.

Worth mention is also Simultaneous Localisation and Mapping (SLAM), related with the robotics field, that estimates depth and pose for automatic and real-time tasks. Structure from Motion (SfM), related with the computer graphics field, also estimates the depth and pose using optimisation processes where the resulting 3D model of the environment is the focus. A normal approach for these problems is to find a corresponding point in multiple images and, by knowing how much the camera has moved between frames (camera poses), it is possible to solve an optimisation term called geometric loss or photometric loss to find the 3D position of that pixel.

Some of the works exemplified herein can be considered SLAM as they perform depth and pose estimation. However, for the sake of particularisation, the research will be focused, as mentioned above, on deep learning methods for depth estimation.

As mentioned before, depth may be important as a way to retrieve morphological information from the fish, justifying the research presented herein.

2.6.1 Datasets

To test the quality of the applied methods it is necessary to have depth labelled datasets. For outdoor environments there is a limited number of datasets due to the expensive use of LiDAR for depth capturing. In the current section the most prominent datasets are presented for indoors and outdoor.

Cityscapes, described in section 2.3.1.1 can be part of this group as well.

2.6.1.1 KITTI

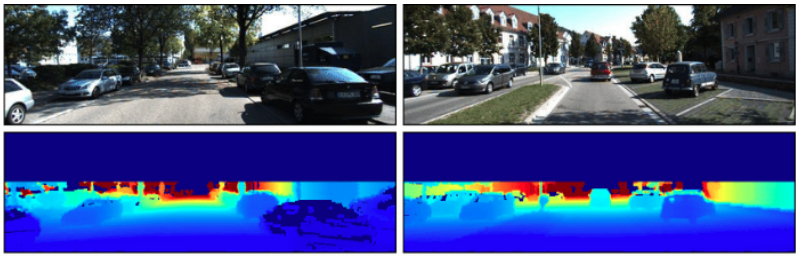

Figure 2.17: RGB and depth examples for KITTI dataset.

KITTI dataset is the most prominent resource for stereo, optical flow, visual odometry, 3D object detection and 3D tracking, and depth estimation tasks. This dataset is captured by driving around cities in rural areas and highways. The LiDAR does not capture information for the upper part of the image which leads to faulty depth information on that part. The KITTI dataset also has several splits according to the tasks. Figure 2.17 shows an example.

2.6.1.2 NYU V2

This dataset (Silberman et al., 2012) is largely used for supervised and unsupervised approaches (example in figure 2.18).

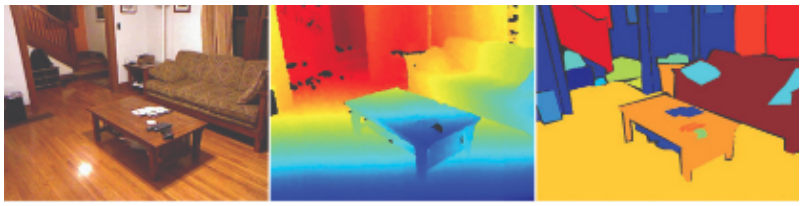

Figure 2.18: RGB, depth and semantic segmentation from NYU dataset.

It has 1449 (640×480) densely labelled pairs of RGB and depth images captured on 464 diverse indoor scenes. The depth acquisition is made resorting to Kinect camera from Microsoft. Normally the dataset is split into 795 training images and 654 test images. The depth ranges from 0 to 10 meters for this dataset.

2.6.2 Evaluation Metrics

There are many evaluation metrics commonly used for depth evaluation, which is a regression problem.

- **Mean relative error** (Rel): $\frac{1}{|T|}\sum_{d\in T}|\hat{d}-d|/d$
- **Mean** log_{10} **error** (log_{10}): $\frac{1}{|T|}\sum_{d\in T}\left|\log_{10}\hat{d}-\log_{10}d\right|$
- **Root mean squared error (RMSE) linear**: $\sqrt{\frac{1}{|T|}\sum_{d\in T}\|\hat{d}-d\|^2}$
- **Root mean squared error (RMSE) log** : $\sqrt{\frac{1}{|T|}\sum_{d\in T}\|\log\hat{d}-\log d\|^2}$
- **Thresholded accuracy**, which corresponds to the percentage of pixels with the predicted values within a certain threshold of the real value: % of d_i s.t. $\max\left(\frac{\hat{d}_i}{d_i},\frac{d_i}{\hat{d}_i}\right)=\delta<thr$

2.6.3 Supervised Depth Prediction

Several works have used hierarchical Conditional Random Fields (CRF) in conjunction with CNN for depth prediction (Wang et al., 2015; Liu et al., 2015; Eigen and Fergus, 2015), however using a fully CNN based model is simpler to train.

He et al. (2018) developed a CNN based on the VGG architecture (Simonyan and Zisserman, 2014), however they used skip connection between the correspondent convolutional and upconvolutional layers to exploit the middle layer features. It is also described a mechanism to self-generate varying-focal-length datasets from fixed-focal-length datasets, and the focal length information is fed to the middle fully connected layer of the network proving that varying focal length may help in depth prediction.

Xu et al. (2018) used a continuous CRF to integrate multi-scale information from different layers of a CNN. This method is fully integrated with a structured attention model network that oversees the amount of information that is transferred between different scales.

Finally, Fu et al. (2018) achieved the most competitive results in the supervised methods by introducing Deep Ordinal Regression Network (DORN). The problem is treated as a classification

task by discretizing the depth values into k intervals. The discretization is not uniform because for higher depth values the uncertainty increases. For each pixel, its probability of belonging to a k_i interval is calculated as a ordinal regression problem.

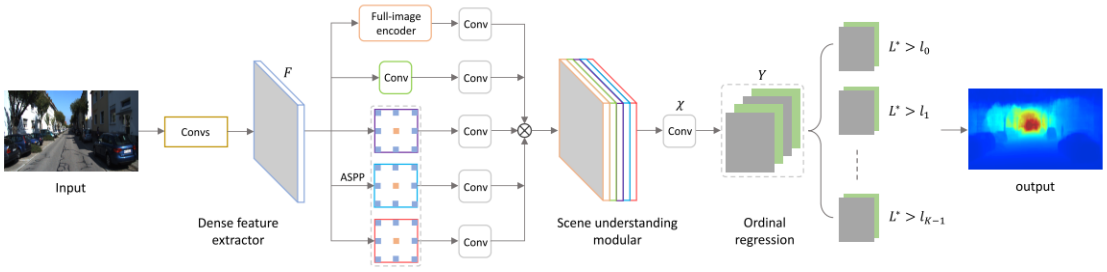

Figure 2.19: Fu et al. (2018) model architecture. For each pixel x, k probabilities of $L^* > \mathrm{l}_k$ were predicted, where L^* denoted the predicted depth for that pixel and l_k are the k discretized depths.

In figure 2.19 it is possible to visualise the architecture containing multi-scale capturing stage in parallel to a convolutional stage (1×1) and a full-image encoder stage, all joined by dilated convolutions with multiple dilation rates, which is one of their contributions as well. The final stage is a ordinal regression optimiser.

2.6.4 Semi-Supervised and Unsupervised Depth Prediction

The acquisition of depth information is time-consuming and expensive, this allied to the large data requirements in deep learning methods leads to the lack of depth labelled datasets. Consequently, the development of semi or unsupervised methodologies is forthwith a requirement. Some of the most popular and state-of-the-art research papers are reviewed herein.

In the semi-supervised field one of the most prominent works was conducted by Kuznietsov et al. (2017) who proposed adding the supervised and unsupervised loss RMSE in the final loss together. For the supervised part, the ground-truth is a LiDAR depth map and the unsupervised part is achieved with stereo-images aligning each pair by enforcing photo-consistency.

Pilzer et al. (2018) used a cycle GAN composed of two networks that supervise each other offering strong constrains. The first sub-generative network, G_l, generates a disparity map from the left image I_l and, using a wrapping function, the estimated right image, $\hat{I}_r$, is obtained. Comparing the estimated with the real right image, I_r, via reconstruction loss, it is possible to train G_l. The second sub-GAN, G_r, has the estimated right image $\hat{I}_r$ producing a disparity map that is used to obtain an estimated left image, $\hat{I}_l$, which is then compared to the I_l. The cycle forces $\hat{I}_r$ to be so close to the real one that it becomes possible to predict the left image from it. The final result is a linear combination of the two GAN's.

Following their former work, Pilzer et al. (2019) proposed a first network with a single view image as input, capable of predicting the disparity map with the goal of acquiring the opposite view (left-to-right). Afterwards, forming a cycle, the predicted image is used to generate a disparity

map used to produce a reconstruction of the original input image, just like described in Pilzer et al. (2018). Additionally to the former work, there is a third CNN that explores the cycle inconsistency between the original input and its reconstruction. The authors believe that inconsistency maps may contain convenient information to exploit the failures of the first two networks. In the paper, the authors name the cycle network as the *student* network and the inconsistency exploring network as the *teacher* network. To conclude, using knowledge distillation, detailed in Hinton et al. (2015) as a methodology to compress a large network to a smaller one, the *student* network is trained so that its outputs match those of the *teacher*.

The work outlined above could be considered state-of-the-art for unsupervised methods however it depends on stereo images for training. Therefore, Godard et al. (2019) proposes an U-net based model trained with single, stereo or a combination of both images as supervision, increasing the freedom during training. Godard et al. (2019) addressed the problem of moving objects by filtering pixels that do not change from one frame to the other, similar to Zhou et al. (2017). Other existing problem was the commonly used photometric loss between consecutive images, just like implemented in Zhan et al. (2018), which introduces high error when a pixel in one frame corresponds to a closer object but, in the next, corresponds to a very distance object. The problem was addressed by masking those pixels in the photometric loss.

2.7 Summary and conclusions

Segmentation, domain adaptation and depth prediction have achieved outstanding results using CNN based models and unsupervised methods are showing promising results on pair with supervised methods. The majority of the work is in above water environments and recently the models have outstanding performance in available datasets. However, there is a lack of underwater datasets which translates in a lack of underwater works. Domain adaptation may solve the supra-mentioned problem since it can transfer knowledge between domains by making use of a domain agnostic feature space. This means that domain adaptation should be able to neutralise some of the disadvantages of working in a underwater environment and perform similarly to the approaches tested in above-water images. Cycle consistency seems the most promising method for domain adaptation with many adaptations of its own and it can be paired with existing methods for segmentation and depth prediction while multitasking can be an answer to obtain segmentation and depth in the same model. Finally, the work related to fish measurement is sparse and it uses mainly older techniques with only some recent work using state-of-the art segmentation techniques (Yu et al., 2020), however all of them perform measurements in controlled environments, sometimes even outside water as is the case of Yu et al. (2020).

Chapter 3

Fish Segmentation for Feature Extraction

The segmentation of the fish is a crucial step to collect information on the number of specimen and respective morphological attributes. This step will be primordial and a foundation for the following steps as it constitutes the core of this work.

The reviewed literature pointed out the advantage of deep learning based methods over others methods. Therefore, the task will be performed with resort to deep learning techniques, where the use of large datasets is almost mandatory to achieve a model with good performance and without overfitting.

Due to the small size dataset and the lack of segmentation masks, it is necessary to increase the number of available images. For that, the simulated dataset will be incorporated in training using methodologies expanded upon the works of domain adaptation in Chapter 2.

To construct a baseline model for comparison before addition of the simulated dataset, 354 mask annotated images and 846 bounding box annotated images of the real dataset were used. With this in mind, several methodologies were tested:

- Segmentation using 354 images with segmentation masks to use as a baseline.
- Segmentation using 1200 images, with 846 having only annotated bounding boxes in order to observe if the bounding box annotation improve the results.
- Segmentation including simulated data as simple data augmentation in the previous model, with the objective of observing its contributions.
- Applying different domain adaptation methods to enlarge the usable data.
- Using multitask learning to ascertain its contributions.

3.1 Datasets

The first dataset is a subset of a larger dataset initially acquired by EMEPC [1] for a fish detection problem. The larger set contains sequences of videos with different species of fish acquired with a

[1] https://www.emepc.pt/. (Access:11-06-2020)

Remotely Operated Vehicle (ROV) [2]. The video dataset was acquired in two different time stamps and with two different cameras: one with a 1440 x 1080 resolution and the other with 720 x 576.

The second dataset is a simulated dataset acquired in Abyssal simulator [3].

3.1.1 Real Dataset

The image set used is a subset of the previously mention dataset with images of Rock-fish (*Helicolenus*), which is a genus of identical fish that were the most abundant in the original dataset. The partition of the subset is due to the nature of the problem, as, normally, the fish farms focus in a single species. The subset has images from 41 different sequences of video with 1412 images in total. The images are annotated with bounding boxes for each fish, however for a more specific annotation, 510 images were annotated with segmentation masks to help in training and to allow better validation of the results.

Figure 3.1: Example of the real dataset with 354 segmented images (top) and 846 annotated with bounding boxes (bottom). Left: annotations. Right: RGB image.

The target (real) dataset was divided in:

•**Train set** - with 1200 images from 36 sequences, of which, 354 have been annotated with segmentation masks and 846 only have bounding boxes (Figure 3.1).

•**Validation set** - with 101 images from 3 sequences, all with annotated segmentation masks.

•**Test set** - with 109 images from 2 sequences, all with annotated segmentation masks.

[2] https://www.emepc.pt/rov-luso. (Access:11-06-2020)
[3] https://abyssal.eu/products/simulator/. (Access:11-06-2020)

3.1.1.1 Simulated Dataset

The real dataset is very limited in numbers and in annotations, therefore a second dataset was acquired in Abyssal simulator to increase the size of the dataset as well as the richness of its information. Domain adaptation has been revolutionary in some fields, creating opportunities for simulated datasets to be used to create models and extrapolate them to real situations. The works of Atapour-Abarghouei and Breckon (2018); Jay et al. (2017); Hoffman et al. (2017); Ganin and Lempitsky (2015); Murez et al. (2018); Godard et al. (2019) are examples of that.

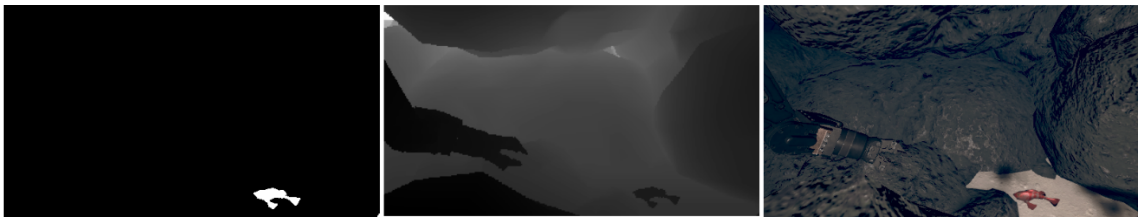

Figure 3.2: Example of the simulated dataset. Left: segmentation mask. Centre: depth information. Right: RGB image.

Consequently, a few video sequences were collected in an environment mimicking that of the real videos with a resolution of 532 x 299. As a result, the simulated dataset has 6200 images with segmentation annotations and depth maps (Figure 3.2).

Before feeding the images to the network, it was taken in consideration a depth threshold based on the visibility of the RGB image. The RGB image has incorporated an haze effect to mimic underwater images, nevertheless, the depth map is always the same, whether the RGB image has high visibility or not. For that reason it was applied a threshold in the depth map with a reasonable distance of 50 meters, which is a standard regarding depth prediction studies and encompasses most real case scenarios.

3.2 Pre-processing

No major operations were applied before training, nonetheless, a few were required. First, given the different resolutions of the images, all were resized to 512 x 256. Secondly, due to the small size of the target dataset, data augmentation operations were attempted. Models trained on small datasets usually do not generalise data well enough, overfitting to the training data and leading to poor performance on the validation and test sets. Thus, data augmentation can prevent learning of irrelevant feature patterns from a small dataset, boosting overall model performance. Taking this into consideration, traditional data augmentation transformations where applied to the training target dataset: horizontal flipping, random rotations (between -25^{o} and +25^{o}) and random scale variances (between -0.05 and +0.05). The operations are performed online, and are randomly applied to each image and mask, meaning that the model sees a slightly different image each time, which should translate in the extraction of invariant features when faced with the operations mentioned before. Figure 3.3 has two examples of the online data augmentation.

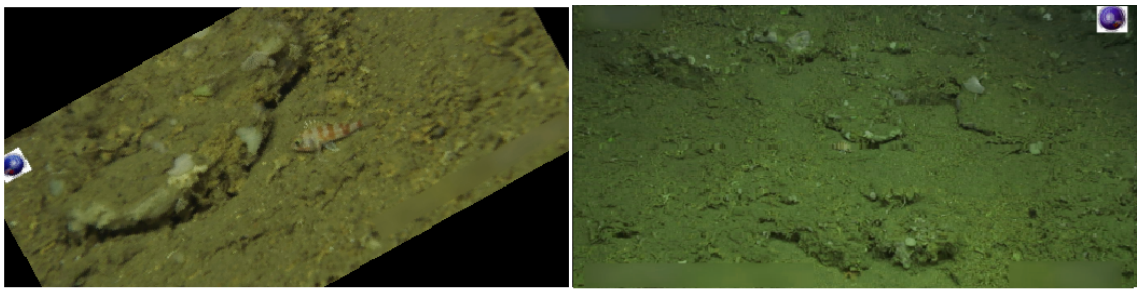

Figure 3.3: Example of the data augmentation operations. The rotation and horizontal flip is visible in the left image, while the scale transformation is visible on the right.

In the following sections regarding the experiments, the performance when applying data augmentation is compared with the performance when not applying it.

3.3 Segmentation Model Training

Resorting to the literature reviewed in section 2.3, various options were considered. Mask R-CNN (He et al., 2017), despite being present in several works with surprising results, has a heavy load computationally speaking and the class prediction together with bounding box prediction are unimportant and can be excluded for the task at hand. Therefore a simpler model, U-net (Zhang et al., 2015), was implemented, proving to be capable of learning during training. To decrease the difficulty of the problem, since the dataset contains a small number of fish per images with few overlapping fish, the model was designed for semantic segmentation, knowing that further post-processing methods can be applied to identify individual instances of fish.

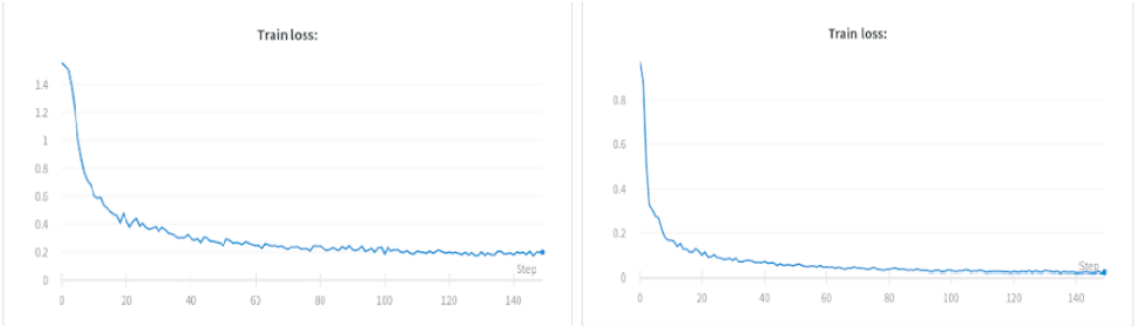

Figure 3.4: Comparison between U-net model (Left) and ResUNet based model (Right).

After comparing U-Net with ResUNet (Zhang et al., 2018), it was found that the latter showed slight improvements (Figure 3.4).

The model, described in figure 3.5, consists in four convolutional blocks (ConvBlock), each followed by a maxpool layer of 2 x 2 and, at the end, a fifth ConvBlock. The ConvBlock is a residual block, equal to the ResNet architecture of He et al. (2015), constituted by two 3 x 3 convolutions (stride 1 and padding 1), each followed by LeakyReLU with a slope of 0.2 and

BatchNorm . The residual connection is added in the end of block, transforming it in a residual block (described in figure 3.6).

In the expansion path, or decoder, the output of the transposed convolutions (UpconvBlock) are concatenated with the equivalent block in the encoder (skip connection) followed by a ConvBlock. The last layer is a convolution (1 x 1) and a sigmoid layer.

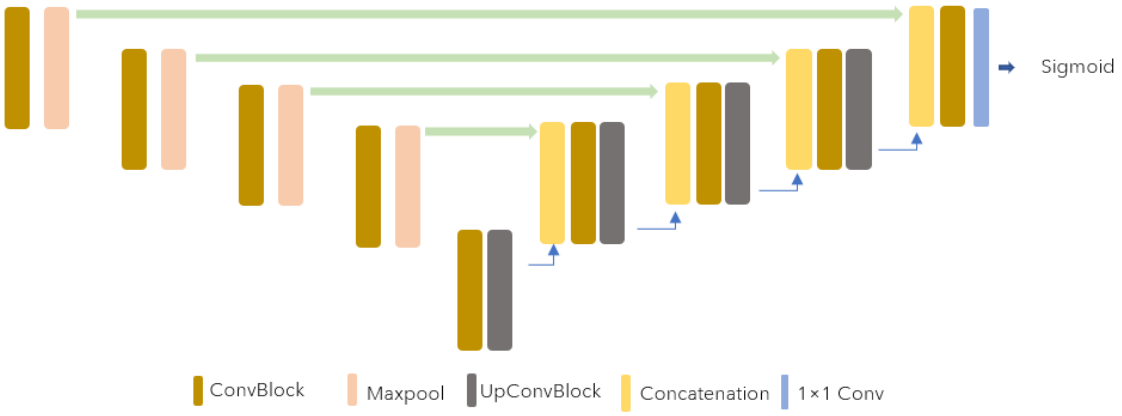

Figure 3.5: Architecture of the model used, based on ResUNet.

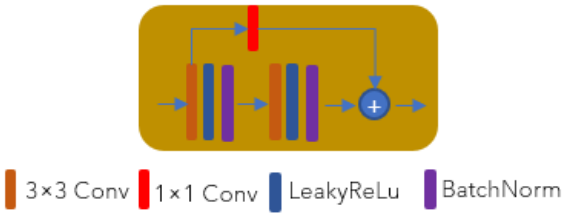

Figure 3.6: Convolutional block for the presented model.

The optimiser used in the network and throughout all the project was Adam optimiser. The learning rate for the segmentation model was 0.0001 and the batch size was 6.

3.3.1 354 mask annotated images

To initially evaluate the limits of the problem, a model based on ResUNet (Diakogiannis et al., 2020) was implemented, using the 354 images from the target domain with the correct, manually annotated segmentation masks. The model would serve as a baseline, representing the simplest form of training for the problem, but, nonetheless important to investigate its difficulties.

3.3.1.1 Training phase

The training loss based on the Jaccard coefficient, referred in section 2.3.2, equation 2.6, was chosen due to the mainstream use and achievements in related works. The loss is 1-Jaccard and, therefore, varies between 0 and 1, with 0 being the optimal result.

Figure 3.7 shows that the training loss decreases greatly in the first 5 epochs, which was expected given the small size of the dataset, which simplifies the training. Also as predicted,

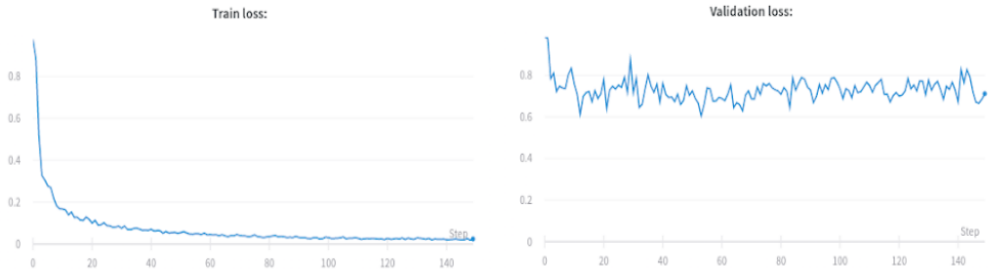

Figure 3.7: Evolution of training using 354 images.

overfitting is a problem revealed early on with the validation loss decreasing only during the first 5 epochs and keeping a high value throughout.

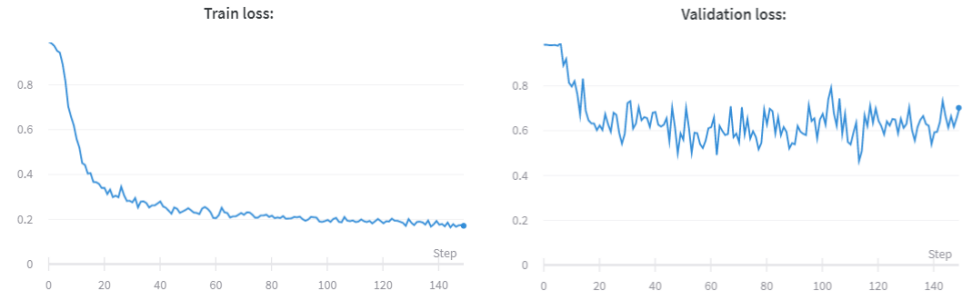

Figure 3.8: Evolution of training using 354 images with data augmentation.

Applying data augmentation operations provokes a mitigation in the training loss descent (figure 3.8. This is expected since the training set varies with each iterations. The validation loss is slightly minor, averaging 0.60 instead of 0.65 without data augmentation.

3.3.2 1200 bounding box annotated images

The target dataset has 1200 images in total, all with bounding box annotations. Although not complete segmentation masks, the bounding boxes could provide valuable information for the model and increasing the dataset is a valid hypothesis to better generalise the model.

3.3.2.1 Training phase

The annotations of the dataset do not correspond to the ideal ground truth for the problem, but the loss was adjusted to fit the requirements of the problem (segmentation) while considering the variables.

For the 354 fully annotated images the loss is a standard IoU and for the remaining 846 it is proposed a new loss. All bounding boxes encompass the object of interest, therefore IoU is a viable solution. Having said so, the area of the object is always smaller than the area of the

bounding box, meaning that, for a perfect segmentation, the IoU loss result when compared with a bounding box could never be zero.

Taking into consideration the supra mentioned, it is possible to calculate a value for the mean IoU loss that would contemplate the area difference. The 354 images with segmentation masks also have bounding boxes, making it possible to calculate the IoU between the actual masks and its bounding boxes. The value obtained indicates the mean loss a prediction needs to have in order to follow the area ratio averaged in the 354 images. That value was 0.3, resulting in a loss for the images with bounding boxes of:

$$L_BB = |IoU - 0.3| \tag{3.1}$$

Other values, from 0.2 to 0.5 were tested instead of 0.3, but the latter proved to be the most effective. Summarising, the expression above forces the predictions to have a IoU loss of 0.3, which translates in the correct area ratio calculated in the 354 images. For future reference, this loss will be called by BB loss or L_BB.

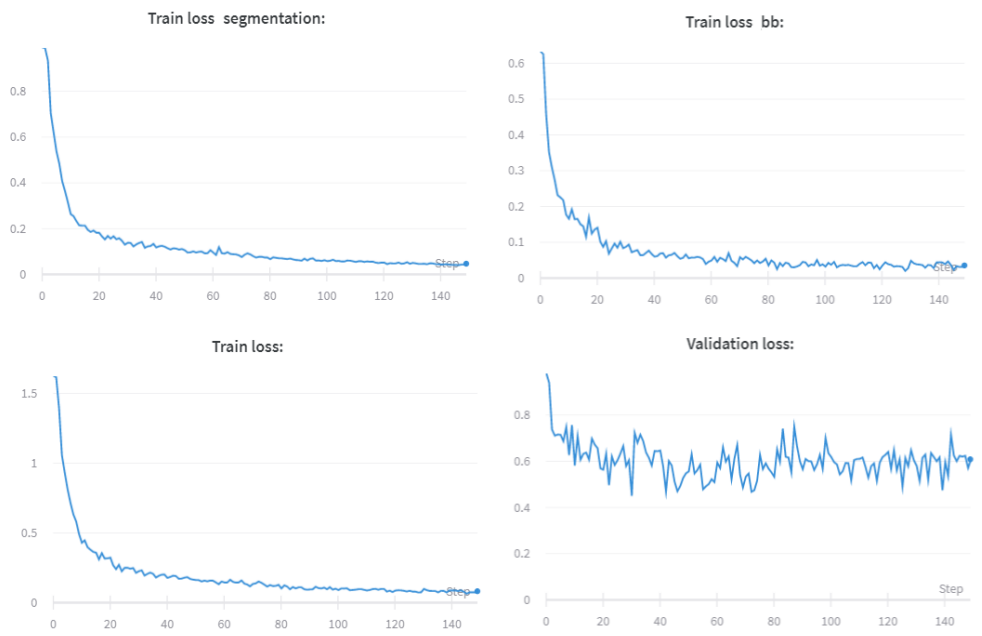

Figure 3.9: Evolution of training using 1200 images. The figures represents the loss for the 354 images (top left), the 846 (top right) and the total training loss (bottom left).

Figure 3.9 demonstrates that in the training phase, once more, the loss falls considerably in the first 5 epochs. The overfitting is noticeable after the same 5 epochs and the validation loss revolves around 0.6. Comparing to the results using data augmentation in figure 3.10, it is noticeable the smoother learning curve and the lower validation loss.

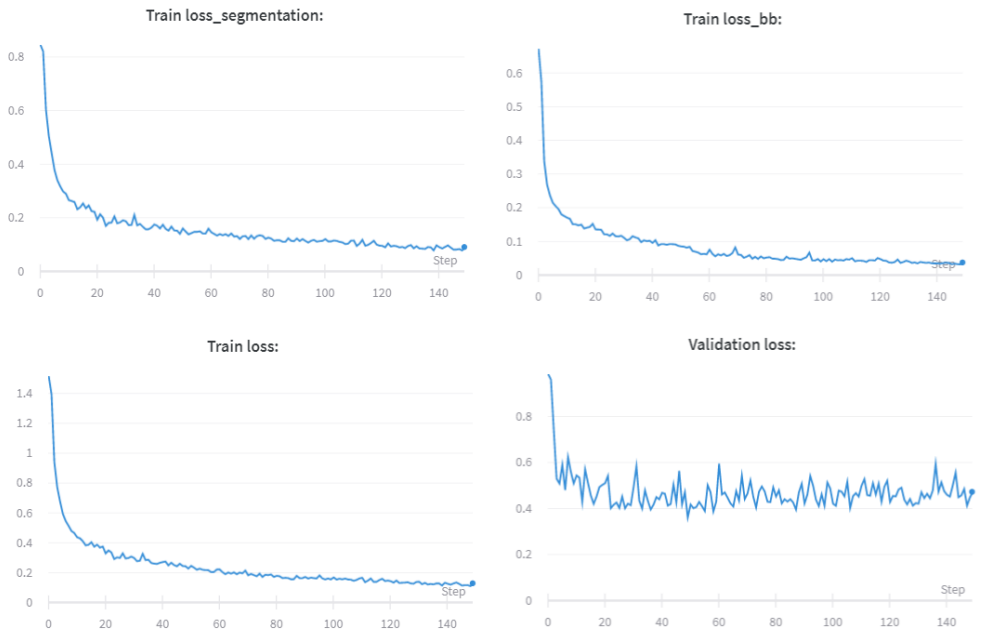

Figure 3.10: Evolution of training using 1200 images using data augmentation operations. The figures represents the loss for the 354 images (top left), the 846 (top right) and the total training loss (bottom left).

3.3.3 Progressive addition of simulated images

To ascertain if the simulated images could be used as supplementary material, they were added directly in the main frame of the model. By adding progressively increasing amounts of simulated images together with the real images, it is possible to evaluate its effect. Of course the two datasets represent two different domains, nonetheless, the simulated images could work as data augmentation even without domain adaptation.

With that in mind, a total of 25% increase in the dataset was tested, corresponding to 1200 real images and 300 simulated, after that 50% and finally 100%. The training loss progression between runs is compared in figure 3.11.

The learning curve is smoother compared with the curve without simulated images, similar to data augmentation results. The validation loss converges and stabilises around 0.6 when considering 25% and 50% of added images. If, in addition to the simulated data, data augmentation is introduced, the training behaviour is similar but the validation loss is slightly lower. Figure 3.12 shows the training progression using data augmentation, and, as in figure 3.11, the use of more simulated data aggravates the overall results, which is the reason for topping the simulated image numbers at 1200. A prominent aspect is the highly unstable validation loss along the epochs, not stabilising even with data augmentation.

This is a problem common to all tested models. It may be because of the small size validation, having its prediction varying greatly even with small changes of the model, or/and due to the high variability in the validation set, that features three different sequences of video but just 101 images.

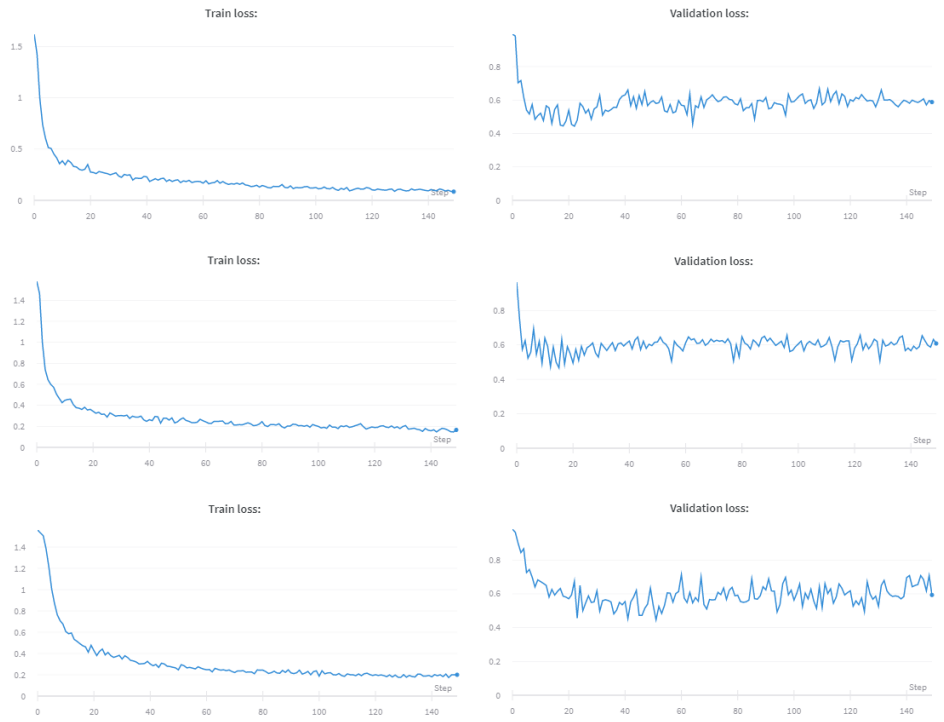

Figure 3.11: Training evaluation with 300 added simulated images (top), 600 (centre) and 1200 (bottom).

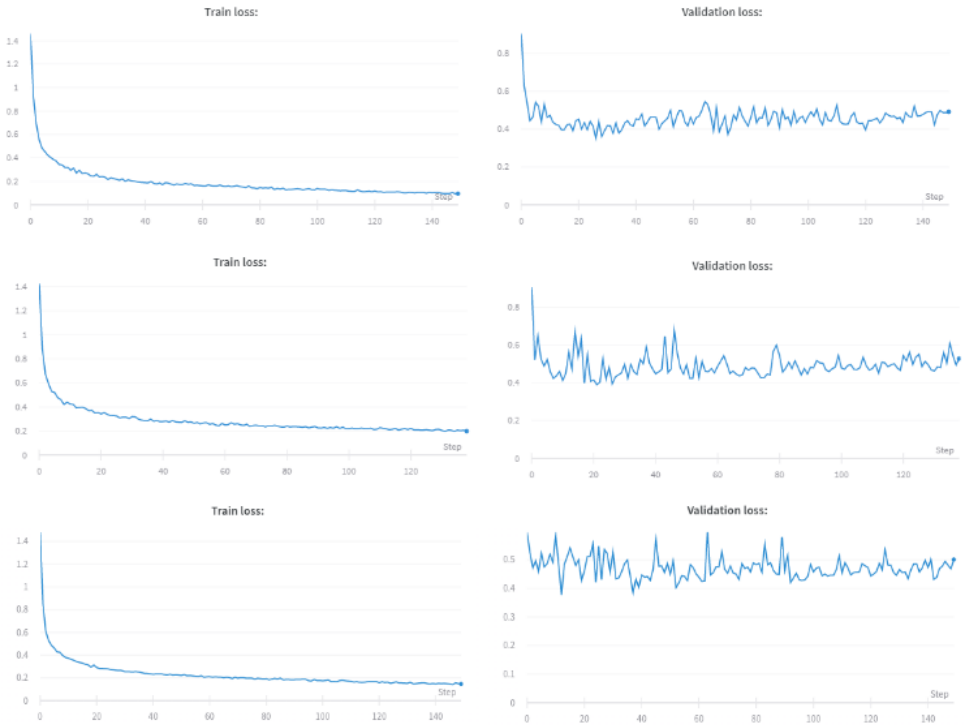

Figure 3.12: Training evaluation with data augmentation. 300 added simulated images (top), 600 (centre) and 1200 (bottom)

3.4 Domain Adaptation

In deep learning, large quantities of data are necessary and, for deeper networks, that necessity becomes clearer. Following the clues of previous positive literature about domain adaptation (section 2.4), some related methods were attempted. The goal was to use the entire simulated dataset to fully explore its advantages. For the domain adaptation models, no augmentation was performed as the constant changes in the dataset could jeopardise the domain transfer. All models here presented were trained with a learning rate of 0.0001, Adam optimiser and a batch size of 10.

3.4.1 Minimising distance between domains

Resorting to the simpler work of Ganin and Lempitsky (2015) it was decided to apply a similar method to the already developed architecture. The method of Ganin and Lempitsky (2015) can be implemented in any forward architecture, meaning its adaptation to the ResUNet is possible. With that in mind, a structure consisting of an encoder, a decoder and a constrain block was experimented.

The goal was to achieve a mid level feature space, f, that could be invariant to the domain while having a supervised loss that could predict for both domains. In their work the feature space, f, is obtained after introducing a constrain in that same feature space, which is between the encoder and decoder. Therefore, the skip connections between them had to be eliminated as they would bypass the constrains mentioned before.

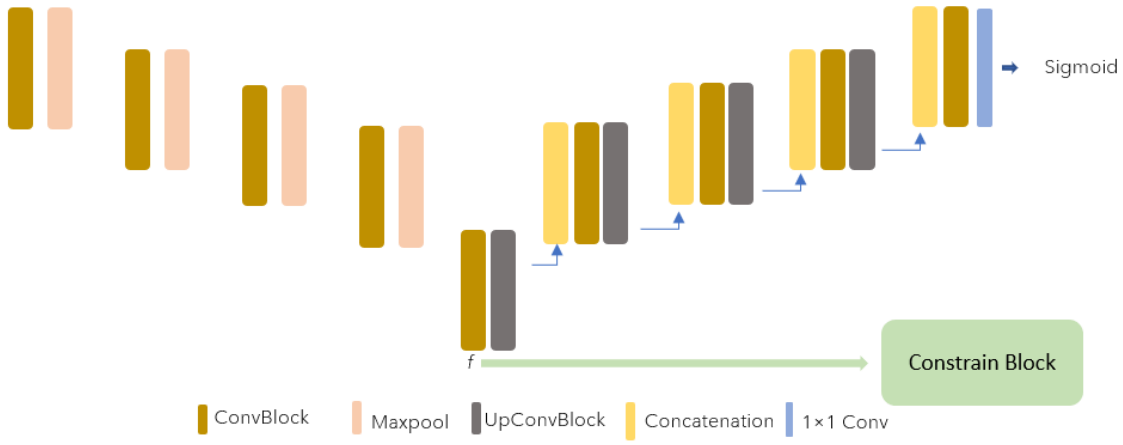

Figure 3.13: Modified architecture to allow domain adaptation.

The constrain block can be any function that approximates the two domains. Ganin and Lempitsky (2015) used a CNN for that task and trained in a end-to-end fashion, but in this work two other methodologies were tested as well. The general architecture is presented in figure 3.13, and the three different constraining blocks were:

- Discriminator block;
- Nearest Neighbour constructed with entire feature space;
- Nearest Neighbour constructed with mean and standard deviation per channel;

3.4.1.1 Constrain block with discriminator

Like Ganin and Lempitsky (2015), it was used a discriminator and a gradient reversal layer for backpropagation between the encoder and the discriminator, where the feature space, f, is the input. Unlike their work, the network is adapted to image segmentation.

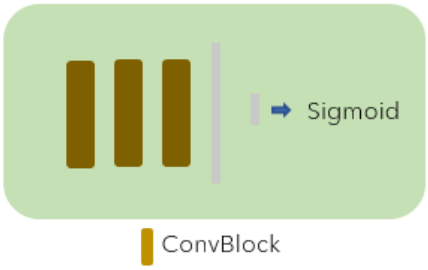

Figure 3.14: Constrain block with the discriminator architecture

The discriminator has the task of distinguish the domains of the images and, if it is successful in its task, the loss is small which translates into a greater loss for the encoder after the gradient reversal layer in the backpropagation. Hence, the discriminator obliges the encoder function to minimise the difference between domains in order to obtain a feature space invariant to domain. Figure 3.14 describes the architecture of the discriminator used. The domain discriminator has three 3 x 3 convolutions (stride 1 and padding 1), each followed by ReLu and BatchNorm, a flatten layer (2048), a fully connected layer (2) and a sigmoid activation.

The training loss for the discriminator is a Binary Cross Entropy (BCE) loss, that enforces the discriminator to learn the domain class of the images.

Figure 3.15 represents the training evolution. The training loss for segmentation has a behaviour similar to previous tests and is, therefore, omitted. The domain loss represents the class label loss of the discriminator and displays a modest decrease over time, suggesting the discriminator is learning more rapidly that the encoder is minimising differences between domains. The validation loss does not decrease throughout the epochs implying difficulties in generalisation, once again.

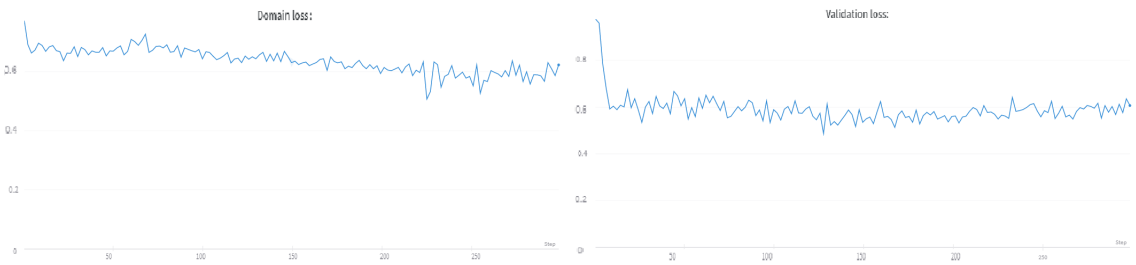

Figure 3.15: Evolution of training with the discriminator architecture for the domain classification loss and validation.

3.4.1.2 Constrain block with Nearest Neighbour

The second method for the constrain block consisted in a creation of an unsupervised k Nearest Neighbour algorithm, mentioned in section 2.1.1. During each epoch, the real dataset samples are stored to construct the data samples used with the kNN algorithm, considering that all 1200 images pass through the network per epoch. In the following epoch every simulated data outputted in the decoder is run through the algorithm in order to find the k nearest neighbours corresponding to real data samples closest to that sample.

Afterwards, the distance between the sample and the neighbours found is measured and works as a loss for the network by minimising it. The loss is a simple MAE and it is applied between each simulated data sample and the mean of the k real samples from the kNN algorithm. The goal is to minimise the inter domain distance in the feature space, f, by approximating simulated instances with the instances in the real domain.

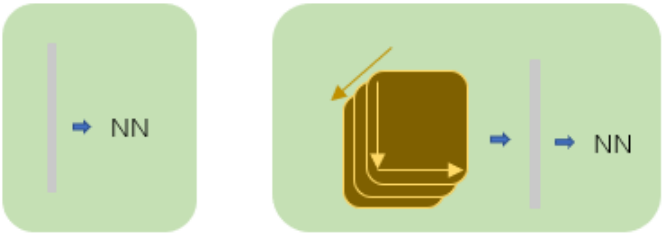

Figure 3.16: Constrain block using the flatten feature space as kNN instances (left) and using per channel mean and standard deviation (right) as input for kNN algorithm.

The feature space, *f*, has a resolution of 256 x 32 x 16, so two different forms were used to build the kNN algorithm. The first uses the full spectrum feature space flattened in a vector, seen in figure 3.16 (left), the other uses the mean and standard deviation calculated for each channel and for each pixel along the channels, represented in figure 3.16 (right). The result is 256 channels, each with one value referent to std and other to the mean, plus 2 channels (32 x 16) regarding the mean and std values for each pixel along the channel direction. The goal of this method is to capture information in the several dimensions of the feature space. The result is flattened and serves as input to the kNN algorithm.

One of the characteristics of kNN algorithm is the choice of the number of neighbours, k. To test the differences between a smaller and a larger k value, two different values were attributed to k: 10 and 1.

Comparing the progression during training in figure 3.17, it is clear that using the means as input provokes decreasing in the domain loss followed by an increase after around 150 epochs, deeming the model incapable of learning any further. The flatten feature space, however, has a slowly but steady descent and the correspondent validation loss is more stable, although it does not show sign of decreasing. The different values for k seem to have little to no effect in both cases.

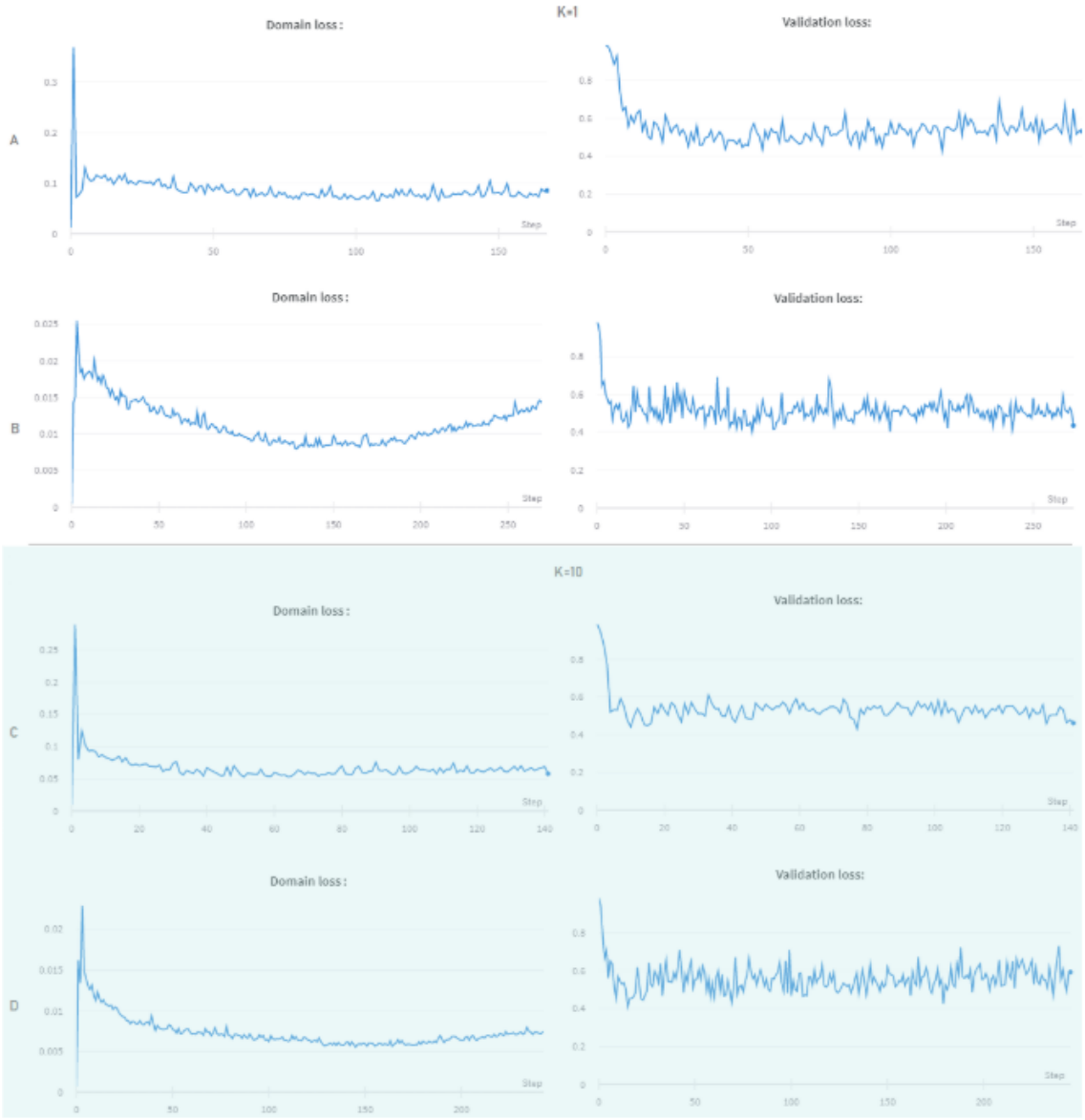

Figure 3.17: Evolution of training for kNN based models (k=1 and k=10). The rows A and C correspond to the use of the full feature space as input to the NN. The rows B and D correspond to the use of mean and standard deviation as input.

3.4.2 Cycle-GAN

Section 2.4 already describes the workflow of the cycle-GAN and, as a method specifically created for image-to-image translation, it fitted in the problem. As a first stage, the cycle-GAN network was implemented with slight differences. The generator architecture used was ResUNet and the PatchGAN methodology of Isola et al. (2017) was chosen instead of a single classification for the discriminator. Meaning that the discriminator classifies patches independently with unrelated classification but can still model high-frequency structures in the images.

The goal is to translate simulated images into real ones and vice-versa, in order to increase the

dataset. Additionally, it was trained a segmentation model on top of the real images together with simulated images after being translated to real ones.

After running for 300 epochs, with a batch size of 10, the model was extracted and used to obtain the transformed versions of the simulated images. The loss used for the GAN was Binary Cross Entropy [4] and MAE for the identity loss and reconstruction loss. The two last losses were multiplied by a factor of 5 due to their smaller values.

The simulated images transformed to the real domain were then joined with the real dataset, making a total of 7400 images and, finally, trained in the segmentation model of figure 3.5. The overall workflow is demonstrated in figure 3.18.

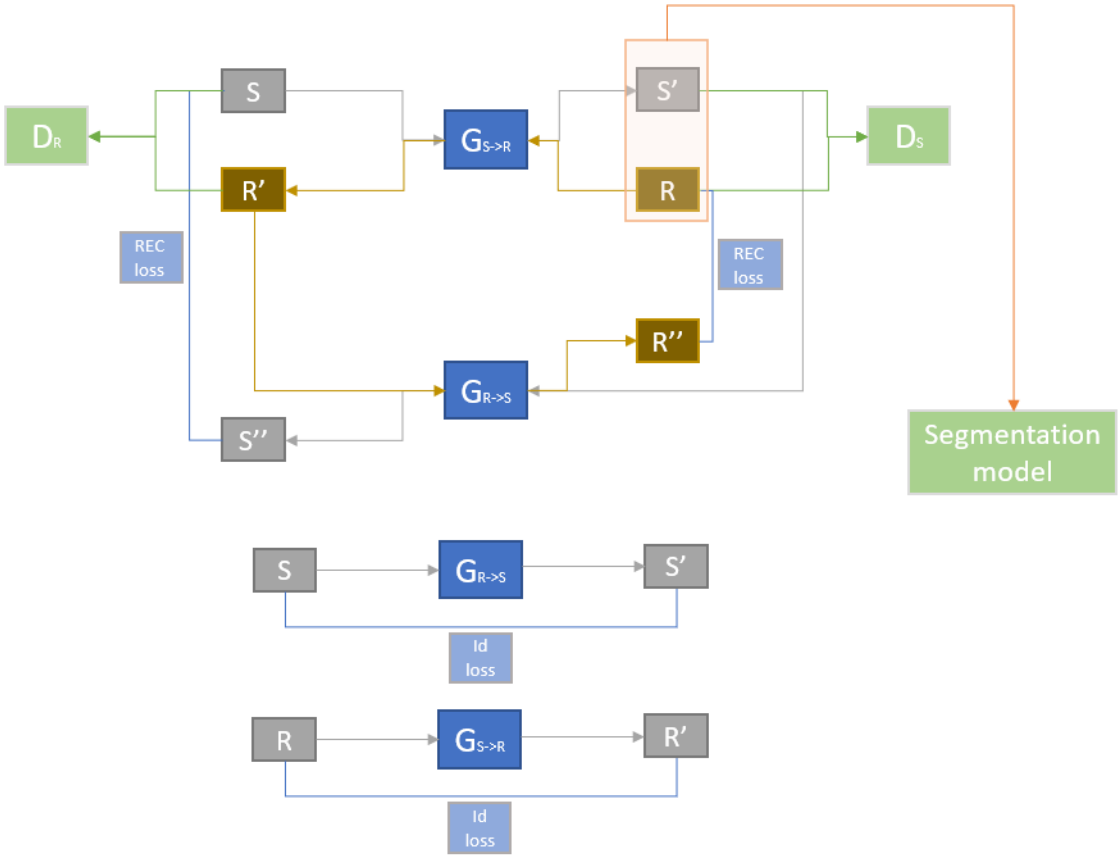

Figure 3.18: Workflow for the cycle-GAN based image translation. G represent the generators from one domain to another. D represent the discriminators. REC loss is the reconstruction loss and Id loss is the identity loss.

Similar to the previous tests, it was also introduced data augmentation during training to compare its impact. Figure 3.19 shows the evolution of the validation loss during training, revealing very little differences between them. The training evolution is on par with previous segmentation models presented before.

[4] https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss&-a-visual-explanation-a3ac6025181a, Accessed: 19-06-2020

A distinguish behaviour, however, is visible in the discriminators and generators losses. The more or less constants values around 0.69 in the discriminators indicate an ambiguous prediction, which is not favourable to the training as the discriminator simply manifest a 50% likelihood of correct prediction. Even when lowering the learning rate of the discriminators to 0.0005 (5 times lower than the generators) the problem persisted. The model still learns, however, is not as effective and the repercussion are visible in the outputted images.

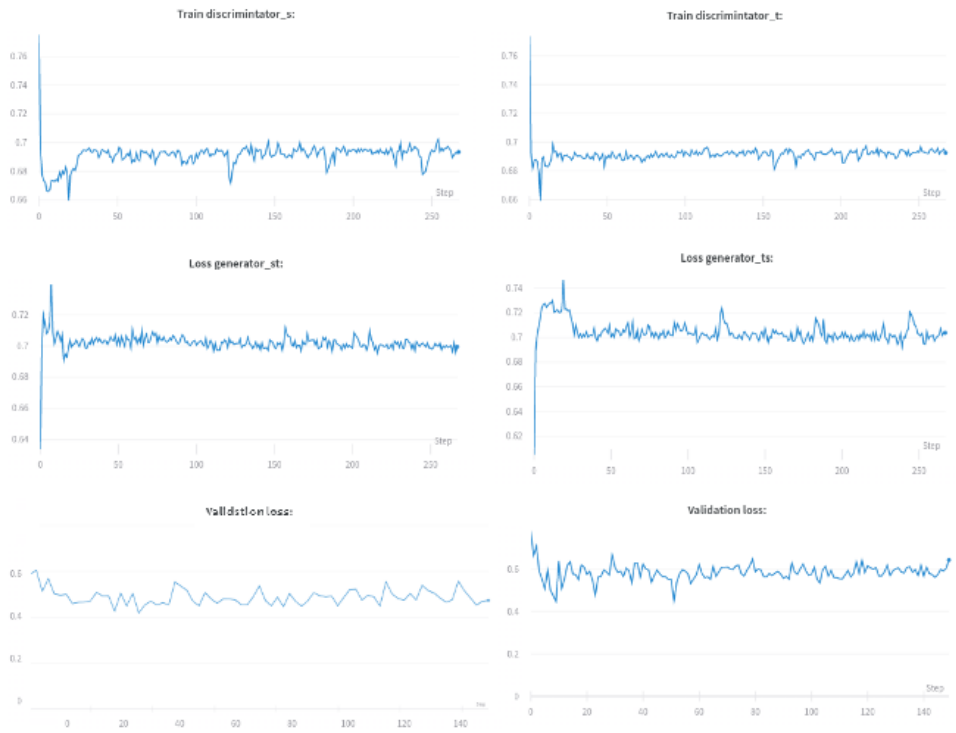

Figure 3.19: The first row corresponds to the discriminators loss, the second to generators and the last row is the validation loss evolution with data augmentation (left) and with no augmentation (right).

The cycle-GAN takes in consideration the distribution of the image and, for this case, the object of importance (fish), is just a small part of the image. The fish occupy less than 7% of the total area, meaning that the domain transfer could be considered accurate and yet, because of the small contribution of the objects to the overall distribution, it could still not be enough accurate for the object in particular.

A new strategy was devised based in the previous one. Instead of separately train the cycle-GAN and the segmentation model, they were trained together.

This strategy is a mirror of the previous one, however, instead of training the segmentation model on top of the results from the cycle-GAN, in each epoch the cycle-GAN is applied followed by a task network that predicts segmentation masks. The corespondent loss (which is the combination of IoU and BB loss) is backpropagated to the full network, influencing the domain transfer as well. The goal is to increase the significance of the fish in the image despite its small size.

Figure 3.20 depicts the segmentation loss for the training set and validation set along the epochs. The loss of the training is marked by spikes every 5 epochs, that is because the real dataset is only considered every 5 epochs to compensate for its smaller size compared with the simulated dataset. The step of 5 compensates for the ratio of the number of images in the datasets. Ideally, the datasets would be seamlessly integrated, but for training purposes the batches of real and simulated are separated. This practise is usual and does not affect training as seen in the figure.

The validation loss is very unstable, with values as high as 0.64 and as low as 0.32. The causes associated with this have been mentioned before, but here it is accentuated, probably because the input of the segmentation model is always changing as the training progresses.

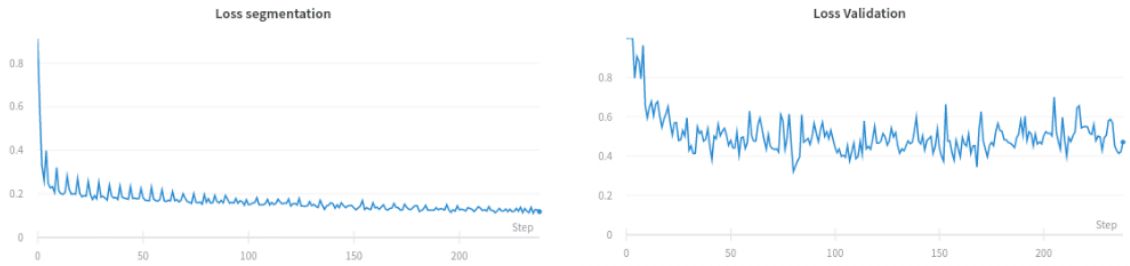

Figure 3.20: Training evolution for the cycle-GAN with task integrated.

3.4.3 Multitasking: depth prediction

The simulated dataset has depth maps paired with each image. To use that information in benefit of the problem, a multitask method was employed. Based on the literature review in section 2.5, a simple change in the segmentation model was implemented: two output channels instead of one. The latter allows to implement a constraint loss to one channel for segmentation and to the other channel for depth. The loss for depth was MAE and it was only applied in the simulated images as the real images do not have depth maps. The hypotheses is that depth information may help in the segmentation and, at the same time, provide depth predictions for the real set inferred from the results of the simulated set.

The tests were performed in the segmentation models with no domain adaptation using 600 and 1200 simulated images. The case with only 300 images was discarded as it would not contain enough depth maps.

Later, the same was replicated using the cyle-GAN model with the integrated task, being the only difference, once again, two output channels. For this model all simulated images were used, implying a bigger ratio of images with depth maps.

3.5 Fish Counting

The last step herein accomplished was the counting of fish per image. Fish counting can be important for fish farms and sea exploring as it allows to monitor the number of fish in a more rapid

way. To decrease the number of small objects, considered noise, it was exerted a morphological opening operation, followed by a morphological closing operation to close small holes inside the foreground objects. For the morphological operation the kernel size used was (5,5). Figure 3.21 represents the sequence of operations applied to a prediction example for the validation set.

Figure 3.21: Prediction (left); Morphology opening (centre); Morphological closing (right).

The aforementioned is followed by a structure analyses of the pixels. Those with 8-way connectivity between themselves are considered an object representation and are, consequently, counted as an object of interest.

3.6 Summary

First, after reviewing segmentation related literature, it was developed a viable segmentation model that could perform well with the dataset at hand. After experimentation, the ResUNet architecture proved to be a satisfactory candidate and was, therefore, used in following tasks.

Experiments using data augmentation were carried out to evaluate its implications. The progressive addition of simulated data also had in mind its contributions, and allowed to test if the simulated data in its original form was similar enough with the real data to help in generalisation.

After that, it was time to accomplish domain adaptation. The method presented by Ganin and Lempitsky (2015) was chosen for its simplicity, and the modified constrain blocks were a way to test other forms of approximating domains. The use of the kNN algorithm was based in works of Sener et al. (2016b,a); Tommasi and Caputo (2013).

As the domain transfer was not performing well, a new, more recent methodology based on cycle-GAN was implemented with two variations: the first uses the cycle-GAN in a traditional way, executing the transformations that are then fed in the task model; the other integrates the task in the cycle-GAN influencing the domain adaptation directly.

To attest the impact of multitask learning, 3 experimental cases were used based on the segmentation model and on the cyle-GAN+task model.

Finally, morphological operations followed by 8-way connectivity grouping applied in the predictions contributed to the extraction of the number of fish.

Chapter 4

Results and Discussion

4.1 Segmentation Model

Considering the model described in the last chapter, a comparison between the different results is presented in table 4.1. The goal is to study the effect of data augmentation, additional simulated data and the addiction of bounding box annotated data. The model for each example was chosen based on the best validation result during training.

Table 4.1: Comparison of results for the segmentation model with variations in the input dataset and the application of data augmentation.

Method	**Augmentation**	**Validation IoU**	**Test IoU**
354 real images	yes	0.57	0.18
	no	0.40	0.17
1200 real images	yes	**0.64**	**0.37**
	no	0.55	0.15
1200 real with 300 simulated	yes	0.64	0.32
	no	0.56	0.13
1200 real with 600 simulated	yes	0.60	0.34
	no	0.53	0.14
1200 real with 1200 simulated	yes	0.59	0.26
	no	0.55	0.27

It is clear that adding data, even when not correctly annotated, provides valuable information. As expected, the small dataset performs poorly and benefits from additional data with an increase of 0.15 for IoU (without data augmentation) in the validation set. For the testing set the model showed substandard generalisation capabilities with IoU values below 0.20, probably due the the dataset size, not even improving with data augmentation. On the other hand the loss implemented to deal with the bounding box annotations (L_BB) proves to be a resourceful method to lead the predictions to correctly segmented masks, at least, when paired with a smaller percentage of correctly annotated images.

Adding simulated images, however, proves to be inadequate with lower values both in the validation set and the test set. The difference between adding 300, 600 or 1200 simulated images is marked by a slight decrease in the mean IoU, suggesting that the simulated data distribution is different from the real data distribution, since it negatively affects the results. For 300 simulated images, the weight of the images in the model is less notorious, therefore the results are similar to the one obtain without any simulated images, but for 1200 images the difference is more apparent, as there are as many real images as simulated ones. Still, the results are not completely apart, indicating that the additional data is not excessively negative to the problem. With a proper domain adaptation technique, the added images may influence positively the results.

The data augmentation is an efficient tool in many works and in this one proved its value once again. The beneficial impact is notorious in the validation set, but specially in the testing set with improvements of over 0.2 in the IoU in the last three cases in table 4.1. The technique is useful in all cases except the one with 1200 simulated images. The large amount of simulated images may be overshadowing the effect of the augmentation.

Figure 4.1 verifies the overall better performance of the segmentation with augmentation and the better results for the model with only 1200 real images as well.

As for the testing set, there are many failed prediction, with several images having no detection, especially in the first sequence of video (figure 4.2, image *(h)*). The testing set has two sequences of video and both have difficulties associated. In the first sequence, with an example in figure 4.2, the fish is blurred and the images have a different hue from others, appearing more bluish than the rest of the dataset. The second sequence has many floating debris, invertebrates with similar colour to the fish and the fish have a smaller size when comparing with the rest of the dataset. In this case, it is standard the detection of other entities other than fish as seen in *(k)*, with only the best models softening the excessive detection, *(l)*.

Due to the differences in the testing set, the model does not generalise well as it wasn't able to learn features absent in the training set, resulting in high losses for the testing set. On the other hand, the validation set has more in common with the training set.

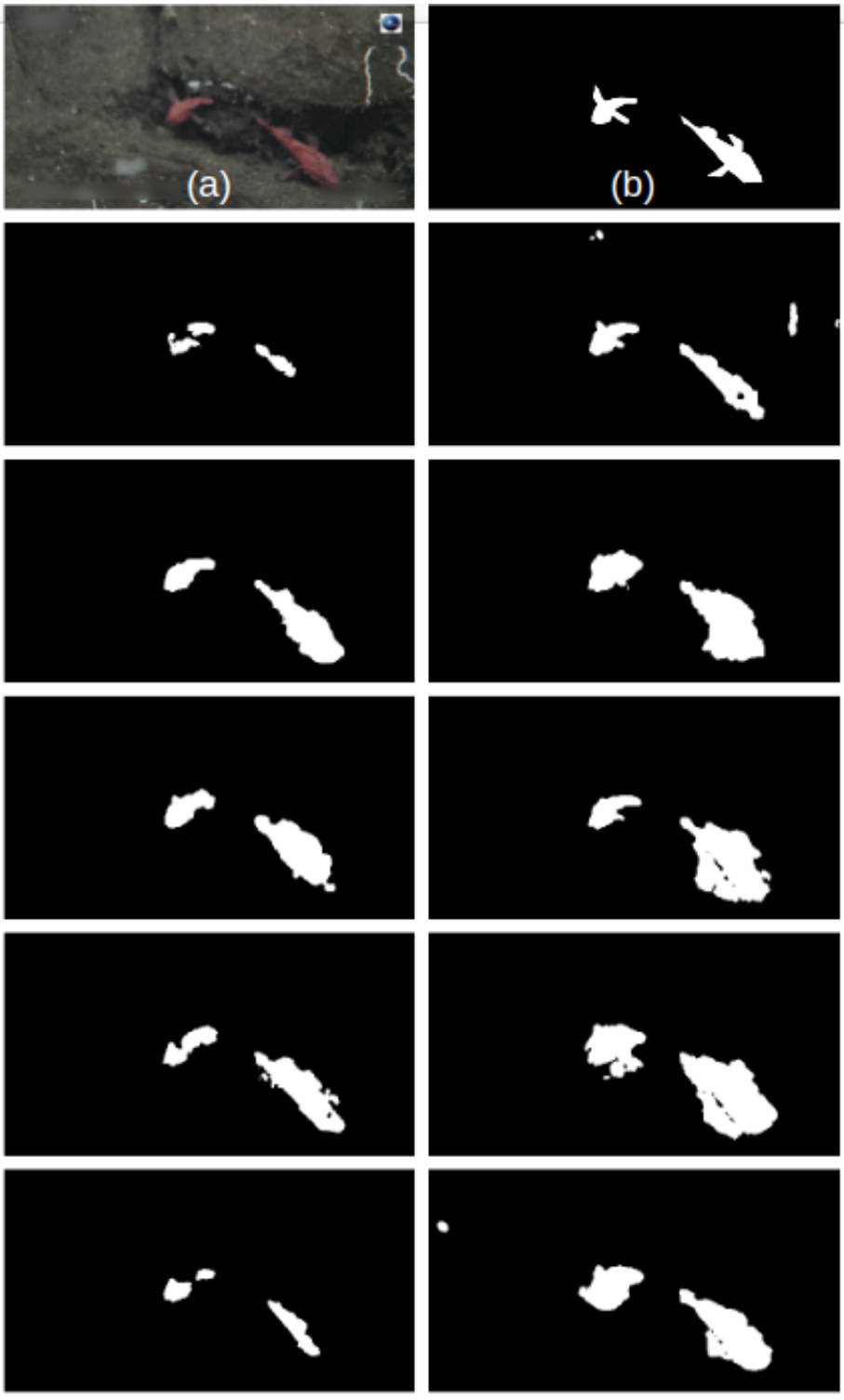

Figure 4.1: Prediction examples for the segmentation model in the validation set. The left column has no data augmentation, the right has. The first row corresponds to the RGB image (a) and the segmentation mask (b). From top to bottom, excluding the first row, the case description is: 354 images; 1200 images; 1200+300; 1200+600; 1200+1200.

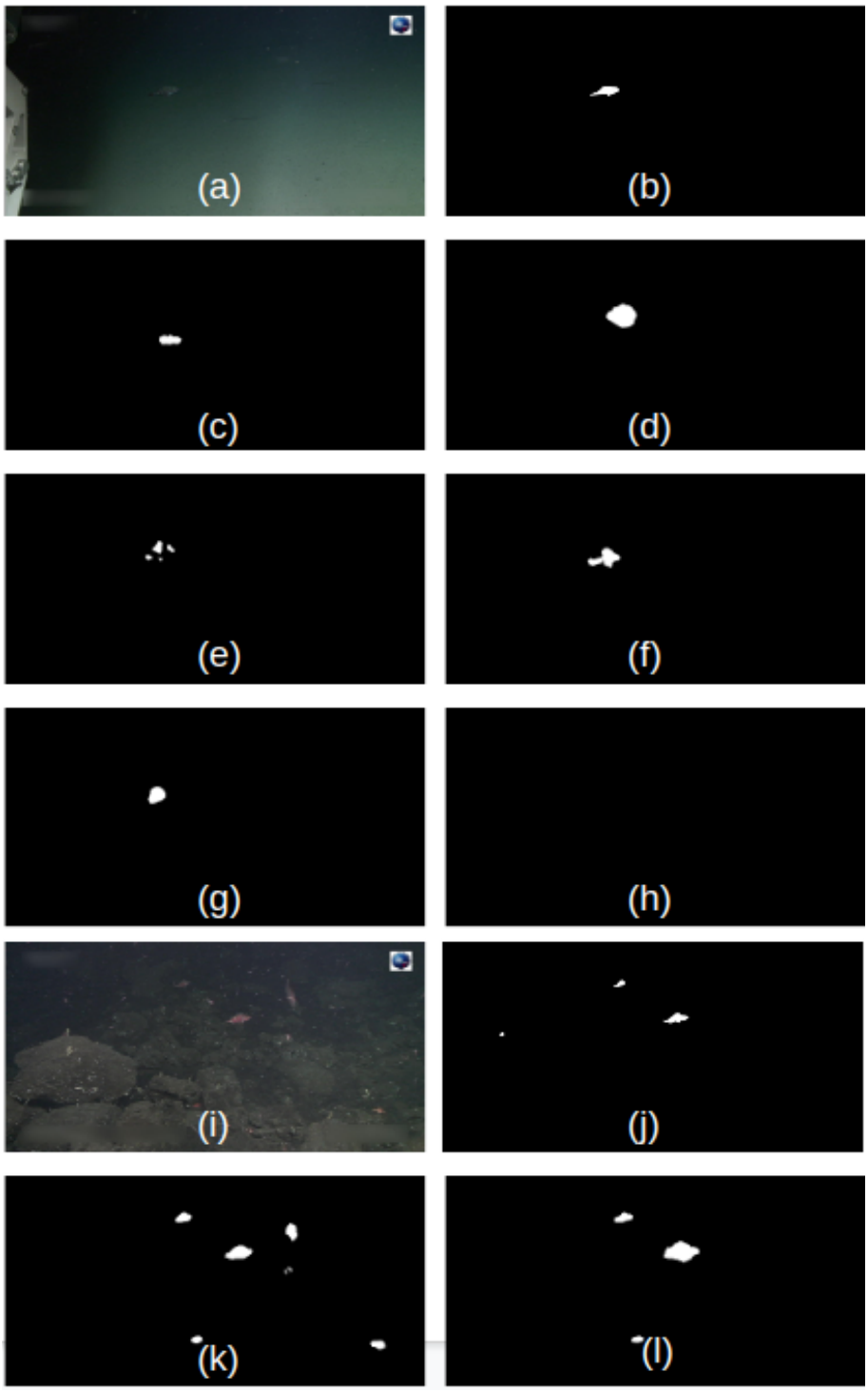

Figure 4.2: Prediction examples for the segmentation model in the testing set. (a) RGB image from sequence 1; (b) GT; (c) 1200 images with no augmentation; (d) 1200 images with augmentation; (e) 1200+300 images with augmentation; (f) 1200+600 images with augmentation; (g) 1200+1200 with augmentation; (h) represents all the prediction for below than 0.20; (i) RGB image from sequence 2; (j) GT; (k) representative example for higher losses (retrieved from the model with 1200 image and no augmentation); (l) representative example for lower losses (retrieved from the model with 1200 image and augmentation).

4.2 Domain Adaptation

Using domain adaptation techniques the results were expected to improve under the assumption that the sample numbers would increase. Theoretically, the model would generalise better, unfortunately, in practise this is not achieved.

Table 4.2 portrays similar values for different constrain blocks in both, validation and testing sets. The values for the validation set are slightly lower than the ones in table 4.1 and the testing results have always lower values, suggesting that the domain transfer is ill performed.

Table 4.3 reports the results for the cycle consistency models. The segmentation model trained with the transformed images is unsatisfactory. As the conditions of training are comparable to the ones in the previous section with the exception of the data, it proves that the dataset is not appropriated for the task and influences negatively the training. Figure 4.3 has examples of the transformed images from the simulated to the real domain, manifesting clear differences between them and further supporting the previous claims.

Table 4.2: Comparison of results for the domain model with variations in the constrain block. In total, 7400 images were used during training - 1200 real and 6200 simulated.

Model		**Validation IoU**	**Test IoU**
kNN vector	k=1	0.57	0.13
	k=10	0.57	0.25
kNN mean/std	k=1	0.60	0.28
	k=10	0.59	0.27
Discriminator	-	0.51	0.13

Table 4.3: Comparison of results for the cycle-GAN models. In total, 7400 images were used during training - 1200 real and 6200 simulated.

Model	**Augmentation**	**Validation IoU**	**Test IoU**
CycleGAN + Segmentation	yes	0.51	0.17
	no	0.55	0.13
CycleGan with task	yes	-	-
	no	**0.68**	**0.29**

For the model with the task integrated in the training of the domain, the validation mean IoU value is the highest of all. Looking for the table and disregarding the rest, it could be pointed that is was one of the best models and the integration influences the domain transfer in a positive way. However, the analyses in section 3.4.2 reveals the problem of this method: the instability. If the validation loss is unstable there is no guarantee that the test loss does not follow the same pattern, implying that, for a high value in the validation set, the test may not follow the same tendency.

Despite that, the model proved to achieve satisfactory results in the validation set and, although not the best in the test set it is not the worst. Additionally, when comparing the domain transfer

of the model with the integrated task and the simple cycle-GAN model in figure 4.3, the transformation is, qualitatively, better. The object of interest is more clear and, overall, the image is more similar to the real environment, proving that the integrated task increases the significance of the fish in the image despite its small size, as hypothesised before.

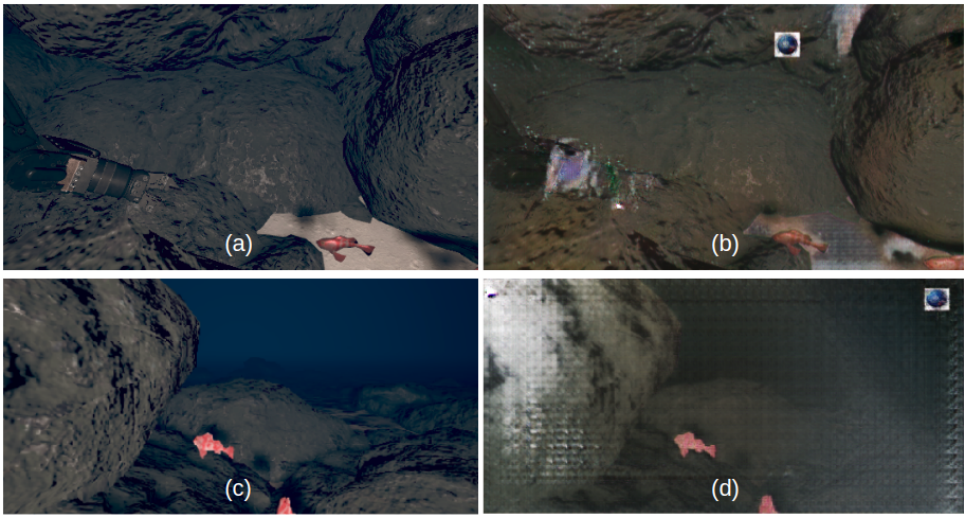

Figure 4.3: Example of an images resulting from the domain transfer cycle-GAN based model. (a) and (b) correspond to the original and transformed images, respectively, for the classic cycle-GAN. (c) and (d) correspond to the original and transformed images, respectively, for the cycle-GAN with integrated task.

The examples in figure 4.4 are compliant with the results. The image for the cycle-GAN with integrated task, *(f)*, is more similar with the ground truth, expressing the shape of the fish and the others have an equivalent performance.

The data augmentation when training the 7400 images does not contribute to the results and the image, *(h)*, as well as table 4.3, confirms that. The augmentation of an already unfit dataset does not improve results.

Figure 4.5 has representation of the predictions for the test set. The observation are interchangeable with the ones made in section 4.1. In the second sequence of the dataset the prediction continue to display false positives, with several miss identified objects. The existence of debris and other animals constantly miss-guides the model to identify non-fish objects. However, image (f), corresponding to the integrated task and cycle-GAN, shows improvements in the differentiation of fish from other object. This is complacent with the results in table 4.3 as it corresponds to the highest IoU value.

In the first sequence, the only model that detects the fish are the kNN based model with mean and std as input and the cycle-GAN with integrated task. All the others do not identify any object.

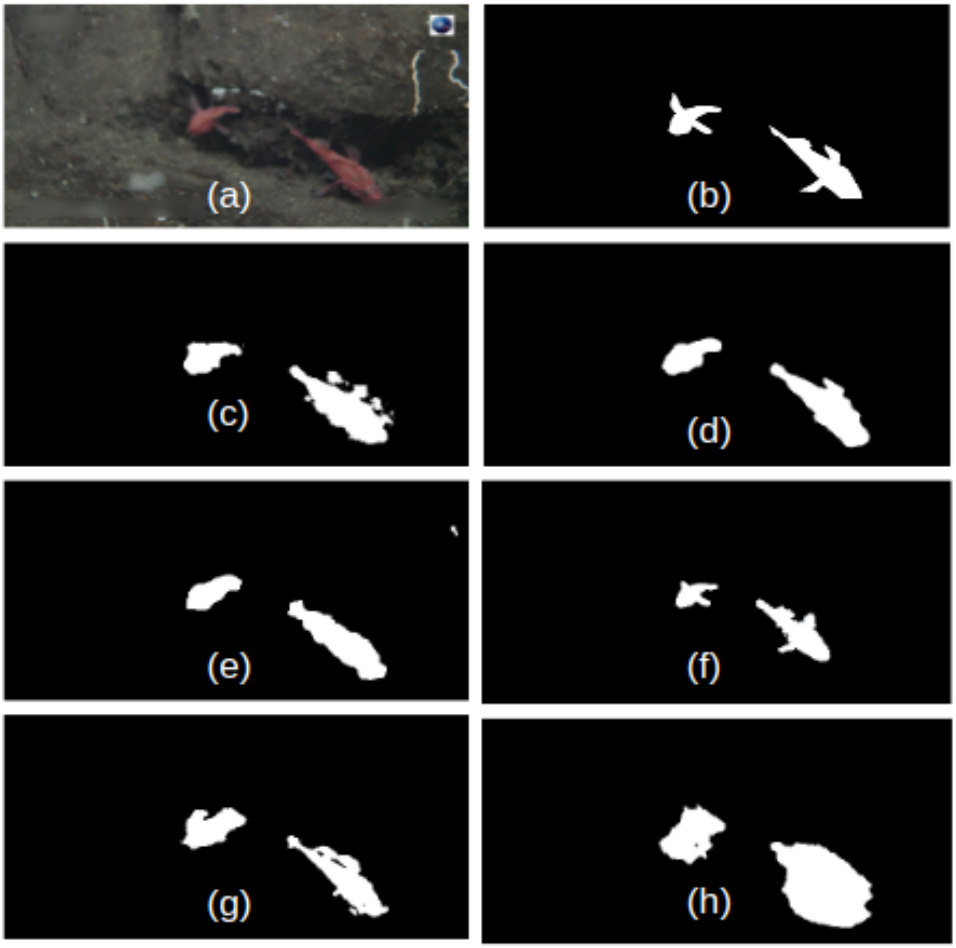

Figure 4.4: Example of the segmentation predictions resulting from domain transfer models in the validation set. (a) is the RGB image; (b) the GT; (c) the output of the model with the Discriminator in the constrain block; (d) represents the kNN with means and std's as input (representative of both k values), (e) represents the kNN with full vector as input (representative of both k values); (f) is the result for the integrated task and cycle-GAN; (g) and (h) correspond to the outputs of the segmentation model with the transformed dataset without and with data augmentation, respectively.

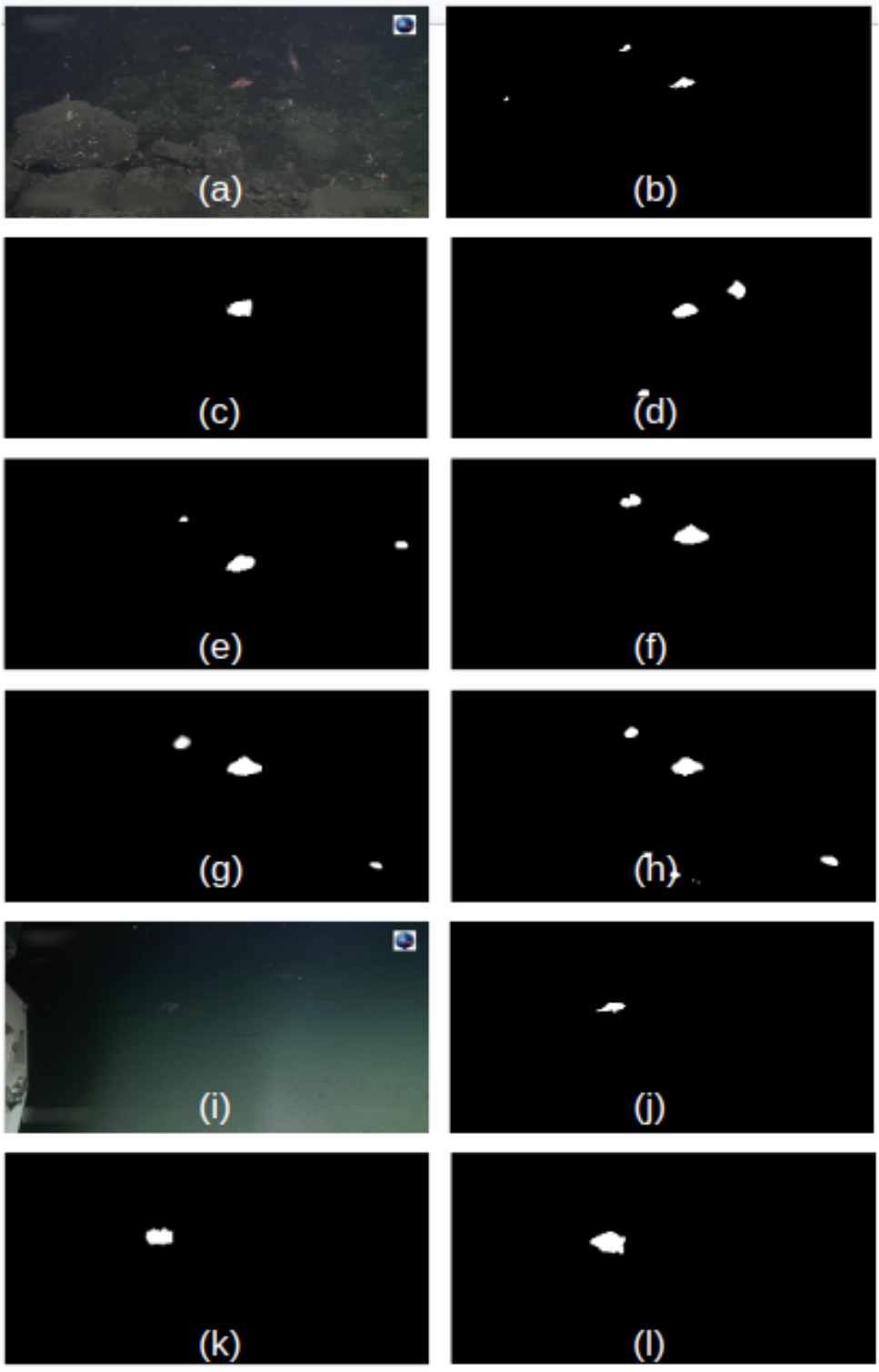

Figure 4.5: Example of the segmentation predictions resulting from domain transfer models in the test set. (a) is the RGB image of the second sequence in the test set; (b) the GT, (c) the output of the model with the Discriminator in the constrain block; (d) represents the kNN with means and std's as input (representative of both k values), (e) represents the kNN with full vector as input (representative of both k values); (f) is the prediction for the integrated task and cycle-GAN; (g) and (h) correspond to the outputs of the segmentation model with the transformed dataset without and with data augmentation, respectively; (i) and (j) are an RGB and GT example of the first sequence; (k) corresponds to the cycle-GAN with integrated task; (l) represents the kNN with means and std's as input (representative of both k values).

4.2.1 Multitasking

Three cases were tested using multitask learning based on the previous models. Using 600 simulated images the IoU for the validation was 0.59 and now it improved to 0.63 and the test from 0.26 to 0.28, but using 1200 simulated images the validation IoU decreased from 0.60 to 0.58 and in the test from 0.34 to 0.27. For the domain adaptation model the results increased from 0.68 to 0.69 in the validation set and remained the same for the test set.

The results do not allow to extract evidence of improvements (or the contrary) by using multitask learning, as they are similar to the ones obtain previously.

Nevertheless, the cycle-GAN+task model was the one that performed better overall, although just in the validation set.

The predictions in figure 4.6 and 4.7 further support the results, being clear the similarity with the previous attempts. As for the depth prediction, the quantitative evaluation is not possible, however, qualitatively, the predicted depth maps are completely unavailing, except for the one in image (j), figure 4.7 that presents something similar to a depth map with the closest points in black and the most distant, in white. It it even possible to identify the fish in the depth prediction.

This suggests that the model generalised and adapted depth map prediction from the simulated data to the real data for specific types of images. As mentioned before, the test set has differences from the rest, and the first video sequence of the set was the only one whose depth predictions depicted this behaviour.

Table 4.4: IoU results using multitask learning. The experiments in the segmentation model included data augmentation except the one based on cycle-GAN. The latter also included 7400 training images - 1200 real + 6200 simulated.

Model	**Validation IoU**	**Test IoU**
1200 real with 600 simulated	0.63	0.28
1200 real with 1200 simulated	0.58	0.27
Cycle with integrated task	**0.69**	**0.29**

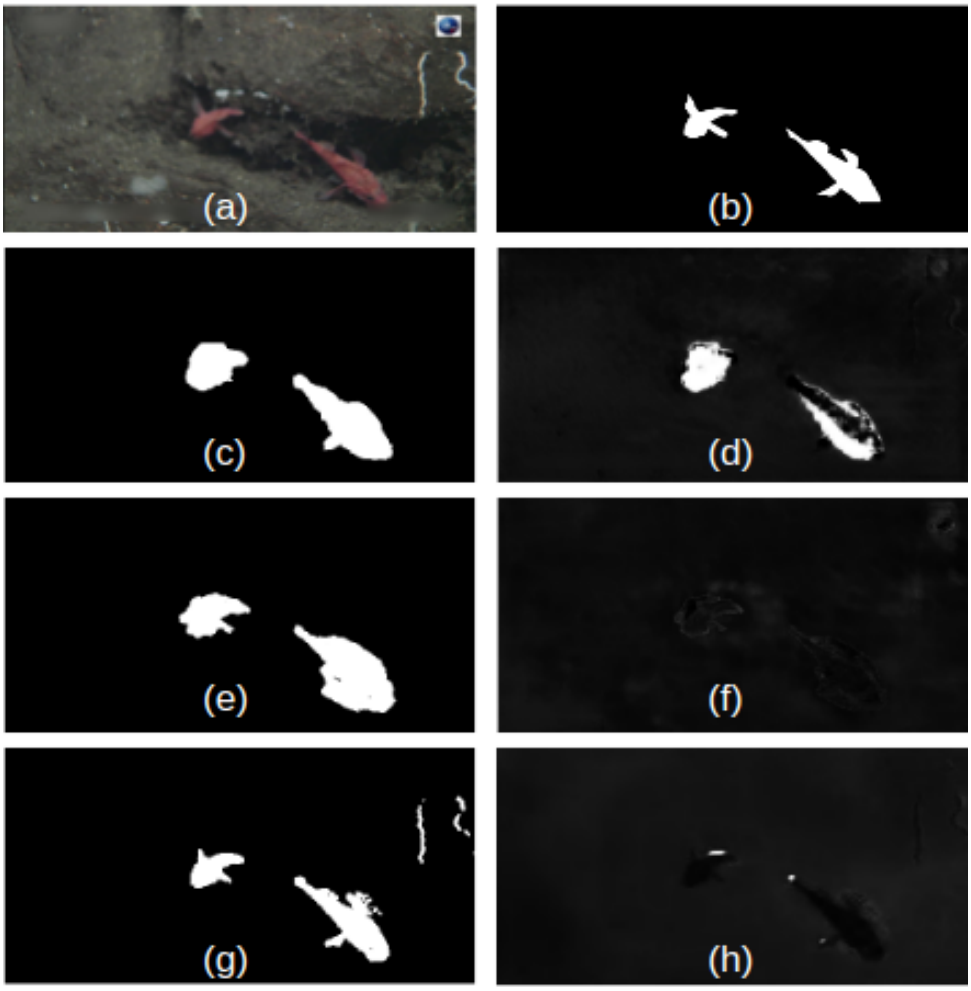

Figure 4.6: Example of the segmentation predictions for the multitask models in the validation set. (a) is the RGB image; (b) the GT; (c) segmentation prediction of the model with 1200 real + 600 simulated images; (d) depth prediction of the model with 1200 real + 600 simulated images; (e) segmentation prediction of the model with 1200 real + 1200 simulated images; (f) depth prediction of the model with 1200 real + 1200 simulated images; (g) segmentation prediction - cycle with task; (h) depth prediction - cycle with task;

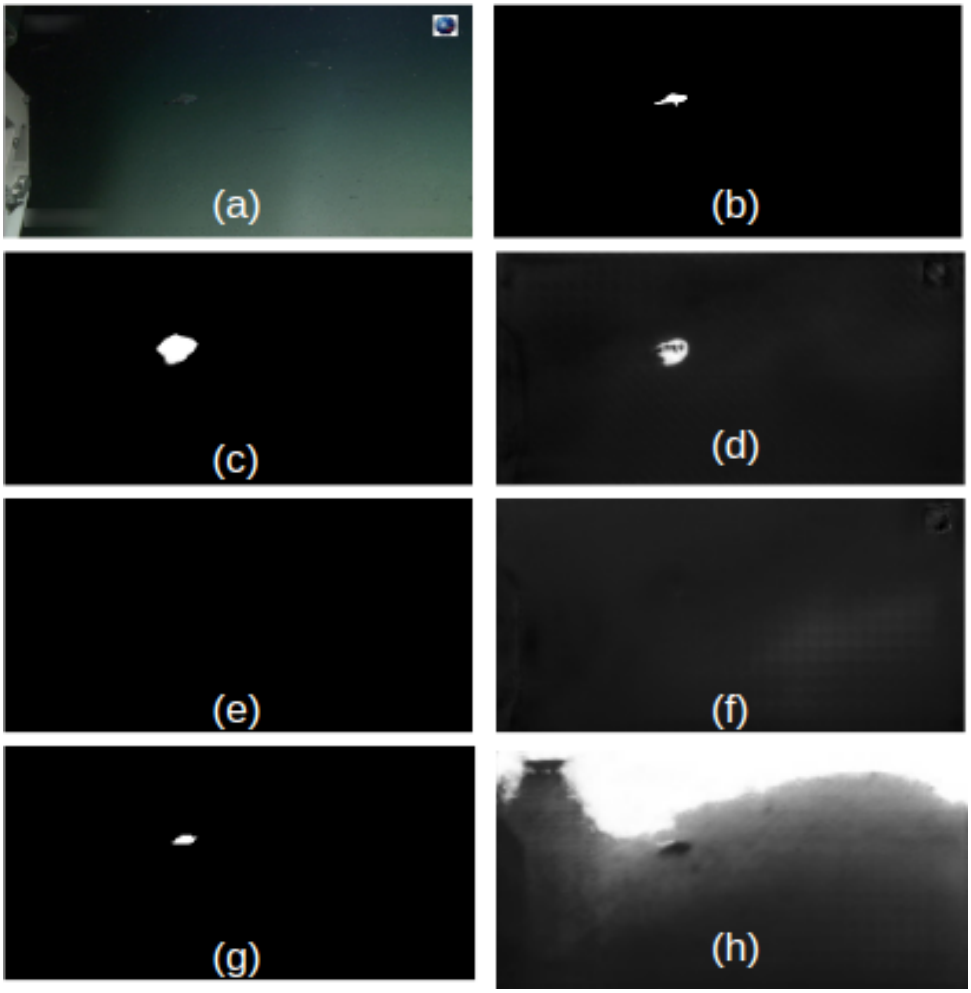

Figure 4.7: Example of the segmentation predictions for the multitask models in the testing set. (a) RGB image; (b) GT; (c) segmentation prediction of the model with 1200 real + 600 simulated images; (d) depth prediction of the model with 1200 real + 600 simulated images; (e) segmentation prediction of the model with 1200 real + 1200 simulated images; (f) depth prediction of the model with 1200 real + 1200 simulated images; (g) segmentation prediction - cycle with task; (h) depth prediction - cycle with task;

4.3 Fish Counting

Using the simple methodology for object counting described in section 3.5, the results obtained are expected taking into consideration previous observations. The validation set has a mean number of fish of 1.95 and 52 images counting only 1 fish with a maximum of 3. The test set has a mean of 1.36 fish with 61 images with just 1 fish and the maximum number of fish in one image is 3.

Table 4.5: Comparison of results for fish counting. For accuracy the higher, the better. For MAE the lower, the better.

Model		**Validation accuracy**	**Test accuracy**	**Validation MAE**	**Test MAE**
354 images	augmentation	0.28	0.15	1.15	1.10
	no augmentation	0.35	0.24	0.75	0.92
1200 images	augmentation	0.76	**0.58**	0.25	**0.50**
	no augmentation	0.68	0.49	0.34	0.71
1200 with 300 simulated	augmentation	0.76	0.56	0.25	0.55
	no augmentation	0.70	0.18	0.31	1.02
1200 with 600 simulated	augmentation	0.85	0.48	0.17	0.62
	no augmentation	0.67	0.20	0.36	0.94
1200 with 1200 simulated	augmentation	0.72	0.33	0.29	0.75
	no augmentation	0.52	0.52	0.67	0.60
NN vector	k=1	0.78	0.20	0.24	0.94
	k=10	0.75	0.44	0.27	0.79
NN mean/std	k=1	0.70	0.56	0.31	0.56
	k=10	0.72	0.55	0.36	0.55
Discriminator	-	0.77	0.27	0.33	0.79
CycleGAN followed by segmentation	augmentation	0.70	0.22	0.56	0.81
	no augmentation	0.76	0.25	0.25	0.86
CycleGan with integrated task	augmentation	-	-	-	-
	no augmentation	0.87	0.25	0.24	0.79
Multitask 1200+600	augmentation	0.72	0.45	0.29	0.77
Multitask 1200+1200	augmentation	0.73	0.45	0.28	0.66
Multitask cycle with task	no augmentation	**0.87**	0.36	**0.17**	0.75

To evaluate the results two metrics were used: accuracy and MAE. The accuracy allows to measure to number of correct prediction, n/N, with n representing the right predictions and N the total number of images. The MAE evaluates the mean differences in the predictions that provide information on the extent of the failures. For the same accuracy value, a high MAE value indicates higher differences in the predictions.

The highest accuracy value for validation is, once again, achieved with the cycle-Gan with integrated task and multitasking, and for the test set the highest value corresponds to the model trained 1200 images and data augmentation, which is compliant with the segmentation results.

The overall values for the validation set are above 0.70 accuracy, which means that less than 30 images are miss classified. Although not high grade results, it is necessary to point that the method used is very simple, and there is a possibility that exploring more robust methods could improve the the efficacy in counting the objects.

4.4 Summary

The results herein presented are the consequence of several methodologies employed and described in chapter 3.

Despite constructing a model capable of performing segmentation, its generalisation is poor even with data augmentation techniques. That being said, augmentation proved to be a positive operation, mainly due to the small size dataset, however when introducing images from the simulated dataset its contributions are less notorious. Using an adapted IoU loss to include bounding boxes annotation for segmentation also proved to be advantageous for the task.

Multiple domain adaptation techniques were attempted to transform simulated data into usable, realistic data but the domain transfer was not without problems. To justify the latter, it is possible that the simulated images are too different from the real ones, or the model is lacking and needs tuning or even a complete restructuring.

Only the model with cycle-GAN and task integrated showed a positive result for domain transfer but even that model has stability issues that question the integrity of the model.

Concerning the multitask experiments the only conclusion is that it does not improve nor deteriorates the predictions.

The object counting methodology is simple to implement but proves that the task is feasible whit accuracy values over 0.7 in almost all cases for the validation set.

Chapter 5

Conclusions and Future Work

In light of the evolving fish farming market, arises the need for overcoming monitoring issues regarding measurement and counting/tracking fish. As the farms grow in size and by the number, a system that could keep control of some important aspects could decrease the overall associated cost and, consequently, provide a variable and inexpensive food source for millions of people. At the same time, off-shore fisheries could also benefit with similar monitoring, providing information to build statistics of each mission, ultimately enhancing the sorting process.

The amount of research in that area is lacking, however, after researching opportune methodologies, some came to light as possible approaches. First, to monitor and measure the fish it is necessary to identify them. Focusing on measurement, the work path was directed towards segmentation of fish. Literature points to machine learning techniques with distinguishable results for that task and, therefore, such techniques build the foundation of this work.

However, there's an intrinsic detail/standard in machine learning: data should be as big as possible in order to propel and generalise learning. Facing the non existence of large quantities of data, some solutions have been proposed to create simulated data and transform it to augment the dataset. This work faces such problem, hence, domain adaptation techniques were chosen to overcome the size weakness.

Several courses of action were put in motion, all explained in chapter 3, including segmentation approaches, domain adaptation and even multitask learning said to improve generalisation. The use of bounding boxes as annotation for segmentation was deemed useful for the work, while multitasking did not change the overall results.

Having said all that, the crucial step of domain adaptation proved to be an impediment for further progress in the research. The benefits expected from reading related literature were not visible herein, tarnishing the work and delaying its primary object: fish measurement.

Nevertheless, the methods presented provide valuable guidance for similar researches and construct a basis for the problem. In the future, the tuning or even a structural change in the cycle-GAN based model expressed in chapter 3 could accomplish a better adaptation. This model showed that, in fact, the domain transfer could be accomplished even if not perfectly. An increase in the validation set could also be necessary to better evaluate results and diminish instabilities.

Finally, efforts to achieve measurement of the fish should be implemented, for instance, resorting to morphological skeletonization as in the work of Garcia et al. (2019), or, as initially planned, using depth map prediction networks to obtain relevant measurements regarding the object.

Appendix A

Literature Review

A.1 Segmentation

Table A.1: Instance segmentation results for COCO 2017 dataset with results for the AP (averaged over IoU thresholds), AP50 (for IoU threshold equal to 0.5) and AP75 (for IoU threshold at 0.75).

COCO 2017 (test)	AP	AP 50	AP 75
He et al. (2017)	37.1	60.0	39.4
Liu et al. (2018)	42.0	65.1	45.7
Du et al. (2020)	42.6	66.5	46.2
Zhang et al. (2020)	43.0	-	-
Liu et al. (2019)	43.3	66.9	46.8

Table A.2: Semantic segmentation results for COCO 2017 dataset with results for mean IoU over all classes.

COCO	mIoU
Yuan et al. (2019)	0.405
Fu et al. (2019)	0.397
Ding et al. (2019)	0.396
Lin et al. (2017)	0.336

A.2 Depth Prediction

Table A.3: Depth results for KITTY dataset. The presented results are accordingly with Eigen split and with caps of 50 meters and 80 meters. K and CS means the model was trained with additional images from CityScapes dataset. It is also indicated if the models were trained with stereo or monocular images.

KITTI	**Superv.**	**Train images**	**abs Rel**	**RMSE linear**	**RMSE log**	$\delta < 1.25$ **%**	$\delta < 1.25^2$ **%**	$\delta < 1.25^3$ **%**	**abs Rel**	**RMSE linear**	**RMSE log**	$\delta < 1.25$ **%**	$\delta < 1.25^2$ **%**	$\delta < 1.25^3$ **%**
			Cap 80 meters						*Cap 50 meters*					
Kuznietsov et al. (2017)	Semi	Stereo	0.113	4.621	0.189	86.2	96.0	98.6	0.108	3.518	0.179	87.5	96.4	98.8
He et al. (2018)	Yes	-	0.086	4.014	-	89.3	97.5	99.4	-	-	-	-	-	-
Xu et al. (2018)	Yes	-	0.122	4.677	-	81.8	95.4	98.5	-	-	-	-	-	-
Fu et al. (2018)	Yes	-	0.072	2.727	0.120	93.2	98.4	99.4	0.071	2.271	0.116	93.6	98.5	99.5
Pilzer et al. (2018)	No	Stereo	0.152	6.016	0.247	78.9	91.8	96.5	0.144	4.660	0.240	79.3	92.3	96.8
Pilzer et al. (2019)	No	Stereo	0.0983	4.656	0.202	88.2	94.8	97.3	-	-	-	-	-	-
Godard et al. (2019)	No	Monocular	0.115	4.863	0.193	87.7	95.9	98.1	-	-	-	-	-	-
Godard et al. (2019)	No	Mono+Stereo	0.106	4.750	0.196	87.4	95.7	97.9	-	-	-	-	-	-

References

Radford Alec, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, January 2015.

Amir Atapour-Abarghouei and Toby P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2800–2810, 2018.

V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2481–2495, December 2017.

Benjamin Bischke, Patrick Helber, Joachim Folz, Damian Borth, and Andreas Dengel. Multi-task learning for segmentation of building footprints with deep neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1480–1484. IEEE, 2019.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Foivos I. Diakogiannis, François Waldner, Peter Caccetta, and Chen Wu. Resunet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:94–114, Apr 2020. ISSN 0924-2716.

Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Semantic correlation promoted shape-variant context for segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8885–8894, 2019.

Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11592–11601, 2020.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *32nd International Conference on Machine Learning, ICML 2015*, pages 1180–1189, 2015.

Rafael Garcia, Ricard Prados, Josep Quintana, Alexander Tempelaar, Nuno Gracias, Shale Rosen, Håvard Vågstøl, and Kristoffer Løvall. Automatic segmentation of fish using deep learning with application to fish size measurement. *ICES Journal of Marine Science*, 2019.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging Into Self-Supervised Monocular Depth Estimation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Advances in neural information processing systems*, pages 2672–2680, June 2014.

Honey Gupta and Kaushik Mitra. Unsupervised single image underwater depth estimation. *2019 IEEE International Conference on Image Processing (ICIP)*, pages 624–628, 2019.

Robert Harb and Patrick Knöbelreiter. Efficient multi-task learning of semantic segmentation and disparity estimation. In *ARW & OAGM Workshop 2019: Austrian Robotics Workshop and OAGM Workshop 2019*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, December 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, pages 4676–4689, 2018.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, pages 3162–3174, 2017.

Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6409–6418, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, March 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Florence Jay, Jean-Pierre Renou, Olivier Voinnet, and Lionel Navarro. Unpaired image-to-image translation using cycle-consistent adversarial networks jun-yan. *Proceedings of the IEEE international conference on computer vision*, pages 183–202, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2215–2223, 2017.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, November 1998.

Stan Z Li. Markov random field models in computer vision. *Lecture Notes in Computer Science*, pages 361–370, 1994.

Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Xiao Lin, Dalila Sánchez-Escobedo, Josep R Casas, and Montse Pardàs. Depth estimation and semantic segmentation from a single rgb image using a hybrid convolutional neural network. *Sensors*, 19(8):1795, 2019.

Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2024–2039, 2015.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection. *arXiv preprint arXiv:1909.03625*, 2019.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

F. Milletari, N. Navab, and S. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, October 2016.

Yipeng Mou, Mingming Gong, Huan Fu, Kayhan Batmanghelich, Kun Zhang, and Dacheng Tao. Learning depth from monocular videos using synthetic data: A temporally-consistent domain adaptation approach. *arXiv preprint arXiv:1907.06882*, 2019.

Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018.

Vladimir Nekrasov, Thanuja Dharmasiri, Andrew Spek, Tom Drummond, Chunhua Shen, and Ian Reid. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7101–7107. IEEE, 2019.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

Andrea Pilzer, Dan Xu, Mihai Puscas, Elisa Ricci, and Nicu Sebe. Unsupervised adversarial depth estimation using cycled generative networks. *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pages 587–595, 2018.

Andrea Pilzer, Stéphane Lathuilière, Nicu Sebe, and Elisa Ricci. Refine and Distill: Exploiting Cycle-Inconsistency and Knowledge Distillation for Unsupervised Monocular Depth Estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9768–9777, 2019.

R. Prados, R. Garcia, N. Gracias, L. Neumann, and Havard Vagstol. Real-time fish detection in trawl nets. *OCEANS 2017-Aberdeen. IEEE*, pages 1–5, October 2017.

Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 3-d depth reconstruction from a single still image. *International journal of computer vision*, pages 53–69, 2008a.

Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, pages 824–840, 2008b.

Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016a.

Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Unsupervised transductive domain adaptation. *arXiv preprint arXiv:1602.03534*, 2016b.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760, 2012.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Tatiana Tommasi and Barbara Caputo. Frustratingly easy nbnn domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 897–904, 2013.

Gaoang Wang, Jenq Neng Hwang, Farron Wallace, and Craig Rose. Multi-scale fish segmentation refinement and missing shape recovery. *IEEE Access*, pages 52836–52845, 2019a.

Gaoang Wang, Jenq-Neng Hwang, Farron Wallace, and Craig Rose. Multi-scale fish segmentation refinement and missing shape recovery. *IEEE Access*, 7:52836–52845, 2019b.

Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan Yuille. Towards unified depth and semantic prediction from a single image. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2015.

Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured Attention Guided Convolutional Neural Fields for Monocular Depth Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3917–3925, 2018.

Hong Yao, Qingling Duan, Daoliang Li, and Jianping Wang. An improved k-means clustering algorithm for fish image segmentation. *Mathematical and Computer Modelling*, 58(3-4):790–798, 2013.

Wei Yao, Zhigang Zeng, Cheng Lian, and Huiming Tang. Pixel-wise regression using u-net and its application on pansharpening. *Neurocomputing*, pages 364–371, October 2018.

Chuang Yu, Xiang Fan, Zhuhua Hu, Xin Xia, Yaochi Zhao, Ruoqing Li, and Yong Bai. Segmentation and measurement scheme for fish morphological features based on Mask R-CNN. *Information Processing in Agriculture*, pages 1–12, 2020.

Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*, 2019.

Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian M. Reid. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.

Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

Ke Zhang, Sun Miao, Tony X Han, Xingfang Yuan, Liru Guo, and Tao Liu. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, May 2015.

Ke Zhang, Sun Miao, Tony X Han, Xingfang Yuan, Liru Guo, and Tao Liu. Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1303–1314, March 2017.

Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 6612–6621, January 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Nan Zou, Zhiyu Xiang, Yiman Chen, Shuya Chen, and Chengyu Qiao. Simultaneous semantic segmentation and depth completion with constraint of boundary. *Sensors*, 20(3):635, 2020.