BIOMASS ESTIMATION OF FISH USING DEEP NETWORKS AND STEREO VISION

by

Mohamad Zaher Abd Ulmoula

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
May 2020

Table of Contents

List of Tables

List of Figures

Abstract

Mass of an object is essential information for many an industrial application. The aquaculture industry estimates the fish's weight by scaling a sample of fish out of growing tanks. This process harms fish and reveals an inaccurate result. The research found there is a high correlation between fish weight and its size in the image. This study aims to use a convolution neural network (CNN) for estimating fish weight.

Firstly, according to our research, VGG19 or similar models were not tested to solve this problem before. Therefore, with known distance we tested CNN models VGG19 then compare its result with semantic segmentation models such as FC-Densenet, where another study applied a semantic segmentation technique on a smaller problem. To do this experiment, we used a fish dataset included 1275 images of harvest Salamon fish and their mass. The VGG19-R archived the lowest mean absolute percent error (MAPE), MAPE = 2.4%, and the FC-Densenet-R revealed MAPE = 6.49%.

To stimulate fish in a tank, we took a picture of a Lego block with a stereo vision camera in different positions to the camera. Then, we used the stereo data [right, left, depth map] as input to the VGG19-R model to estimate the area of the object. The model achieves MAPE= 2.37% for the testing dataset. The result shows that the stereo vision camera could help to measure objects at different depths like fish inside the tank, where the depth map information works as a re-scaling factor to object area in the other inputs.

Acknowledgements

I wish to express my deepest gratitude to my supervisor, Professor Thomas Trappenberg, who guided and encouraged me to be professional. Without his patience, motivation, and immense knowledge, this study would not have realized. Besides my adviser, I would thank all the computer science faculty members, especially Dr. Fernando V. Paulovich and Dr. Malcolm Heywood, for being a reader.

I wish to acknowledge my parents and my uncle 'Ali's' family for their continuing support throughout the period of this research study. They kept me going on, and this work would not have been possible without their support.

Chapter 1

Introduction

The research study is about the deployment of a deep learning model capable of estimating the mass of an object from its image by using stereo vision technique. The technique explored in this research study does not require fixing the distance between the camera and the object. As a research contribution ,I have studied and develop a deep learning model that reads stereo data [right and left images, and depth map] to estimate the area of the object in those images. Furthermore, the depth information is embedded in the input image.

The weight of an object is important information for many applications. In this study, we focus on the aquaculture industry. In recent years, the use of fish and its subsidiary products has been risen. According to the research, fish products cover 16% of the human diet around the world [37]. As per the official OECD statistics, Canada is the fifteenth country in producing aquaculture products [28]. The aquaculture industry's Gross Value Added (GVA) increased by over 50% between 2007 and 2017 from $265 million to $572 million in Canada [1]. The aquaculture industry added over 14000 full-time jobs opportunity to the market in 2009 for the Canadian market [31]. The Canadian aquaculture farms produce a variety of fin-fish and shellfish products, and the main fin-fish products are salmon, trout and steelhead [1].

Fish breeding in a farm setting involves monitoring. The monitoring process is a routine task to protect fish from diseases and stress incidents [37]. In addition, aquaculture helps to improve the quality of the products. There are two main phases of monitoring: pre-harvest and post-harvest [37].

The pre-harvest monitoring involves studying fish behaviour as a group and as individuals inside the fish tank. This monitoring is essential for farmers because the fish are sensitive to the environment and detecting small changes in fish behaviour could prevent a significant loss [37]. Moreover, pre-harvest is vital to sales departments. The information we get from monitoring helps marketing department to choose for the selling channels earlier, depending on the fish's psychical features [37].

Post-harvest monitoring mainly assists in sorting the fish to different markets [37]. That depends on physical and intrinsic chemical attributes. Some of the physical characteristics are shape, size, texture and colour [37]. The intrinsic chemicals are fat and protein content, and blood spot [37]. They play a significant role in customers' purchase decisions and show the quality of the product [37].

In the aquatic business, a farmer looks for several factors to determine the fish behaviour as a group; For example eating activity, detecting the excess feeding, which adds pollution to the water, and speed. Also, some other factors are under biological studies, such as the leadership or merging groups, and mixing species [37]. There are additional features, such as sex, length, width, skin colour and maturity, which are considered crucial for enhancing the fish quality besides the weight of the fish. While breeding, such features equip the farmers with the information to segregate the fish by assigning them to different tanks. In addition, it gives a general view of the health of the fish [19, 37].

The monitoring process is not an easy task. It is persistent, costly, time-consuming and also adds stress to the fish. The farmer has to catch some fish to measure their features such as length, width and mass. Generally, the procedure for fish catching involves the use of the net. This whole procedure of fish catching puts a strain on the well being of the specie albeit handled properly, which could either cause death to the fish or reduces the quality of their meat. Further, the monitoring procedure is only performed on a small group of fish, so it does not give a full picture of the fish in the container [37].

As a result, an automated solution is proposed, in this research study, to improve the monitoring process and help the aquaculture industry [19, 37]. The proposed automated application must have the following three attributes. Firstly, it needs to assist in reducing the cost of production by reducing human involvement. Secondly, it should not add any stress to the fish. Finally, it should be cheap and reliable. The best technology for enabling many monitoring tasks is computer vision [37].

Computer vision (CV) is one of the exciting subjects in machine learning. It is used to solve the real-world problems by analyzing image data to obtaining information. As an example of the application of CV is the use of AutoPilot in self-driving cars which was first brought out by Tesla [33]. Computer vision uses optical sensors (cameras), which have become cheaper and more powerful over time. It does not harm the fish. Those two reasons make computer vision the right technology for solving the monitoring issues [37].

In the last decade, neural networks (NN) have been developing. We see applications using NN everywhere, starting from detecting objects in images to defeating a human team in video games [7, 29, 33]. The reason for success is that NNs excel in fitting complex data such as images or a text dataset. For image datasets, there is a particular type of layer neural network called a convolutional layer [7]. The convolutional layer uses a convolutional filter to learn from the image abstract patterns. By adding a series of layers, the model learns to detect a sophisticated pattern like eyes or a full human face. In a modern application, of NN has many layers, and it is called a deep learning model [7]. Chapter 2 of the thesis reviews some details about machine learning, computer vision and deep learning.

The aquaculture farms have to monitor the fish and know the mass of fish inside the container tank. Our solution to automate the monitoring process is to use computer vision with a deep learning model. The two-dimensions images has all the information to detect if a specific object is in the image or not. But to estimate the object size, we need to know the cameras parameters, and the distance between the camera and the object. Because this distance affects the estimation of the physical size of the object in the photo. The closer the object to the camera, the bigger the image. Traditionally, we could use a known objects as reference measurement and it should by the same image. Then by comparing another objects to it. For example, we could put an object such as a ruler on the other side of the tank, but swimming fish could cover it. Besides, water is not clear enough to show the ruler. Therefore, we need a solution that can give the depth of objects in the image without any extra information. Further, it should be cheap. The best fit for this problem is stereo vision.

Stereo vision is a setup of two cameras that capture the same view from a different point on the viewing plane [11, 32]. There is no limitation for the distance between the two cameras but it affects the min and max depth that could be estimated from the images [32]. The stereo vision captures two images, a right view and left view.

From those images, we could calculate the depth map for each pixel from the disparity map, which is the difference of matched pixels [11, 32].

In summary, we worked on two tasks in this study. First task, we developed a deep learning models to estimate fish weight from known distance. That depending on the correlation of the fish's area in an image to its mass [37]. Therefore, in second task is estimate an Lego block area with different depth from the camera using stereo vision data. Where the depth information is re-scaling factor for the area of the block.

In this study, we are not claiming that we could estimate fish weight inside the tank yet. That because many other problems need to solve first, such as clearness of the images underwater and poor light conditions [37]. However, we are confident that it could do a good job in case we had a chance to test it on stereo data of fish underwater.

1.0.1 Related work

The fish's weight is essential to fish farmers for improving the product's quality. Therefore, a lot of research are conducted to estimate the biomass of fish from images [19, 37]. That is because the optical sensors are the best technology for this task. First, it is easier to take pictures of the fish without applying any pressure on it. Second, in the last three decades, the camera's sensors are getting cheaper and more reliable [37].

Estimating the mass from the length is a common approach. De Verdal et al. noted in their sea bass fish study that there is a correlation $r^2 = 0.98$ between the fish features in the image (length, height, area, perimeter and volume) and the weight of the fish [37]. Length-weight correlation used in some studies like Dios et al. Depending on this correlation some studies like Dios et al. built a stereo vision system to select the fish's length manually to evaluate the mass of the fish depends on this equation 1.1, which is a relation to finding the weight from the length of the fish.

$$Powercurve : M = aL^b \tag{1.1}$$

Where a and b are the two coefficients and they depend on the water and fish feeding [23]. Dios et al. get an error rate close to 5% for salamon fish [23]. Similarly, Sancheze-Torres et al. estimate the fish length from contour area, then used the estimated length for estimating the Mass, by using 3rd degrees polynomials for both of the functions as in the equation 1.2 [19].

$$L = f(C), M = g(L) \tag{1.2}$$

where L is the length, C is area of the fish contour, M is estimated fish mass, and $f()$ and $g()$ are polynomial functions [19]. Sancheze-Torres et al. report length estimation with mean absolute percent error (MAPE) = 3.6% and MAPE = 11.2% for fish mass estimation [19].

In another study, they applied three different mathematical models 1.1, 1.3 and 1.4 on datasets contains 120 measurements of Jade Perch fish.

$$Polunomial : M = a + bs + cL + dH \tag{1.3}$$

$$Liner : M = a + bS \tag{1.4}$$

Where S is the fish's area in the image. H is the fish height. After testing the models on 64 images. they found the equations 1.3 and 1.4 show the lowest error, where MAPE = 5%. But equation 1.1 error is MAPE = 10%.

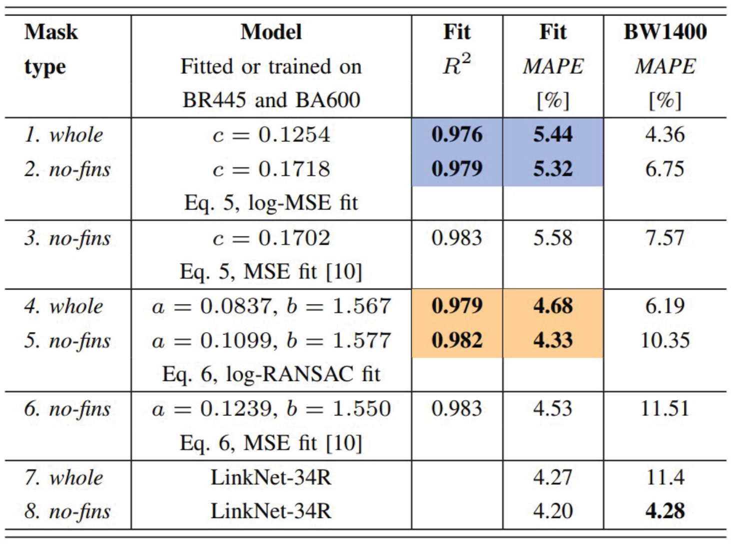
Konovalov et al. developed another mathematical model using S with a, and c, which are coefficient parameters for food and species. MAPE value is 5% for the Asian sea bass fish dataset [37, 19].

$$M = cS^{3/2}, c = 0.17 \tag{1.5}$$

$$M = aS^b, a = 0.124, b = 1.55 \tag{1.6}$$

Further, Konovalov et al. work on estimating the weight of the fish from images by using a neural network and they apply the semantic segmentation model (LinkNet-34). It can segment the fish in the image, then they use the output mask for estimating the fish weight. They train the model using three datasets of Barra fish with a total 2445 images, and all the images scaled one mm-per-pixel. For estimating the fish weight, they used the mathematical model equations 1.5 and 1.6. Konovalov et al. used deep learning regression to use the output of the LinkNet-34 as input for the unknown regression part. The results are presented in table [19].

Nonetheless, those models were built for estimating harvested fish where the fish is not moving and the fish hold to get the right image. Plus, all of the image issues,



Mask type	**Model** Fitted or trained on BR445 and BA600	**Fit** R^2	**Fit** *MAPE* [%]	**BW1400** *MAPE* [%]
1. whole	$c = 0.1254$	**0.976**	**5.44**	4.36
2. no-fins	$c = 0.1718$	**0.979**	**5.32**	6.75
	Eq. 5, log-MSE fit			
3. no-fins	$c = 0.1702$	0.983	5.58	7.57
	Eq. 5, MSE fit [10]			
4. whole	$a = 0.0837$, $b = 1.567$	**0.979**	**4.68**	6.19
5. no-fins	$a = 0.1099$, $b = 1.577$	**0.982**	**4.33**	10.35
	Eq. 6, log-RANSAC fit			
6. no-fins	$a = 0.1239$, $b = 1.550$	0.983	4.53	11.51
	Eq. 6, MSE fit [10]			
7. whole	LinkNet-34R		4.27	11.4
8. no-fins	LinkNet-34R		4.20	**4.28**

Figure 1.1: Mass estimation models of Konovalov et al. from [19]

such as light or clearness, are taken care of by the researcher while taking the photos. In a real situation, the fish are moving in the tank and images are not clear because of the pollution in the water. Therefore, those models will not have the same result [37]. They would be useless for pre-harvest monitoring applications.

In this study, we built an end-to-end convolution network for estimating the biomass of the fish. With known distance, We tested VGG19 and FC-Densenet for training the model. We used the fish dataset for training those models, and the VGG19 models got MAPE 2.4% and FC-Densenet got 6.49% at the test dataset.

In further experiment, we worked to build a deep model that includes depth preparation. It read data from a stereo vision camera to estimate the area of a object at different depth. This was only done on a n example setting with Lego blocks as a proof of concept since we did not have such data with fish. This model achieved MAPE 2.37% for the testing dataset. This experiment is detailed in the chapter three.

Chapter 2

Background

2.1 Computer vision

Computer vision (CV) is a sub-field of artificial intelligence, and more recently, a sub-field of machine learning because of applications of machine learning techniques. Computer vision is a studied field focusing on media data to solve real-world problems. The goal of it is extracting useful information from images or videos, then use that information to solve the problem [32].

There are two main challenges to make computer vision compatible with human vision. Firstly, we do not have a full understanding of the human vision. Good progress made to understand the human eye and the way information transferred to the brain. However, there is much work that needs to do to know how the brain processes this information [45, 4]. Secondly, the world is complex, and many variables affect any vision system like light conditions or clearness. The accurate vision system should be able to see and extract information in any condition and from an infinite number of scenes [45, 4].

Regarding all the challenges, computer vision makes a good achieving in many tasks like [45]:

- Optical character recognition (OCR).
- Machine inspection
- Retail (e.g. automated checkouts)
- 3D model building (photogrammetry)
- Medical imaging
- Automotive safety
- Match move (e.g. merging CGI with live actors in movies)
- Motion capture (mocap)
- Surveillance

- Fingerprint recognition and biometrics

Recently, using deep learning technical makes a massive jump in solving many computer vision tasks such as:

- Object Classification.
- Object Identification.
- Object Verification.
- Object Detection
- Object Landmark Detection.
- Object Segmentation.
- Object Recognition.

This progress leads to making autonomous vehicles real, and optimists predict the autonomous vehicle will be reliable to replace human drivers by 2030 [21]. Medical image analysis is another area that got improving in past years which to detect cancers in early stages [4]. Moreover, recently the other CV technologies like virtual reality (VR) and argument reality (AR) added to be part of our daily life. Now, most of the phones work as VR or AR set for teaching or entertainment applications. For example, PokemonGo, Anatomy 4D [2].

2.1.1 Stereo vision

The stereo vision is a system content two cameras or more for perceiving the depth information. For example, a human's vision system is a complicated stereo vision system. Where the brain is handling all the processes [45].

The stereo vision is used applications like Robots, automobiles, virtual reality, argument reality, and security cameras. Also, many other applications needs the depth information. In this section, we are explaining how the stereo vision is used to measure the depth from two images [25, 43].

Usually, the cameras of the stereo vision are separated horizontally to get two different 2D images for the same view. The distance between the lenses of the cameras is called the baseline. The cameras of the system should be identical, so they have the same focal length, which is the length between the lens and reflected the image on the sensor in the digital camera [15, 43].

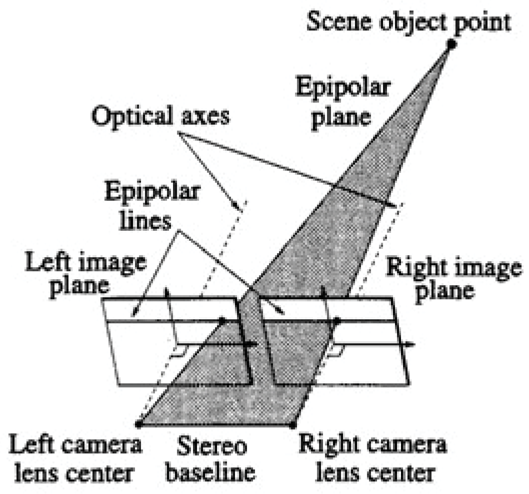


Figure 2.1: Stereo Vision components from [15]

As shown in figure 2.1, the stereo vision has several components: the centers of lenses, baseline, which is the distance between the lenses, images' planes, which are the reflected image, and the focal length is the distance between the lens and the reflected images' centers.

2.1.2 Depth

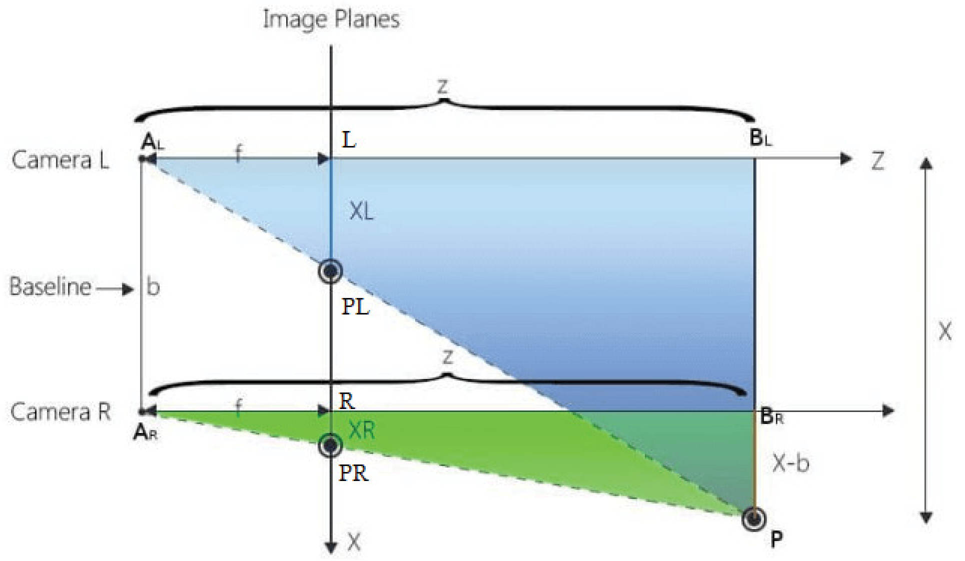


Figure 2.2: Geometry Projection of stereo vision from [24]

In the diagram 2.2, there are two right angles triangles. Starting from blue triangle, heads of this triangle are A_L optical centers of the left camera, P point in the world, and B_L which the project of P on the Z from center of the left camera lens. Let assume that X is the distance between the optical centers and the P in the real world. The adjacent of A_L is the distance of the 3D point P intersecting the lines drawn from the optical centers. There is a smaller blue triangle, the heads of this one are A_L, L is the center of the left image planes, and P_L is the P project on the image plane. By applying triangles similarity between those triangles, we can find:

$$\frac{X}{Z} = \frac{X_L}{f} \tag{2.1}$$

Similarly, the green triangle includes two triangles. The heads A_R which is the optical centers of the right camera, P point in the world, and B_L which the project of P on the Z from the center of the right camera lens. However, for the green triangle, the distance between the optical centers and the P in the real world is equal to $X - b$, where the b is the baseline of the stereo vision. Like the blue triangle, calculating the similarity between the two green triangles is:

$$\frac{X - b}{Z} = \frac{X_R}{f} \tag{2.2}$$

So, combining the equation 2.2 and 2.1 is equals:

$$Z = \frac{bf}{X_L - X_R} \tag{2.3}$$

$$d = X_L - X_R \tag{2.4}$$

$$Z = \frac{bf}{d} \tag{2.5}$$

where:

1. Z is the depth
2. b is the length between the lens center.
3. F is the focal length (Which is equal in both cameras).
4. d is the disparity (the disparity).

So, by using the equation 2.5, we could generate the depth map for each pixel in the disparity map [15].

2.1.3 Camera calibration

In the pinhole camera model, taking pictures by a camera is the process of converting a 3D image to a 2D image. The mathematical expression is this:

$$Wa = AP \tag{2.6}$$

Where W is a scale factor, a is image point matrix $\begin{bmatrix} X & Y & 1 \end{bmatrix}$, A describes the world points $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}$, and P are the camera parameters [51].

$$P = \begin{bmatrix} R \\ T \end{bmatrix} K \tag{2.7}$$

$$K = \begin{bmatrix} f_x & 0 & 0 & 0 \\ s & f_y & 0 & 0 \\ c_x & c_y & 1 & 0 \end{bmatrix} \tag{2.8}$$

As in the figure 2.3, the camera has to type of parameters the extrinsic parameters and Intrinsic parameters. The extrinsic parameters consist of translation T, and rotation R. They describe the world coordinate to camera coordinate. In other words, the extrinsic parameters define the relation between point P coordinate in the world's coordinate P_w and camera P_c coordinate.

The intrinsic parameters K, where (c_x,c_y) are the coordinate of optical center, (f_x,f_y) is the scale factor, and s is the Skew coefficient of the image axes [51]. The intrinsic parameters represent the geometric and optical characterize. And they aid to convert images from 3D coordinate to 2D coordinate [51].

The Extrinsic Camera Matrix $\begin{bmatrix} R & Y \end{bmatrix}$ is representing the camera position in the world. This matrix is (4×4) array to described the location of the camera and what direction the camera is pointing. So, we can write the equation 2.6 in matrix form as:

$$W \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \times \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.9}$$

Multiplying 3×3 matrix with 3×4, the result will be 3×4. Then it will be multiplied by 4×1. The result will be a 3×1 matrix [12, 40].

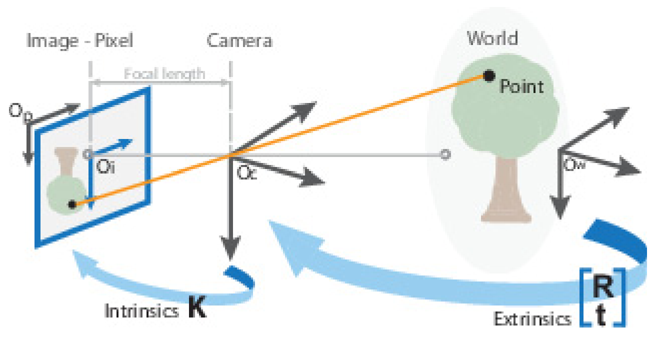


Figure 2.3: Calibration Coordinate from [24]

The calibration is a method to retrieve the camera parameters from 2D images of knowing 3D world parameters. Zhang, in his paper [51] develops a method to estimate the camera parameters by taking several pictures for a known pattern like a chessboard.

Zhang's method is:

1. Capture a few images to know pattern image, with different orientation.
2. Recognize the feature points in the images.
3. Estimate the extrinsic parameters and the intrinsic parameters.
4. Improve all parameters, including lens distortion parameters, by minimizing the difference between known image feature point values to estimated ones.

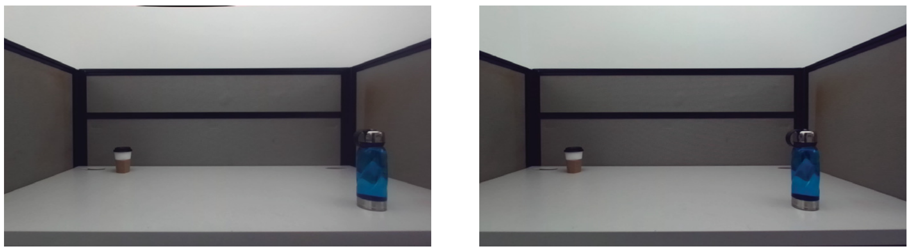
2.1.4 Disparity map

The disparity map is the difference between pairs of images for each pixel, and even it is represented as a gray-scale image. The range of the values is in range [0-255]. 255 is the closest possible to the camera because the closet object to the camera has a higher disparity [15]. The 0 is mean the pixel is maximum depth from the camera or un-calculated disparity for some algorithms.

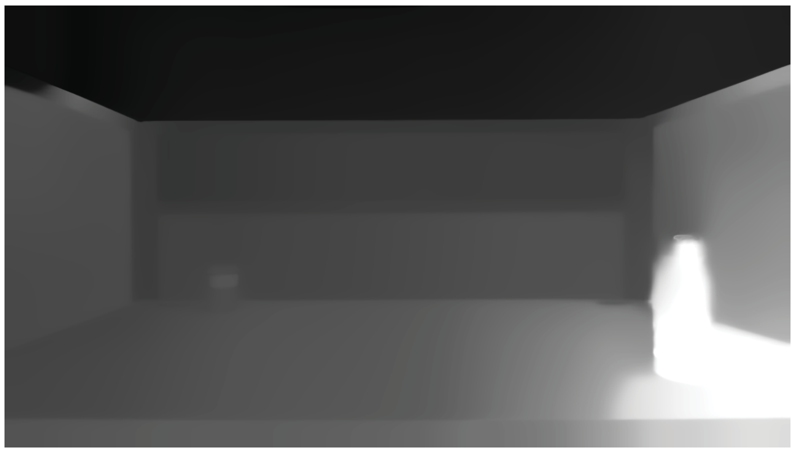
Most of the stereo block match algorithms have the same four steps to generate the disparity map [38].

1) **Cost Initialization:** This creates the 3D cost volume -The matching cost of each pixel at different disparity levels- so the output is [X, Y, D].

2) **Cost Aggregation:** which aggregates the spatial cost for each pixel.



(a) (b)



(c)

Figure 2.4: (a) is the left image, (b) is the right, and (c) is disparity map for both images (a) and (b).

3) **Disparity Optimization:** find the best disparity for each pixel.

4) **Disparity Refinement:** this step is to reprocess the disparity map to smooth the output disparity map.

So, disparity map algorithms are consuming a lot of resources, because of these steps calculated for each input image, which makes it hard to run an algorithm on any machine without concessions on the accuracy of the output.

2.2 Machine learning

Machine learning is part of artificial intelligence. For understanding the machine learning concept and why it is important. We need to compare it with classical programming. To make a traditional program, you need to have two components, the data and the rules, which are applied to the data to get the answers. That is working if you know the rule (process).

In contrast, machine learning applications, we do not have rules. This because the rules are complicated, or we do not know the right rules to achieve the answers like finding relations between data has hundreds of features. Therefore, machine learning algorithms use the data, and the answers to find the rules, then used the rules to get answers for the new data [7].The machine learning algorithms have many branches: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

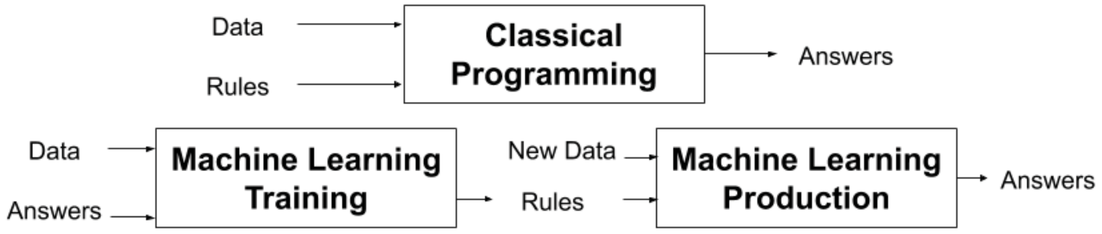


Figure 2.5: Classical programming vs machine learning from [7] with some changing .

In this study, we consider the supervised learning, unsupervised learning, and semi-supervised learning that because they used for build depth estimation models. The supervised learning is the most used case; it learns to map the input data (called features) to generate the target from knowing the target (also called annotation). The annotation even a has been made by a human. For example, to build a model predicts cat images, we will need to many photos half of them content cat and another half not content cat, then annotated those images to 1 where the picture has a cat or 0 if not have a cat in the image. Therefore, the model will learn to return 1 when it found a cat in the input image and 0 when it did not.

Alternatively, for some other type of problem, the developer does not annotate the data, the labels generated with input, like stock market data, where the current price is one of the input features, and the next price is the target. The supervised learning approach used mostly for all machine learning problems. Such as image classification, language translation, and object detection [7]. Support Vector Machine (SVM), Random Forest, Naive Bayes, and many other algorithms are examples for supervised learning algorithms.

In contrast to supervised learning, the unsupervised learning model learns to transform input data into another exciting format. That is happening without using any labelled data. Clustering, dimensionality reduction and association rule learning are examples that need unsupervised learning to solve [7], K-means is one of the unsupervised algorithms used for clustering problems.

Lastly, semi-supervised learning is a concept between supervised learning and unsupervised learning. Where semi-supervised learning does not use labels to train the model. Still, it used the input images as labels to train the model.

2.2.1 Deep learning

Deep learning is a series of multi neural network stock as a layer. This means that the deep learning model is learning hierarchical representation, and the number of layers is the deep of the model. We will first explain the concept of the neural network, then focus on deep learning.

Neural network is one of the machine learning algorithms. It is like all ML algorithm learning to find the most useful representing of the input data to get closer to the expected output. But what makes NN different from the other ML's algorithms is the flexibility of how it represents the data. The operation could be a linear, translation, or nonlinear.

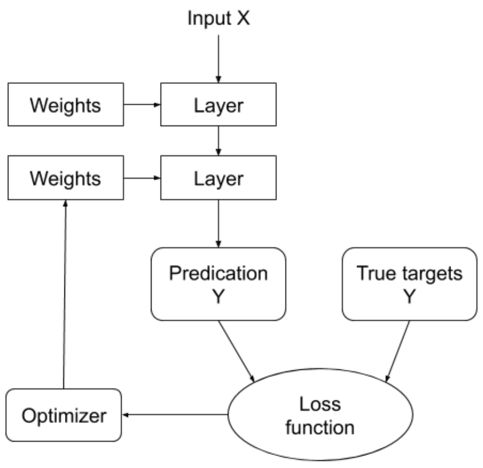


Figure 2.6: Sample deep learning model components [7] with some changeing.

The neural network contents N number of neural, where each neural takes one input X and applies the f function (called activation function). For example, the equation $f_i = X_i W_i + B_i$ used the linear activation function for the neural network. Where W is a weight for i, and B is a bias. The output of the layer could be f_i, which used as input for another neural layer. Yet it could be $Y = \sum f_i$ for regression, or applied other activation function like a binary step.

Training deep learning has two phases. Firstly, the forward-propagation phase is applying the activation function for each layer until calculating the predication Y. Secondly, the back-propagation phase, where the optimizer works to adjust the weight of NN layers, slightly depends on the learning rate. That operation is for reducing the loss cost for after each iteration. The training process is repeating two methods forward-propagation and back-propagation for many iterations until getting the lowest loss value using all the training input data. Then the NN evaluated using the forward-propagation phase on all the test input data.

The deep learning has been using in many domains of data images, text, or audio for different types of problems like classification, prediction or regression. Moreover, it trained to generate data or convert data, such as, put colour to the gray-scale images, create music or text. Besides that, the deep learning pushes reinforcement learning to a new level [7].

Now, a deep reinforcement learning model trained to drive cars without human interaction. Also, the OpenAI team trained a model to play a video game called Dota

2 and got to achieve a great result even the got to make five models contribute to winning the game [7, 29].

2.2.2 Semantic segmentation

Semantic segmentation is one of technique used for image classification problems by labelling each pixel in the image to one of the classes. The output from the semantic segmentation model is an image that has input dimension, but each pixel the output image has a colour corresponding to the object of that pixel. For example, in figure 2.7, where all pixels classified as people coloured in blue. But the cars in green. That feature benefits to detect and allocate the object in a picture. Further, it helps to know the size of the object in the image.

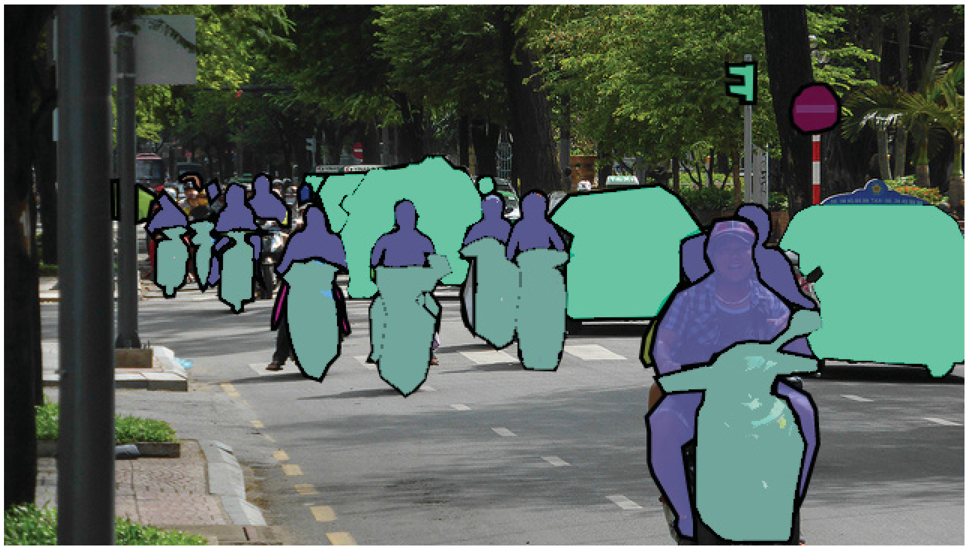


Figure 2.7: Semantic Segmentation annotation from [20] with some changing

Back to the example in figure 2.7, which it is from Coco dataset for the semantic segmentation annotation (ground truth): The cars, motors, people, and traffic lights highlighted in different colours [20]. However, the other part of the image will be ignored or considered as a background class. So, the model trained to identify 4 categories in the picture.

This technique of classification is useful, and it has been used in many applications like medical images classification, self-driving cars, and satellite image analysis. It shows excellent results, especially regarding the medical images where the detected object represents a small space, which makes other technique useless [8, 47, 17].

Convolutional networks

The convolution network is a deep learning model that has several convolution layers. The convolution layer has some matrices of nodes. By applying a convolution function on the input data, the nodes' weight learned to obtain different information from the input data. Usually, the convolution network has content convolution layers followed by a non-linear activation function, pooling layers, and batch normalization.

The NN is a hierarchical data representation, and that is very obvious in the convolution network. The top convolution layer learns the low level of features like edges, colours. Yet, by going deeper into the model, the feature starts to be more complicated so that the layer activates. For example, if the model trained to detect dogs' faces. The earlier layer of that model activated by edges. The next layers activated by a dog's ears or eyes. Later, the layer activated by the all face area in the input image [50]. That means earlier convolution layer content information about the small region of the image. The deeper convolution layer contains information about the larger area of the image [50].

Usually, by adding a more convolution layer, the number of channels (filters) will be increasing, and each learned one of the features. For reducing the number of parameters, the pooling layer added to the network where the pooling layer works as down-sampling the filter size [7].

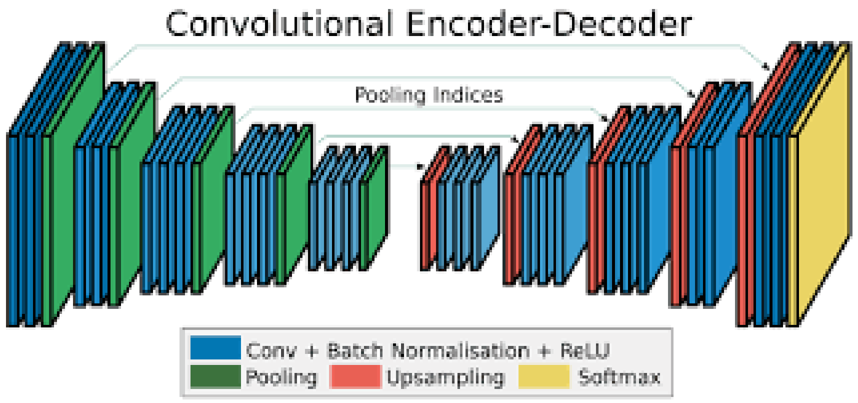

Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax

Figure 2.8: Encoder-Decoder Architecture from [17]

The difference between the standard classification model and semantic segmentation model is in the traditional classification models; after the convolution network, the model end with spatial tensor. Therefore, we need to add a fully connected layer to map the spatial information and reduce the size of it to do predication. On the other hand, for the semantic segmentation model, we need to retrain the low-resolution spatial tensor, which content high-level information to generate the segment. For that, we add a new convolutional layer and an up-sampling layer to increase the size of the channel, those steps called Encoding and Decoding 2.8.

Generally, the semantic segmentation models are content two parts Encoder and Decoder. The encoder works to encode the input feature from the images. Then the decoder converts the learned features to regenerate the image has the same size of the input each pixel has colour depends on the class of that pixel from the input image.

The encoder architecture could be any convolution networks; For example, VGG16/19, ResNet, or MobileNet because of the idea of the encoder is extracting information from image and compose it in smaller tensor [13, 41, 44]. Then the decoder has several convolution layers and transposed convolutions layers (or deconvolution layer) to rebuild the output.

One of the earlier semantic segmentation is the Fully Convolution Networks For Semantic Segmentation. In that paper, they test three different known convolution layer architectures [AlexNet, VGG16, and GoogleNet] as the encoder. Still, for the decoder, they up-sample the output of the last Conv layer, and they found the shallower layer has the information about the location of the class. Therefore, they merge the output of the previous layer with deeper ones. That increases the accuracy of the segmentation, as in Figure 2.9.

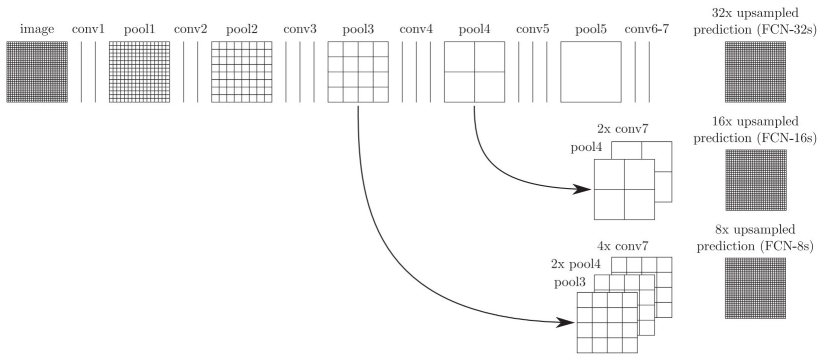


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 2.9: Fusing for FCN from [39]

Later in the SegNet, they build an encoder similar to VGG16, followed by the corresponding decoder [17, 41]. They used the output of the Max-pooling layers

(Pooling indices) merged with the previous convolution layer. U-net and DeconvNet have similar architecture but with some differences [17, 39].

One of the resent state-of-art semantic segmentation models is FC-Densenet. This model achieves a great result with a small number of parameters compared to other architectures. FC-Densenet obtains 66.9 Mean IOU (MIOU) on the CamVid dataset with 9.6 million parameters, but the closest model makes 66.1 MIOU with 140.8 million parameters for the same dataset [3, 48].

The main reason for that result is the use of the Densenet blocks, which gives three benefits: 1) feature reuse; all the layers in each block can reuse the previous feature, 2) parameter efficiency; using a Dense block reduces the number of the parameter but without reducing the effectiveness of the model, 3) implicit deep supervision; short paths between all dense blocks perform deep supervision [48].

FC-Densenet model contents dense blocks in encoder and decoder. There are shown sampling layers between the blocks in the encoder, then the up sampling layer in the decoder. Before each down sampling layer, there is a residual connection from the previous output. Also, there is a skip connection between each residual link to equivalent tensor in the decoder [48]. The figure 2.10 shows the architecture of FC-Densenet.

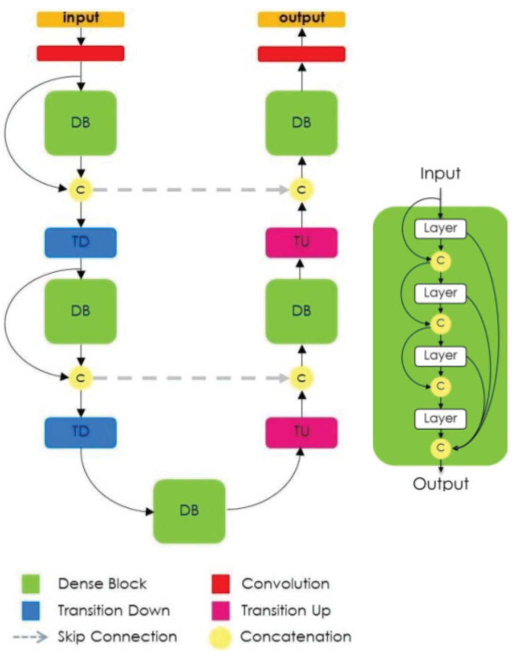


Figure 2.10: FC-Densenet from [48]

Transfer learning

At the initial state, the convolution layers' weights have random values. Weights do not have any useful information at that time, but by training weights updated to be meaningful. The training process usually needs a lot of time and resources. Therefore, transfer learning is a proper way of saving time and data resources.

In the last section, we discussed convolution networks and how the firsts layers learn low-level features, but the deeper layer absorbs the higher-level features. Those low-level features are edges, colour, and some sample shapes. Although they are essential because they appear in any images, using weighted layers trained on any image dataset is better from the start of training the model from randoms weights.

There are two main benefits of using transfer learning. Firstly it is saving training time because some of the layers do not need to train; they have the information they need. Secondly, for some applications, it is hard to get enough images to train the deep model from scratch. Transfer learning reduces the data amount, which required for fitting the model and get better accuracy.

Technical speaking, we train the model on one of the big data set like ImageNet, or COCO. After that, we retrain the model excepting the earlier layers' weights, which contains the low-level of information, then retrain the model by using our dataset [35].

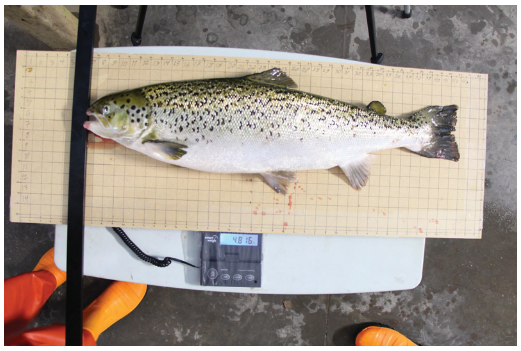
Image augmentation

The deep learning model requires many data sampling to train. In addition, data needs to be labelled, and labelling images cost time and human resources besides getting enough data. One of the solutions is using image augmentation. The image augmentation is applying image processing to generate new images from the existing ones.

Some of the image functions are a horizontal and vertical Shift, horizontal and vertical flip, random rotation, random brightness augmentation, and random Zoom augmentation. All those functions are great for generating new images from the same dataset. That will supports model for better generalizing over the data and keep it away from over-fitting on the training dataset. Regarding the semantic segmentation model, the augmentation process should be applied to both input image and label by the same parameters (except brightness), or that will have a harmful effect over the final result.

Nevertheless, image augmentation should be considered as hyper-parameters because as it helps to avoid over-fitting. Yet, it could cause under-fitting by adding many new images. Where the number of the model's parameters is not enough to fit the generated data, therefore, the augmentation should be implemented as a hyper-parameter to tuned to get the best fit for the model [7].

Figure 2.11 shows an examples of image augmentation functions [30].

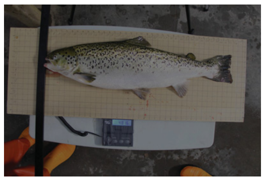

(a) Original image

(b) Horizontal Flip

(c) Vertical Flip

(d) Shifted Image

(e) Increase Brightness


(f) Decrease Brightness

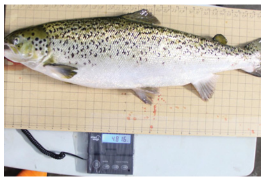

(g) Zoom-in

Figure 2.11: Examples for augmentation process, d shifted image by 100 PX on X axis and 250 PX on Y axis

2.2.3 Estimated depth by deep leaning

Recently, many researchers started work on depth estimation by using deep learning due to the importance of depth information for many active fields, such as driverless

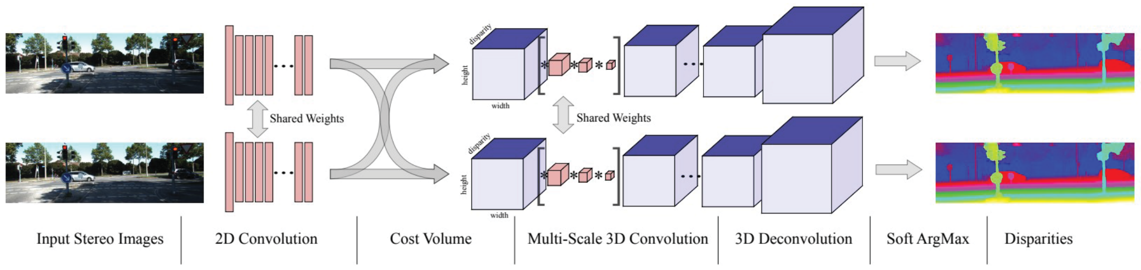


Figure 2.12: GC-Net model [18]

cars, drones, and any robots. Estimating depth using algorithms, as we explained previously, is consuming a lot of resources, which is not available in robots like drones. Also, some deep learning models achieve to estimate depth from one input image like MonoDepth, which is excellent for use by devices that have one camera like phones, or medical equipment [10].

In general, there are several types of models developed in the last year 2019, over 45 new results submitted on the KITTI 2015 dataset to new models estimate depth by using a different approach [9, 27]. KITTI 2015 dataset is benchmark data for stereo vision challenge; the data took by stereo vision setup and a laser scanner for generating the depth map. The dataset contents 200 images for training and 200 for testing [27].The main two categories are binocular vision (stereo vision), and monocular vision.

GC-Net is one of the earlier models developed by a team from Skydio [18]. The GC-Net is a stereo vision model. It takes a right and left image as input data and output is the disparity map. It contains three parts of feature extraction, cost volume, learning context. The feature extraction is content multi convolutional layers, and they are shared weight between the two input images (right, and left).

The cost volume is the core of the model, where it works to generate 4D tensor $[width \times height \times (maxdisparity + 1) \times featuresize]$. It does that by concatenating feature X from the left image with corresponding feature X from the right image across each disparity level. The cost volume learns to match the features with probate disparity value [18].

The next part in the model is the learning context, which contents the encoder and decoder building 3D convolution layer to learn the feature representation from the cost volume [Width, Height, Disparity]. The GC-Net model trained by the scene-flow

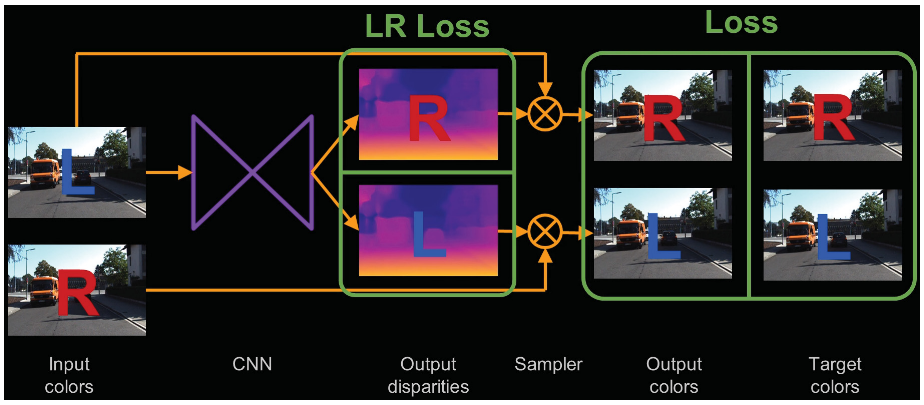


Figure 2.13: Mono Depth model concept [46]

dataset. They used 35454 and tested 4370 images. Then, the team fine-tuned the model's parameters by retained on the KITTI dataset. Both of scene-flow and KITTI have a stereo image with max disparity 192 [18]. The Skydio team found 2.87% error on dataset KITTI 2015 for D1-all pixels [18].

Another innovative model is MonoDepth [10]. As we explained before, the disparity is the shifted value between the left and the right images. Godard et al. used this idea to train a model that generates a disparity map from one single input, but both of the images used for training the model without having labels [10]. As in the figure, the model is trained to create left-to-right and right-to-left disparity maps. Then, they used a Sampler to reconstruct the input images to used them for calculating the training loss [10]. The loss function C is complex. It derived from different scaled levels C_s, which is combination of three different loss functions.

$$C = \sum_{s=1}^{4} C_s \tag{2.10}$$

$$C = \alpha_{ap} C_{ap} + \alpha_{ds} C_{ds} + \alpha_{lr} C_{lr} \tag{2.11}$$

$$C_{ap} = C_{ap}^{l} + C_{ap}^{r} \tag{2.12}$$

$$C_{ds} = C_{ds}^{l} + C_{ds}^{r} \tag{2.13}$$

$$C_{lr} = C_{lr}^{l} + C_{lr}^{r} \tag{2.14}$$

Where C is the total loss cost, C_s is loss cost for each scale level. C_{ap} is **the Appearance Matching Loss**, which is the similarity difference between the input image and reconstructed coloured image. C_{ds} is **Disparity Smoothness Loss**; it enhanced the disparity output by calculating the L1 of the estimated disparity gradient. C_{lr} is **Left-Right Disparity Consistency Loss**; it measured to improve the accuracy of disparity by reducing the difference between the left and right disparity map. α are the weight for each loss function.

The MonoDepth model is an unsupervised learning model (Semi-Supervised learning), and they used stereo vision for training. It just needs the left image or both images as input to predict the depth map. It achieves 23.8% for D1-all by using monovision and 9.2% by stereovision.

Other teams from Nvidia works to improve the same model GC-net by using the same loss function from MonoDepth [42, 10]. Their contribution summarized in three points 1) They replace the Relu and batch normalization layers by ELU activation function to make running and training model faster. 2) They add their novel layer ML-Argmax instead of the softmax layer, which used initially with GC-net. Lastly, they used this loss function, which could be used as supervised and unsupervised learning by add or remove term lider from the equation 2.15 [42].

$$C = \alpha_{ap}C_{ap} + \alpha_{ds}C_{ds} + \alpha_{lr}C_{lr} + \alpha_{lidar}C_{lidar} \tag{2.15}$$

where C_{lidar} is the difference between predicated disparity and ground truth disparity.

However, most of the depth estimation models are computationally demanding; they need GPUs with a high amount of memory like a Titan X, or GTX 10XX. The Nividia team's model speed is 320 ms on Nividia Jetson TX2 and low resolution $[513 \times 161 \times 48]$, and this the height speed [42]. Therefore, Yan at el. works to build model can be run on Nividia Jetson TX2 GPU, they achieve 97.3ms with resolution $[1242 \times 375 \times 192]$ [49]. AnyNet also inspired by GC-Net. Yet, Yan and his team made three changes to makes their model run faster with good disparity estimation [49]. Firstly, the model uses the U-net model as feature extraction. The features have used from three different levels (1/4, 1/8, 1/16). Secondly, to save resources and reduce the estimation time, the model calculated the cost volume like GC-Net to build the 3D convolution tensor from the smallest input image (1/16), then used residual prediction for calculating disparity for next stages (2 & 3). Then to improve the disparity, they used disparity regression 2.16 instead of ArgMin as an activation function for generating the disparity output [49].

$$\hat{D}_{ij} = \sum_{n=0}^{M} k \times \frac{\exp -C_{ijk}}{\sum_{k'=0} M \exp -C_{ijk'}} \tag{2.16}$$

where is the D is the output disparity of each level. k is the disparity level from 0 to max disparity M. C is the output cost between the two predicated disparity right and left. The used $L1$ to measure the loss at each stage; the total losses is the loss training [49].

We tried to use two of those models GC-Net and AnyNet, to test them on our data. However, this is what not possible because those models need big data set that has thousands of images such as the KITTI dataset or scene-flow for training [9, 26]. Using transfer learning in this type of problems, because the stereo vision setup affects the model results. Theoretically, the new dataset should be taken by the similar cameras for focal length and the equal distance between the cameras (Base Line).

2.2.4 Performance evaluation

There are many measurement scores; each one of them has specific features, and it could use with one or more machine learning algorithms. In this study, we used some of those measurements. For regression models, we used R2 and (mean absolute error) MAE, and IOU for evaluating the semantic segmentation output.

The coefficient of determination (R^2)

The R2 score is a statistical measure representing the proportion between two variances, the model's prediction variance and variance of the data. R2's value is between 0 and 1; For example, if the R2 value is 0.5 that approximately half of the model predictions' are close to the data variance. Mathematically, R^2 is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2} \tag{2.17}$$

Where the Y_i is the true value, $\hat{Y}$ is predicated value, and $\bar{Y}$ is mean of the Y.

Mean absolute error (MAE)

It measures the difference between the ground truth and the predication without considering the direction of the difference. This the equation for calculating the MAE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y_i}\right| \tag{2.18}$$

Where the Y_i is the true value, and $\hat{Y}$ is predicated value.

Mean absolute percent error (MAPE)

MAPE measures the error between the prediction and real data but in percent term. It is like mean absolute error; it calculates unsigned error, then it calculates the percentage average.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i - \hat{Y_i}\right|}{Y_i} * 100 \tag{2.19}$$

Where the Y_i is the true value, $\hat{Y}$ is predicated value, and $\bar{Y}$ is mean of the Y.

Mean square error (MSE)

MSE measures the difference between the ground truth and the predication without considering the direction of the difference. But because it uses square, maks it sensitive to the outliers predication. The MSE defined by this equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2 \tag{2.20}$$

Where the Y_i is the true value, and $\hat{Y}$ is predicated value. as shown in the example in table 2.1. The value of the MSE is 1.47, but MAE is 0.97.

Intersection over Union IOU

IOU is a measurement used for object detection model, which used a rounded box, semantic segmentation, or instance segmentation. With semantic segmentation output, most of the pixels classified as background; therefore, using accuracy score will give misleading information about the efficiency of the model. But the IOU measure

Error	$\lvert Error \rvert$	$Error^2$
0.34	0.34	0.1156
-0.06	0.06	0.0036
-1.48	1.48	2.1904
-0.12	0.12	0.0144
-1.7	1.7	2.89
-2.12	2.12	4.4944
-1.65	1.65	2.7225
-0.13	0.13	0.0169
-1.07	1.07	1.1449
-1.06	1.06	1.1236
Mean	**0.973**	**1.472**

Table 2.1: Example of the difference between the MAE, and MSE

ratio of overlapping the presentation with ground truth over the union on them. The equation calculates the IOU.

$$IOU = \frac{\text{Area of overlapping}}{\text{Area of Union}} \tag{2.21}$$

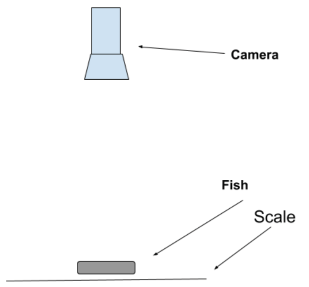
Chapter 3

Experiments

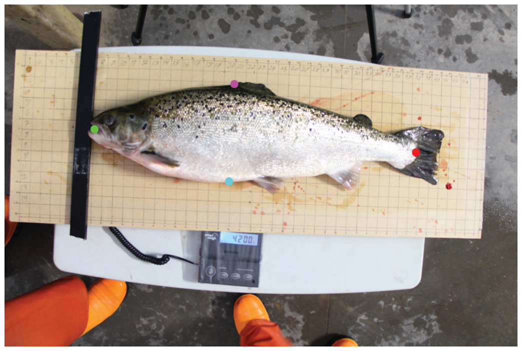
3.1 Fish dataset

We had access to the dataset contained 1275 images of salmon fish divided into five folders for different sampling days. Those fish were scaled after the harvested process. Diagram 3.1 outlines the setup to take fish pictures. Besides the weight of the fish, the dataset contains measurement information for 1208 fish of the total dataset.

The measurements in the dataset included, weight in kilogram and pound, greatest depth (width) cm, body length (height) cm, points coordinate of the fish measurement on the image(length points (P1, P2), width points(P3, P4)). Figure 3.1 shows an example for one of the fish and measurement points.



(a)



(b)

Figure 3.1: (a) Setup for taking fish images during harvest. (b) Length is (Green point is P1, Red point is P2). Width is (Fuchsia is P3, Aqua is P4). Mass is 4.2 kg.

The distance between the camera and the table was not recorded. It is important information because the depth affects the ration of mm in the real-world to a pixel. For example, the length of fish in Figure 3.1 is 614 mm and 1202 pixels. The ratio is 0.51, which means each pixel is equal to 0.51 mm in the real-world. However, we can calculate the distance in each image from the information we provided with the pictures.

By knowing the object dimensions in an image, and the real-world and knowing the camera's features [sensor, focal length], we could calculate the distance between the camera and that object by this Equation 3.1. Here d is the distance between the camera and the object, f_{mm} is the focal length of the camera. l_{mm} is the length of the fish in mm. i_{pixels} is the width of the image in pixels; l_{pixels} is the fish length in pixels. And s_{mm} is the width of the camera's sensor in mm. Depends on the metadata, which saved in the images, the camera is Canon EOS 700D, and it has a 22.3 × 14.9 mm sensor.

$$d = \frac{f_{mm} \times l_{mm} \times i_{pixels}}{l_{pixels} \times s_{mm}} \tag{3.1}$$

Figure 3.2 shows the distribution of the distance between the camera and the table for each batch of images. The batches' mean range between 79 and 86 cm. Except, Batch 10's average is equal to 64 cm. However, the last batch has 262 images, so it is close to 20% from total dataset

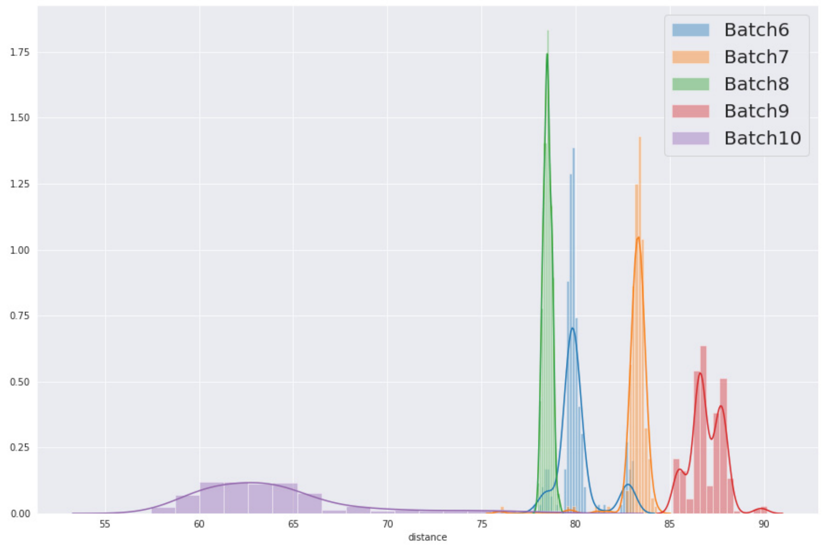


Figure 3.2: Compare histogram Plot of distances for each batch in the dataset

3.2 Estimating fish weight

This research study propose estimating the biomass of the fish from an image. Without fixing the distance between a camera and object. Therefore, we want to check the possibility of using a convolution neural network to estimate the biomass of the object by training two different models. We have conducted several experiments to test different deep learning models; the first one is using standard convolution neural networks, and another one is a semantic segmentation model.

3.2.1 Estimating fish weight with VGG19

in order to test using a convolution neural network for estimating the mass of fish from an image. We used the fish dataset, which contents the fish on the wood plate and the weight of each fish in the kg. For that purpose, we build a convolution network model that can trained to predict the fish from the images. The model has two parts, i.e. the feature extraction using VGG19 architecture. Then the regression part; it contains several full connection neural networks for applying the None-linear regression model [41].

Feature extraction

VGG19 is one of the legend deep learning models. It has been used in many image classification applications since it was invented by the Visual Geometry Group (VGG) from the University of Oxford in 2014 [41]. According to Google Scholar, over 30000 research papers cited the VGG19 paper [41].

The trick makes VGG19 useful in using two 3×3 convolution layers, which will assist to cover a large area with a fewer number of parameters. As shown in figure 3.3 shows the area could the filters from two convolution layer cover it. To know the difference, we can compare the parameter number of a convolution layer with kernel size equal to 5×5, with two convolution layer with kernel size equal to 3×3. Using one layer of 5×5 filters, the number of a parameter is $5 \times 5 = 25$, but using two layers of 3×3 filters, the number of parameters is $2 \times (3 \times 3) = 18$; which means the number of parameters to covering the same area reduced by 28% [41].

By reducing the number of parameters, we minimize the number of images required

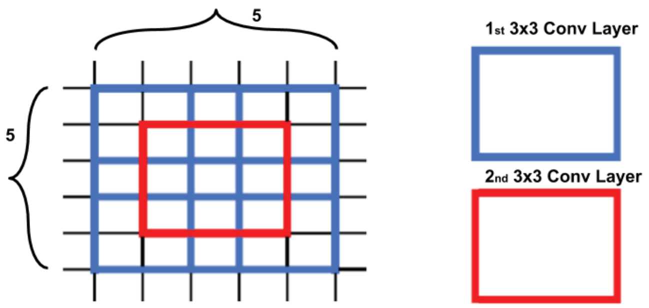


Figure 3.3: VGG19 filters covering area

to train the model with fewer parameters, the training process will be faster and avoid over-fitting [41]. It shows an excellent result to use it as an extraction feature. Therefore, we choose it as feature extraction for our regression model.

The VGG19 model has 16 convolutional layers; they divided into five blocks of convolution layer and max-pooling layers. Each of the first two blocks has two convolution layers and ends with a max-pooling layer for reducing the size of the filters. For other blocks, each block has four convolution layers followed by the max-pooling layer [36, 41]. Before we start fitting the model, the VGG19 is loaded with ImageNet weights for decreasing the time needs for training the model from random weight.

Regression part

The regression part depends on full connection layers (Dense Layer), however, the first layer is the global average pooling layer for connecting 2 dimension layers with the full connection layer. Usually, the flatten layer used to convert the output of 2 dimension layers to the Dense layer, but we found this approach is adding many parameters for the model. For this problem, our model VGG19 model output is [?, 8, 12, 512]; so, by flatting this tensor we end with 49152 parameters than using the Dense layer has 512 nodes we end with total parameters equal to 45,264,705 parameters. Where the input size is [256, 384, 3].

The regression model starts with the global avg pooling layer or the global max pooling layer followed with three dense layers in this order of output nods [512,128, 64]. Each of those dense layers activated by Non-linear function Relu. The last layer in this part are has a liner activation. Figure 3.4 shows our model architecture.

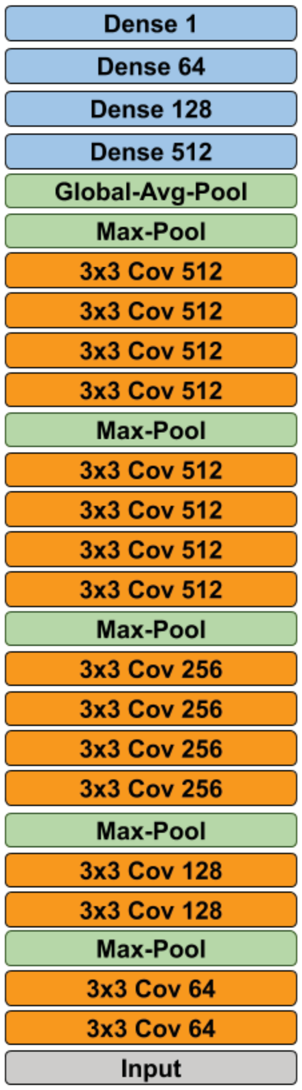
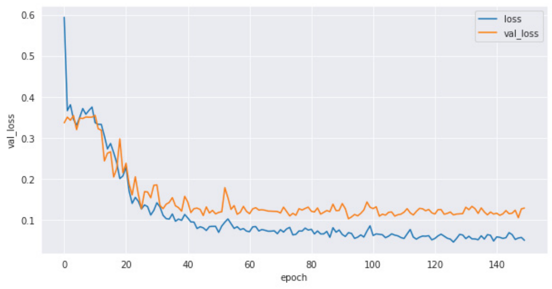


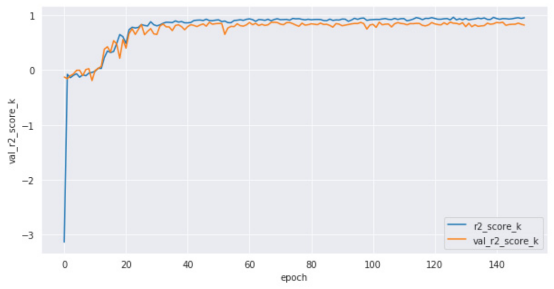
Figure 3.4: VGG19 model with Regression output

Training the model

The first two convolutional layers were frozen before starting training the model, and to get the most benefit of transform learning, which are the first block in the VGG19 model [41]. Freezing the layer means that the weight of the layers is not updated during the training. But layers like max_pooling do not have weights, so there is not any point from freezing them; just the convolution layer can be trainable or not. During the training and by fixing the weight of first two layers, the model keeps using the low-level features which learned from ImageNet date like edges and colour [36]. Those features are common for all different scenarios, even when ImageNet does not



(a)



(b)

Figure 3.5: (a) The Loss Rate. (b) R^2 score.

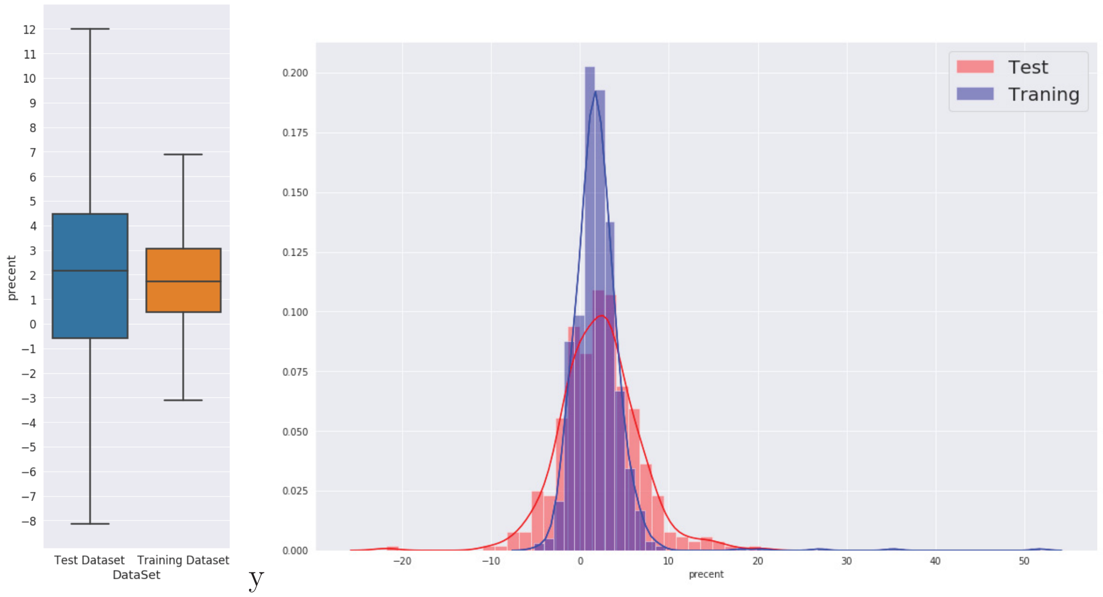
contain fish images.

The model is trained using the Adam optimizer with learning rate set to 1e-4 for 150 epochs and the loss function is the Mean Absolute Error (MAE). The learning rate decreases by 6.67e-13 after each epoch. The dataset was divided into two parts, i.e. a training set and a testing set with ratio 70% for the training dataset and 30% for the testing dataset. The training set contains 891 images and 384 images for testing, which took about 4 hours and 14 minutes using a 1080Ti GPU.

Three types of image augmentation were used (horizontal flip, vertical flip, and shift image) with a probability of 20%. We chose those functions because those processes do not change the scale of the fish in the image. But they are helping to generate a new image to improve the generalization of the model over the location of the fish in the picture. The images randomized for each epoch for supporting the model for better generalizing and for avoiding stack in a local minimum.

Results

As evident from Figure 3.5, we can notice the loss value starts high close to 0.6; then, it decreases with more training epochs for both training loss and validation loss. After the epoch 35, we can see there is a difference between the training and validation loss and that gap persists til the end of the training. We keep the best model's weights at a highs validation R^2 score, which happened at epoch 126 with a rating of 0.88. That point is shown on the chart as red point. As we found, there is a gap between the two loss values, which usually show over-fitting. The behavior of the loss rate could possibly be because of the over-fitting of the training dataset.



y

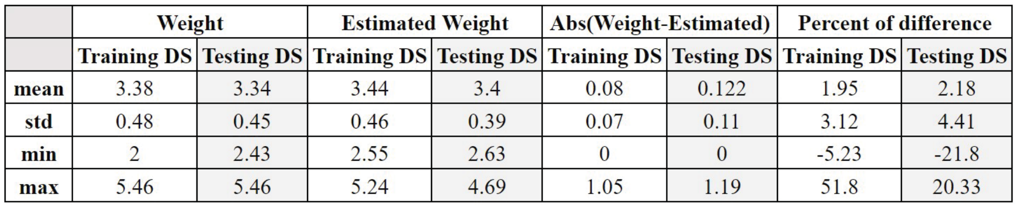
Figure 3.6: The VGG19 Regression model results for both training and testing dataset. For left, the Box plot and histogram plot to the right.

In order to generate the histogram plot, as shown in Figure 3.6, we calculate the percent of the error between the estimated mass and the real mass for each image that depends on Equation 3.2.

$$percent\_error = ((\hat{Y} * 100)/Y) - 100 \tag{3.2}$$

where Y is the real weight of the fish, and $\hat{Y}$ is the estimated weigh. For example, if the actual mass is 5 kg and the estimated 4.5 kg, that mean estimated mass is less than the real mass by 10%.

Figure 3.6 shows two distributions. Both of them are fitted with a normal distribution, and the mean for the training dataset is 1.95% and 2.18% for the test dataset's distribution's mean. That means the model is estimated more weights from the actual weight for both training and testing datasets. However, the difference between both means is about 0.23%, with benefit to training dataset's mean because it is closer to zero the 0.23% equal 0.77 kg from the mean of the actual weight. But the more significant difference between the distribution is the Standard Deviation (STD), where the training dataset's std is 3.12% but 4.41% for the testing dataset. Also, the difference between them is not high; it is about 1.29. That means the training dataset's estimated weights are more close to the mean from the testing dataset.



	Weight		Estimated Weight		Abs(Weight-Estimated)		Percent of difference	
	Training DS	Testing DS	Training DS	Testing DS	Training DS	Testing DS	Training DS	Testing DS
mean	3.38	3.34	3.44	3.4	0.08	0.122	1.95	2.18
std	0.48	0.45	0.46	0.39	0.07	0.11	3.12	4.41
min	2	2.43	2.55	2.63	0	0	-5.23	-21.8
max	5.46	5.46	5.24	4.69	1.05	1.19	51.8	20.33

Figure 3.7: Table of regression results of VGG19 model

So, the training dataset results and testing dataset results are mostly close. Also, one of the points we can notice that in the training dataset that one of the images shows a high percent error over 50%. Mostly this error happened because of the shifting augmentation where some images have shifted, not on the center. And by applying the shift function, a fish part of the fish will be out of an image and cause this error.

but to get a better view over how much of the data are under the distribution, we used the box plot.

In the box plot, has two boxes, i.e. orange represents training dataset result, and blue represents testing dataset result, as shown in figure 3.6. The space between the minimum and maximum is covering 99.3% of the data. Which means over 99% of the training is in range between 7% to -3%. However, testing's results is in range between 12% to -8%.

In general, the model gives a good result as a starting point for this study. Therefore, we worked on examining the VGG19 model by changing the augmentation probability. Besides that, we checked if there is a difference between the global max-pooling (GMP) and the global average pooling (GAP). There are three criteria for comparing the results of the models. First, the means of estimating weight closer to zero. Second, reduce the difference between the training dataset and the testing dataset. Third, the box plot shows the percentage of samples to error, which helps to percent error range for the majority of samples to compare between the models.

The results of the training models are shown in Figure 3.8. Further, both GAP and GMP present a close result for all augmentation probability. However, by comparing average error percent, we can see that the GMP layer results show the mean is closer to zero for both training and testing datasets, with a small difference from the GAP layer.

While comparing the outcomes of two approaches GAP and GMP, we choose to

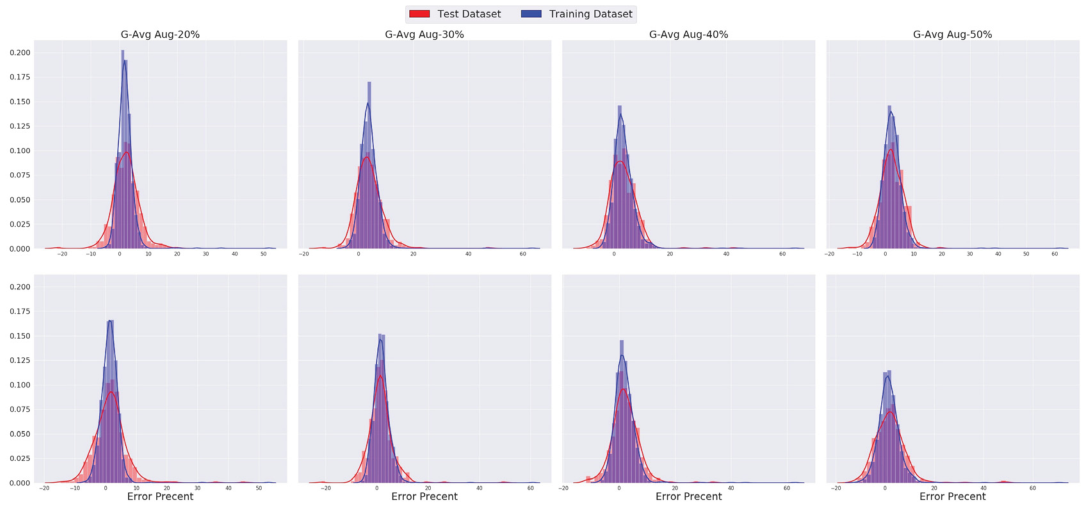


Figure 3.8: VGG19 model with Regression results

distinguish between them at 30% augmentation as an example. The average percent error of the testing dataset for the GMP model is 1.44%, but it is 3.82% for the GAP model. Nonetheless, the std for both model are very close. Std for the testing dataset of GMP is 5.05%, and it is 5% for the same dataset of GAP.

Further, to check the difference between the training and the testing dataset results. However, the model GAP, the absolute difference between the means of the training and testing dataset is 0, but for the Std is 1.28. On the other hand, the GMP model shows 0.38 for the same average absolute difference and 1.42 for the difference between the Std.

Moreover, comparing the results of those models in Figures 3.9 and 3.10. Figures are shown the error of 99.3% of data, which explains the error range for the majority of data. For GMP model is between -5% to 8%. Yet, the range for GAP is between -2.5% to about 10.5%.

Another plot is 3.11c; we plotted it to investigate if there valid evidence about which model reveals the best result for the fish dataset. The X-axis is the augmentation level, and the Y-axis is the Mean Absolute Percent Error (MAPE) result for the estimation output. From the plot, we can recognize that GMP model trained on 30% revels the smallest values. For the training dataset, MAPE is 2.6% and 3.4% for the testing dataset.

By matching the tree criteria we put with the results, we can say that the model

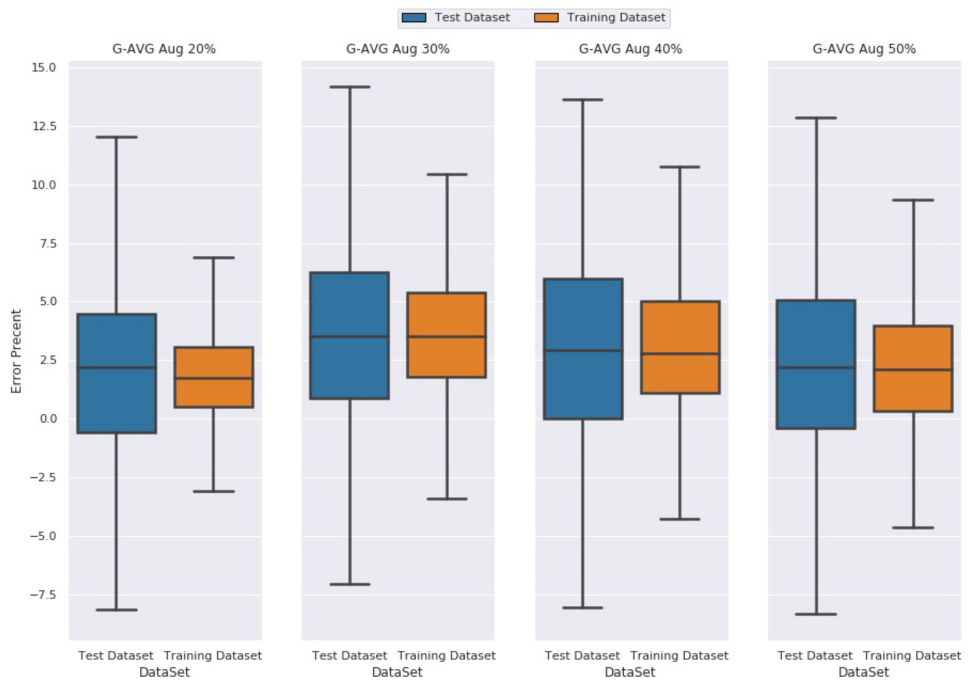


Figure 3.9: VGG19 G-AVG Box Plot

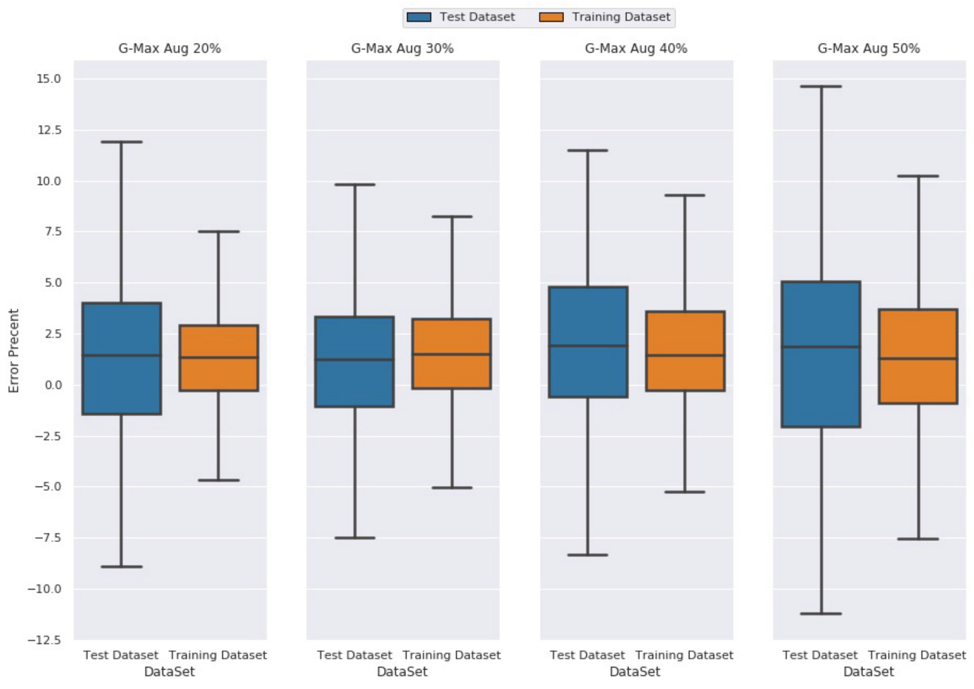
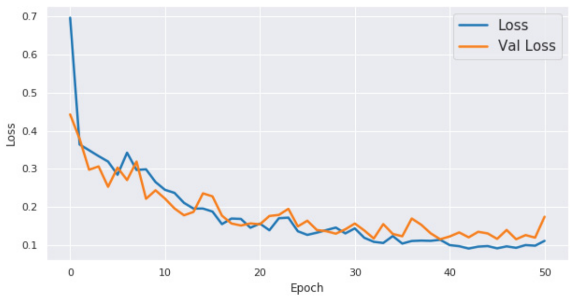
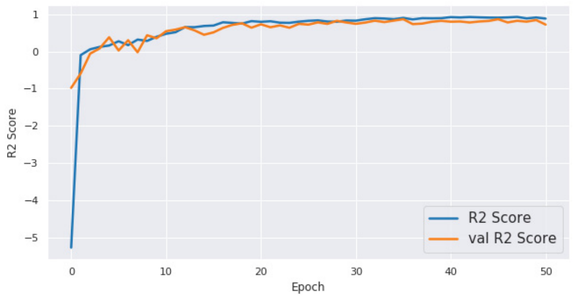


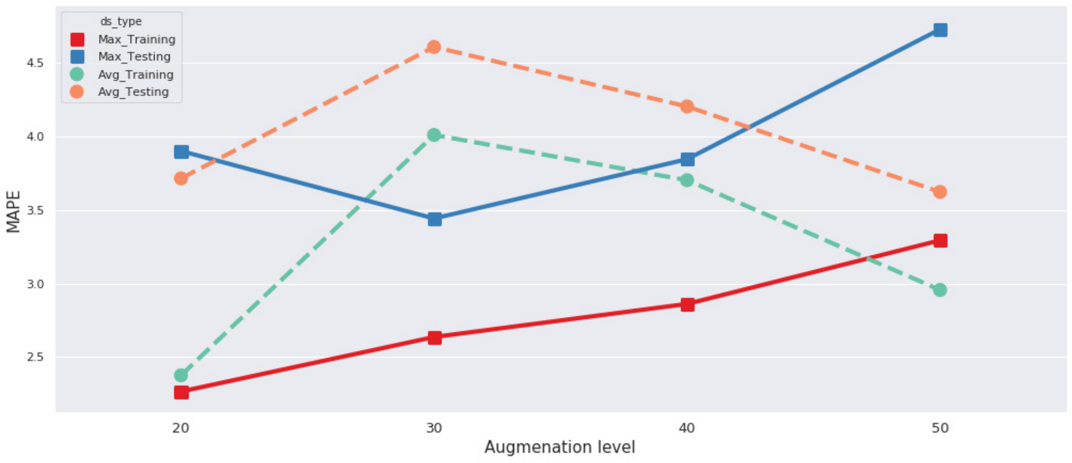
Figure 3.10: VGG19 G-Max Box Plot



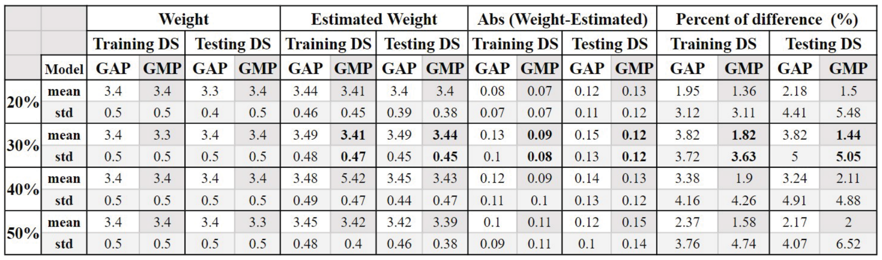
(a) Loss rate of GMP 30%



(b) R^2 score of GMP 30%



(c) MAPE



		Weight				Estimated Weight				Abs (Weight-Estimated)				Percent of difference (%)			
		Training DS		Testing DS		Training DS		Testing DS		Training DS		Testing DS		Training DS		Testing DS	
	Model	**GAP**	**GMP**	**GAP**	**GMP**	**GAP**	**GMP**	**GAP**	**GMP**	**GAP**	**GMP**	**GAP**	**GMP**	**GAP**	**GMP**	**GAP**	**GMP**
20%	**mean**	3.4	3.4	3.3	3.4	3.44	3.41	3.4	3.4	0.08	0.07	0.12	0.13	1.95	1.36	2.18	1.5
	std	0.5	0.5	0.4	0.5	0.46	0.45	0.39	0.38	0.07	0.07	0.11	0.12	3.12	3.11	4.41	5.48
30%	**mean**	3.4	3.3	3.4	3.4	3.49	**3.41**	3.49	**3.44**	0.13	**0.09**	0.15	**0.12**	3.82	**1.82**	3.82	**1.44**
	std	0.5	0.5	0.5	0.5	0.48	**0.47**	0.45	**0.45**	0.1	**0.08**	0.13	**0.12**	3.72	**3.63**	5	**5.05**
40%	**mean**	3.4	3.4	3.4	3.4	3.48	5.42	3.45	3.43	0.12	0.09	0.14	0.13	3.38	1.9	3.24	2.11
	std	0.5	0.5	0.5	0.5	0.49	0.47	0.44	0.47	0.11	0.1	0.13	0.12	4.16	4.26	4.91	4.88
50%	**mean**	3.4	3.4	3.4	3.3	3.45	3.42	3.42	3.39	0.1	0.11	0.12	0.15	2.37	1.58	2.17	2
	std	0.5	0.5	0.5	0.5	0.48	0.4	0.46	0.38	0.09	0.11	0.1	0.14	3.76	4.74	4.07	6.52

(d) VGG19-Regression model results’ table

Figure 3.11: VGG19-R results

GMP with 30% augmentation fits most of those criteria. However, the results do not show valid evidence that GMP model is better in all cases. Yet for this fish dataset and after tuning the model, we got this result. Table 3.11d shows the results by numbers for all the models.

the analyzing of the fish dataset reveals that each batch of images has a different distance between the fish and the camera. Which means the distance or pixel size is a necessary coefficient to estimate the mass. They could assist to reduce the error of estimating the weight of the fish.

We added extra input to the VGG19-R model, where this input is taking the distance directly to the regression part in the model. To train the model for scaling, the estimated weight depends on the distance. Despite that, the result did not change as much an anticipated. By comparing the best result of both models, we found that MAPE decreased by 0.47 for the training dataset and 0.41 for the testing dataset.

In fish dataset section, we discussed the average distance for each batch of images. Most of the batches have a close average, except for one batch. So, a small part of the data has a different distance, which explains the slight improvement of adding distance to the model's input.

At the end, the model can estimate the weight of harvested salmon fish fairly well, With the assumption, that the fish has the same scaling in all images. In other words, the distance between the fish and the camera is identical for all photos. Besides that, we found how the max-pooling and average-pooling sometimes give a different result in some situations.

3.2.2 Estimating fish weight with semantic segmentation

The proposal for this experiment is building a model using semantic segmentation classification to classify the salmon fish images and create black-white masks to be like a filter for each pixel in the picture, close to Konovalov et al. approach [19]. Yet we used a smaller semantic segmentation architecture from the one they used in their paper. Lighter models could be run faster in case we need to use it with video data; besides, it could run on embedded chips like Nivida Nano Jetson or Google Coral.

FC-Densenet is a useful model that shows excellent results with a few numbers of parameters. The number of parameters is the main factor to know the amount

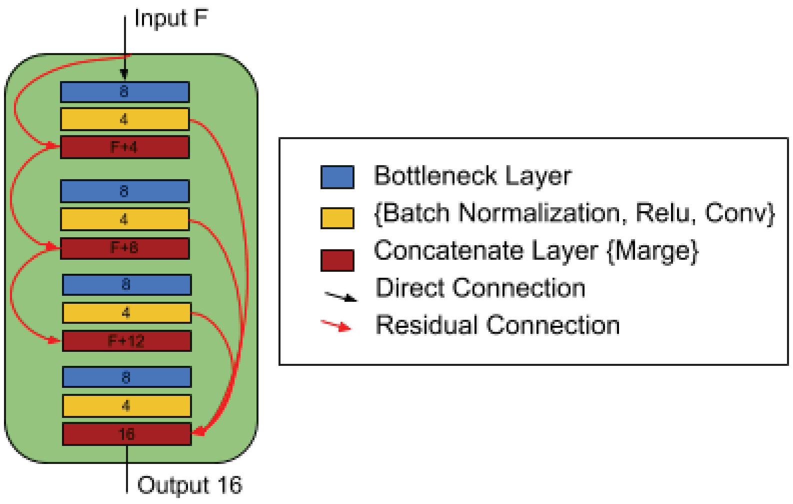


Figure 3.12: Dense Block of our model

of memory that is needed to run the model [16]. The architecture of our model is inherent from FC-Densenet has been tested on Crops image by Lottes et al. [22]. They created a stereo output model, where one of the output is for classifying a three different class [Crops, weed, and soil]. The other branch is for representing the location of plants' stems in the image. The model is content to parts encoder and decoder. It contents three dense blocks in each one. We chose this architecture instead of the original one because it configured to fit a small dataset with a few numbers of classes like our dataset [22].

First, we trained the model on the fish dataset, but it shows sensitivity regarding the light and white colour. The reason is that there is a different light situation, and the scale plate is different for some batches. Therefore, the model was modified to increase the parameters to fit those data perfectly and improve the accuracy of the model.

There are four blocks in our model, as opposed to the three proposed in the original paper [22]. Every dense block starts with bottleneck for reducing the number of input filters from F filters to 8; So, the filter size keeps as is, but the cut will be on the dimension of the filters, that saves computation cost. This step is followed by [BatchNormalization layer, Relu Activation layer, and Convolution layer], which they produce four filters from the previous eight filters. Then the model concatenates the block's input with those four filters to make the input for another loop of the same layers [BottelNeck, BatchNormalization layer, Relu Activation layer, and Convolution layer]. Those steps repeated four times. At the end of the block, all those four filters' outputs were then concatenated in one tensor. The result of the concatenation is a tensor has 16 filters, and this tensor is the output of the block. The diagram 3.12 shows the black structure of the dense block.

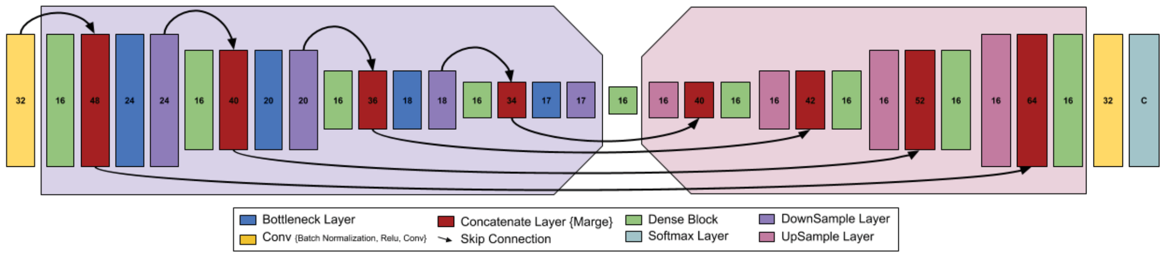


Figure 3.13: semantic segmentation model architecture

In the FC-Densenet paper, the number of features after the concatenate layer is called the growth rate [14]. In our model, the growth rate is 16 features. Accordingly, the encoder has four blocks, each one followed by concatenate, where it marge the output of the block with block's input, then the bottleneck layer and down-sample layer. The encoder ends with the block Dense.

The repeating complexity inside each block and between the blocks gives three advantages: first, it encourages the reuse of features. Second, it decreases the vanishing-gradient effect. Last, it reduces the number of parameters without decrease the model performance [14, 16].

The decoder is the inverse operation of the encoder, where it replaces the down-sample by the up-sample layer, and it does not have a bottleneck layer. Further, there is a skip connection between the concatenate layer in the encoder and the decoder to send information from the encoder to decoder for improve up-sample image [13, 22]. In the end, our model has about 72000 parameter. compared to the 11.5M parameter of the original Linknet [5].

Training semantic segmentation model

A total of 296 images were used to train the model. In addition, we applied four types of augmentation flip (vertical and horizontal), shifting, and brightness with a probability of 50%. The brightness augmentation added for this model because the full dataset had taken in different batches, and there is a different light situation for some of those batches. The brightness augmentation should support to generalize the model over the light difference. Besides the augmentation, the images smoothed by Gaussian blur function with $mean = 0$ and $std = 1$, and normalized to the range [0,1]. The model takes image size as [384, 256].

In contrast to Konovalov et al., we used a coloured image instead of grayscale images [19]. By looking at examples from their datasets, we can see that put on a white plate does have any shape pattern, which makes an excellent contrast to show the fish. The fish dataset we have, the salmon fish put on wood plat has a squares pattern, which makes it hard for our model to learn by using the shape as the only information.

The Adam optimizer is used to train the model with a learning rate equal to 10^{-3}, decay equal to 6.67e^{-7}, and Intersection Over Union (IOU) loss function [34]. Because of the limited data size, just 15% of the data is used 52 for testing. In Figure 3.15, there are examples of testing and training images output, ground truth, and input image. And in Figure 3.16 show the other example for images is not annotated.

Before going to discuss the next step, it is good to explain some points that we noticed in the loss rate of the model Figure 3.14; we can see there are some peaks in the validation loss rate before and after the best model point. Because of two reasons. Firstly, the data set is small; it has 30 images for validation. Plus, we applied the stochastic gradient descent method. Therefore, there are five batches for each validation iterations after each training epoch. Secondly, the validation data set shuffled for each iteration. That means there is a high probability for some images which hard to train to be in one batch and shift the batches' mean to high value. However, those peaks are getting smaller with increased model training.

Another fact we noticed from the semantic segmentation output is for some of the testing dataset's images show sensitivity for the white colour. For example, the predicted mask at the second-row in the left column has a small white spot. Also, two of the images in 3.15 show a similar error in white colour from the balance's plane. Mostly, this comes from the white scales on the fish under the lateral line.

Nevertheless, those errors do not affect the result too much because it is a small area added to the output mask. By looking at the testing dataset mean IOU score is higher 98%. So, the noise in the label is less than 2%; the model will ignore them.

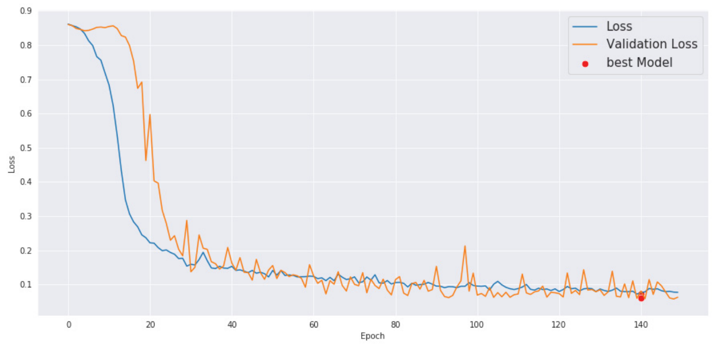


Figure 3.14: The loss rate for traning FC-Densenet model

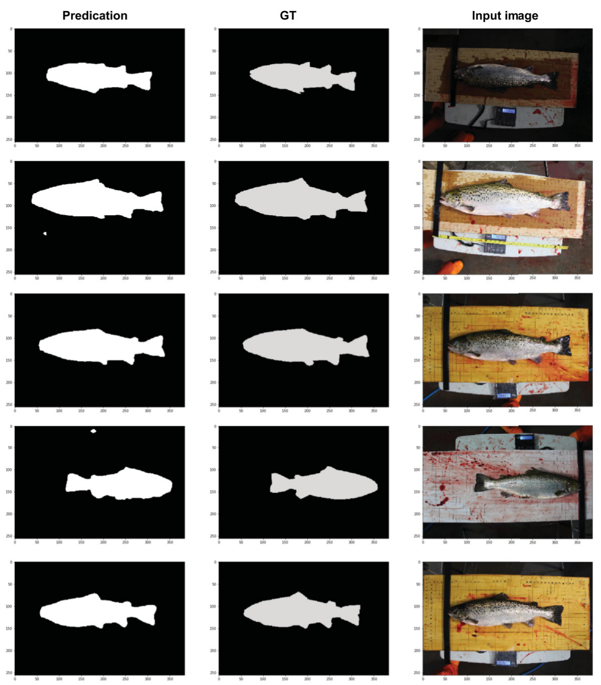
The semantic segmentation model works as a feature extraction. It converts the pixels' of the fish in the input image to white, and the other pixels to black. Then, we want to add a regression part to the model that can estimate the mass of the fish from the size of the white area from the previous section.

Accordingly, the dense layers added to the model's, similar to the VGG19-R model in the last experiment. Then train this part to use the white area from the output mask to estimate the fish's weight. Strictly speaking, the fully connected layers should train to find the relation between the number of white pixels in the mask and the mass of the fish. That means it will train to fit the correlation between the fish's area in the image and the weight of it [37].

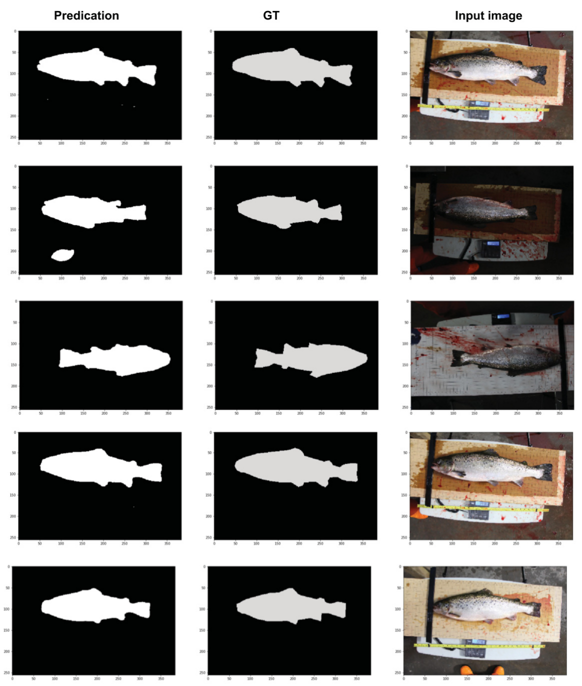
Regression part

It starts with the Lambda layer for rounding the "Sigmoid" layer to create a Black/White image by rounding the probability output to [0, and 1]. Then several Dense layers added to the model with those numbers of nodes [512, 64, 16]. All those layers in the regression part have a Relu activation function. The regression part that ends with the Dense layer has one node with the linear activation function, which shows a better results from the Relu function. It is used in the VGG19 model mostly because of the input data differences.

One dropout layer added after the first dense layer for increasing the regularisation, which helps to avoid the over-fit the training dataset. The dropout layer does regularisation by setting the weight to zero for some nodes, and this changes for each epoch. The dropout regularisation reduces the weight of the nodes, which leads to



(a) Training DataSet



(b) Testing Dataset

Figure 3.15: Some examples of the Semantic Segmentation predication From the annotated data

avoiding over-fitting.

Training model

In contrast to the last model, we froze all the layers of the semantic segmentation layers and just used the output of the model as input to the regression part. For training, we used 85% of the dataset and 15% for testing. We add augmentation to the data with probability 35% and three different functions [vertical and horizontal flip, and shift]. The learning rate is 0.01, with reducing equal to $5 * 10^{-8}$ after each epoch, and batch size equal to 32 images.

Figure 3.18 shows the loss of the training process for 250 epochs, yet we used a checkpoint technique to save weights at a higher value for validation R^2 score, this also helps to avoid over-fitting. The best weight was achieved at epoch 207, with the loss value equal to 0.2 for the training dataset and 0.24 for the testing dataset.

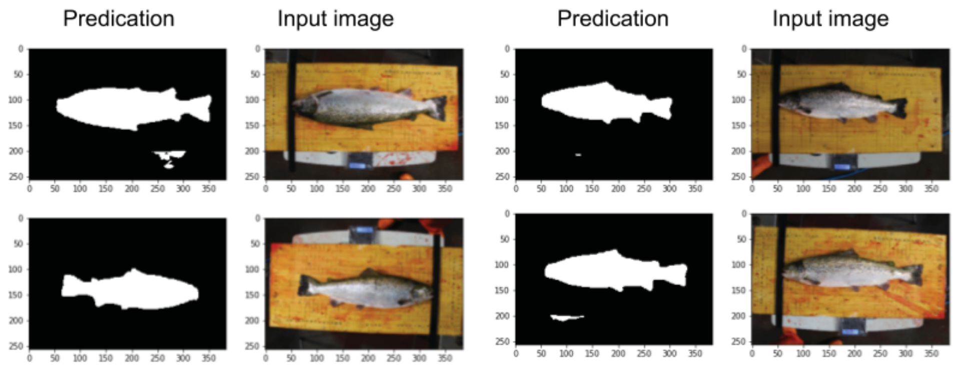


Figure 3.16: Some other examples for model predication of data not annotated

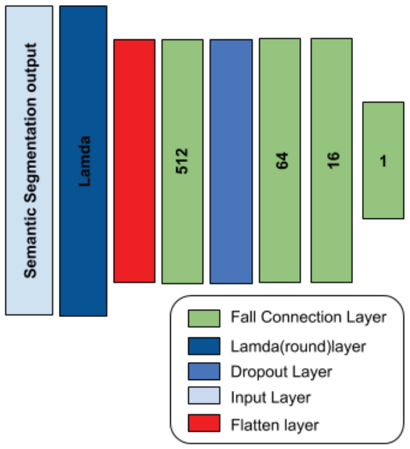
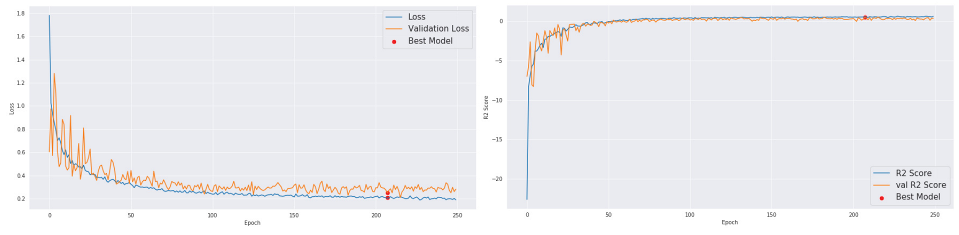


Figure 3.17: The regression part of after the semantic segmentation model

Results

The result of the FC-Densenet-R is close to the VGG19-R model. As shown in Figure 3.19, over 99% of the data estimated mass in the range between -20% to 14% error for the testing dataset, and -15% to 11% for the training dataset. But to compare the out models to LinkeNet-34R model, we calculated the MAPE equation 2.19 [19]. The table 3.1 shows the result of our models VGG19-R and FC-Densenet-R; also, it shows of LinkNet-34R. The VGG19-R reveals the best result compared to the other models.

Moreover, we compare the results of both models VGG19-R and FC-Densenet-R, that for understanding the difference between them. We plotted in Figure 3.20 the estimated mass on the Y-axis and the real mass on X-axis. The plot shows the relation between the actual weight and the estimated one, and the optimal solution should be



(a) Loss rate (b) R^2 score

Figure 3.18: Learning Rate and R^2 Score of the FC-Densenet-Regression model of 50% Augmentation

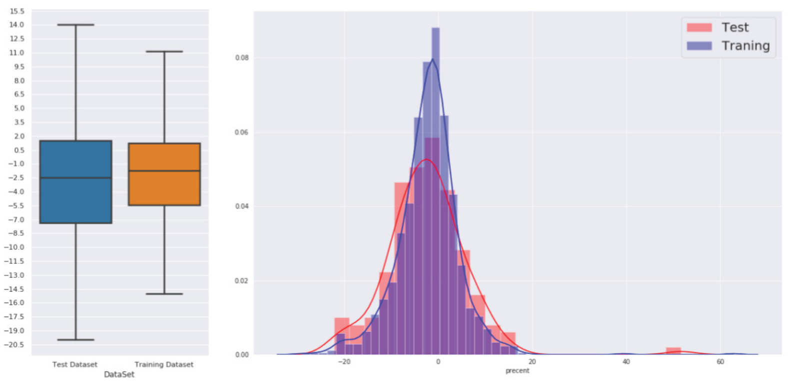


Figure 3.19: The regression part of after the FC-Densenet-R model

Model	R2		MAPE %	
Dataset	Training	Testing	Training	Testing
VGG19-R	0.87	0.9	2.38	2.4
FC-Densenet-R	0.56	0.52	4.93	6.49
LinkNet-34R	-	-	4.27	4.2

Table 3.1: The results of the models [VGG19-R, FC-DenseBlocks, and LinkNet-34R]. [19]

like the red line in Plot 3.20. The blue group is a result of the FC-Denasent-R model, and the orange group is for VGG19-R. The plot shows two main points. Firstly, both of the models fit the distribution of the data, where the regression lines indicate a positive relationship between the real and estimated weights. Secondly, the chart shows most of the errors that come from the fish which have weights over 3.75 kg. That error increased dramatically for weights over 4.5 kg.

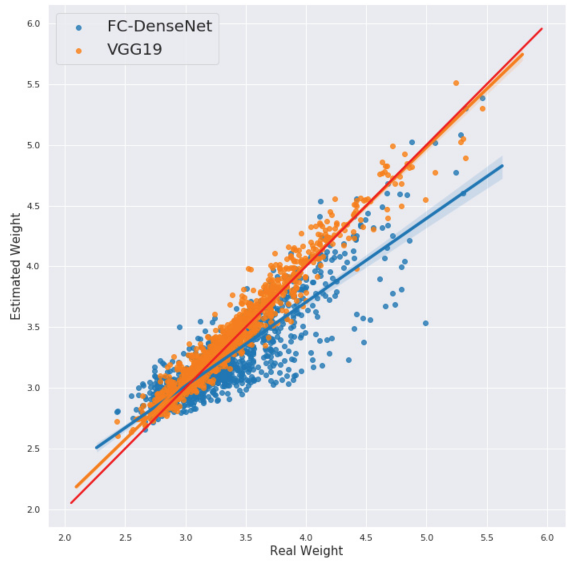


Figure 3.20: Plot for corolation between the estimated weight to real weight

3.3 Stereo model

3.3.1 Stereo dataset

In the previous experiments, we tested a different type of deep learning models for estimating the fish mass from one input image. For both of the models, VGG19-R and FC-Densenet-R show that the deep learning and computer vision methods could help to predict the fish mass just from the image. However, the main problem is to estimate fish weight inside the water tank. Where the fish is alive and moving; besides, the distance of it from the camera will be changed.

Solving that problem using CNN required a dataset to use for training the model. So, we need to have a camera that can work inside the water; those cameras are usually expensive and beyond our budget, for this thesis. Also, there are other logistics problems of taking a new dataset, like getting permission to take new images and the risk of pollution of fish water. Therefore, for this study, we created a dataset that can simulate fish inside the tank. Fish could be on different distances from the camera, and they are in different directions.

In order to simulate a fish, we used plastic Lego blocks. As objects the Lego blocks have different shapes and sizes. We change the location of the block from the camera; In addition, we placed the objects at different angles of the camera. As a

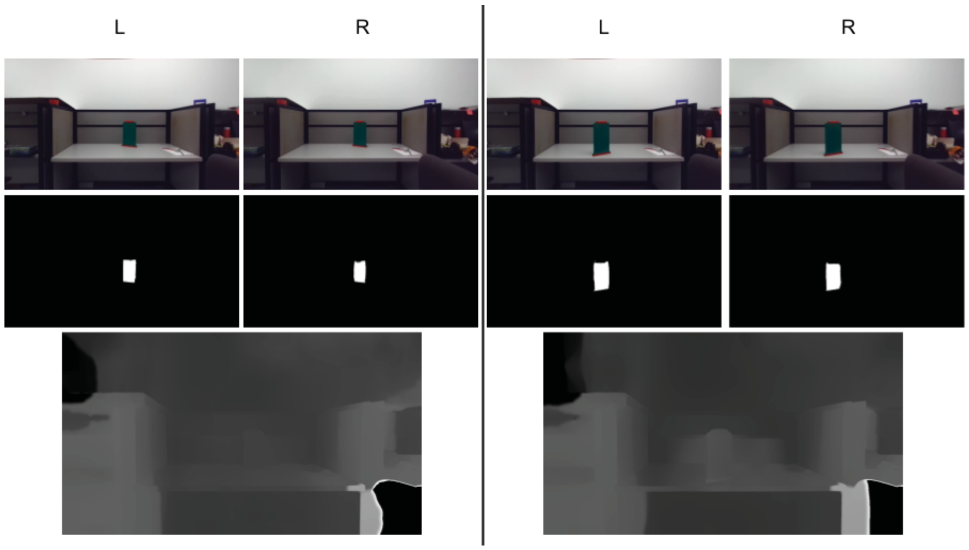


Figure 3.21: Two examples for the same shape, but it is in different locations and directions.

consequence, the model needs more than an input image to use as a scaling factor. In other worlds, we need to know the depth information to aid the model for re-scaling the pixel size in depend on the depth.

$Area_{cm^2}$	$Width_{cm}$	$Height_{cm}$
609	19	32
608	29	21
580	10	58
570	38	15
380	38	10
150	10	15

Table 3.2: The blocks dimensions

The stereo vision used to take the images for the dataset [right and left] to generate the depth map (disparity map). The image were taken by using ZED Camera. The ZED camera is a commercial stereo camera used for industrial applications. It can create a disparity map, and from it, the ZED camera generates the depth estimation map from stereo images for each pixel in the picture. Camera provides: right, left, disparity map, and depth map.

Using the colour filter for the green colour to convert the images left and right to mask pictures. Those masks are similar to the semantic segmentation mask in the previous experiment. The photos have five different areas in $cm^2[150, 380, 570V, 580H, 608, 609]$

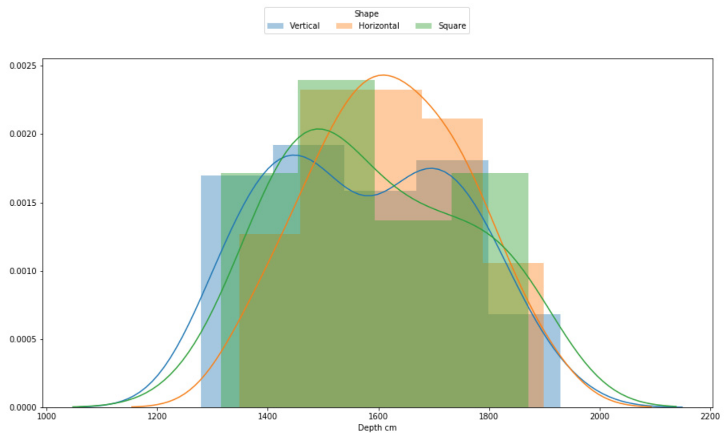


Figure 3.22: The distribution of the depth between the stereo camera and the center of the blocks.

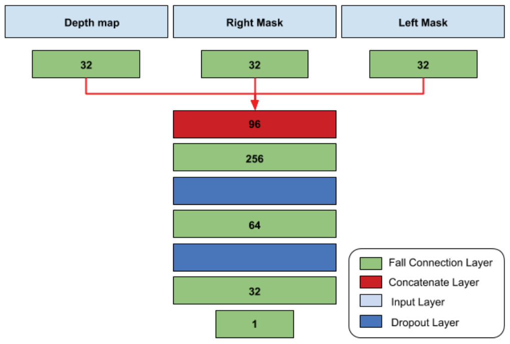


Figure 3.23: Stereo model used three images as input.

and random distances. The figure shows the distribution of estimating depth by the camera to the center of the blocks. The estimated depth is between about 1.2 M to about 2 M. We shot 132 images (22 images for each area), they are enough for training the model more details about it in the next section.

Figure 3.21 shows the two examples from the stereo dataset for the same block. In the left images, the block rotated to the left, and it is far from the camera by 1800 cm. The right block turned in the right direction, and it is away from the camera by 1433 cm. However, the mask of the left block is smaller than the right block.

3.3.2 Model architecture

The first model we develop takes the images as inputs [right mask, left mask, and depth image]. Figure 3.23 shows the model structure. There is a flatten layer, and the fully connected layer has 32 nodes after each input to convert the image from two-dimension to one-dimension. Then, the fully connected layer (FC) will work as an encoder to the image and comprise the data in just 32 nodes.

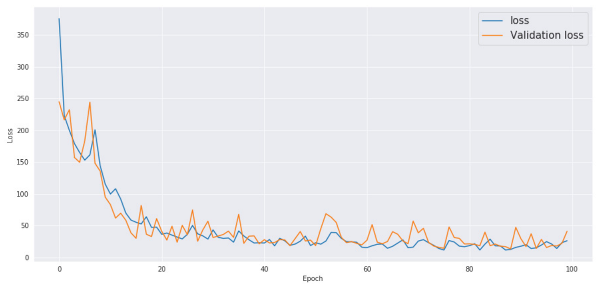
After that, we combined all the fully connected layers in one layer for processing images' data together in the next several layers. The regression part in this model contents many layers [FC 256, dropout layer (0.4), FC 64, dropout layer (0.2), FC 32, FC 1]; all the FC layers followed by the relu activation function layer. While we have a small data set, and it is hard to add many augmentation functions without effect the image scaling, adding the dropout layer for avoid over-fitting the training data.

Our assumption is the model's regression layers should learn from the combined input the correlation between the area of the object in the image and scaling it depends on the depth. The learning rate is $5e-05$, and decay is $2e-07$, but for the augmentation, we used the flipping and shifting with 50%, and for each batch, the images randomized. And, the inputs normalized in the range [0-1].

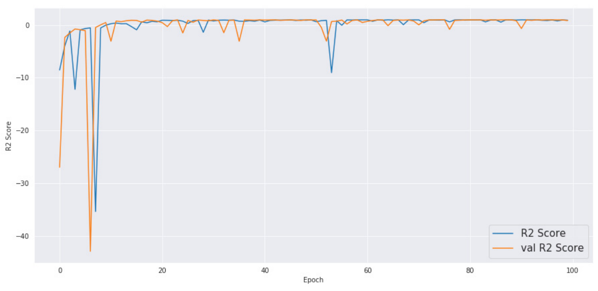
The mean absolute percent error for training is 5.76%, 17.14% for the testing dataset. This result shows the model mostly is remembering the training dataset with learning any features. Even by trying to change model layers nodes number, that does not improve the model's result. That leads us to think for different way can train the model and gives more efficient results.

VGG19-R shows a significant improvement in estimating the biomass of fish from the images. However, providing the model with depth map will not be practical because, unlike the fish's dataset, the blocks have a small area in the photos and located in multiple places in each image. Therefore, we concatenate the masks with a depth map to generate a tensor with three dimensions [left mask, right mask, and depth map]. The left and the right mask train the model to focus on the block and their area. It should be the output of the model, but the depth map should support the model to re-scaling the output area depends on the depth.

Like the last experiment, we built the VGG19 model first and loaded the Imagenet's weight before starting training the model. We froze the first two convolutional layers to make the fitting process faster. Then we train the model on the stereo dataset, with a learning rate equal to 3e-5, 100 epochs, eight images as batch size, the input size is (640, 360, 3), augmentation probability 60% and 26 images for testing



(a) Loss rate



(b) R^2 score

Figure 3.24: Learning Rate and R^2 Score of the VGG19-Regression model on the stereo dataset

and 106 images for training with 10% of them for validation. All the input images normalized between [0,1].

As we expected, the model gives a result with MAPE error 1.8% for training and 2.37% for the testing dataset. The histogram plot in figure 3.25 shows the percent error between the training and testing dataset. We can notice there are two peaks in the training and testing dataset error. Plus, the box plot shows the range of the training error is significantly bigger than the testing error.

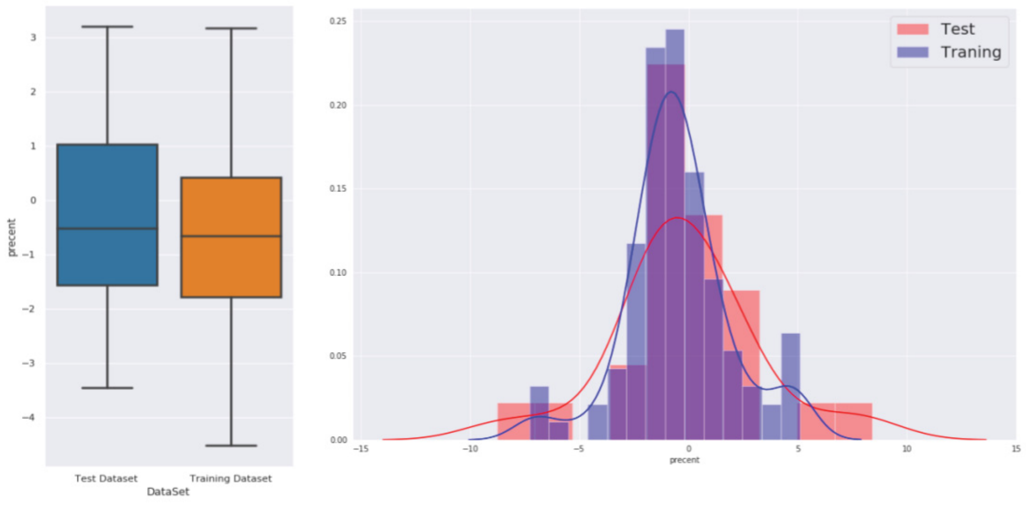


Figure 3.25: Estimating area

Therefore, we analyze the model result depends on the area of the block. As is shown in Figure 3.26, there are six different plots, one for each block. Starting from area 150, the distribution for both training testing dataset is over zero, and the mean is 2.37% for the training dataset and 4.45%. Differently, the area 380's plot shows

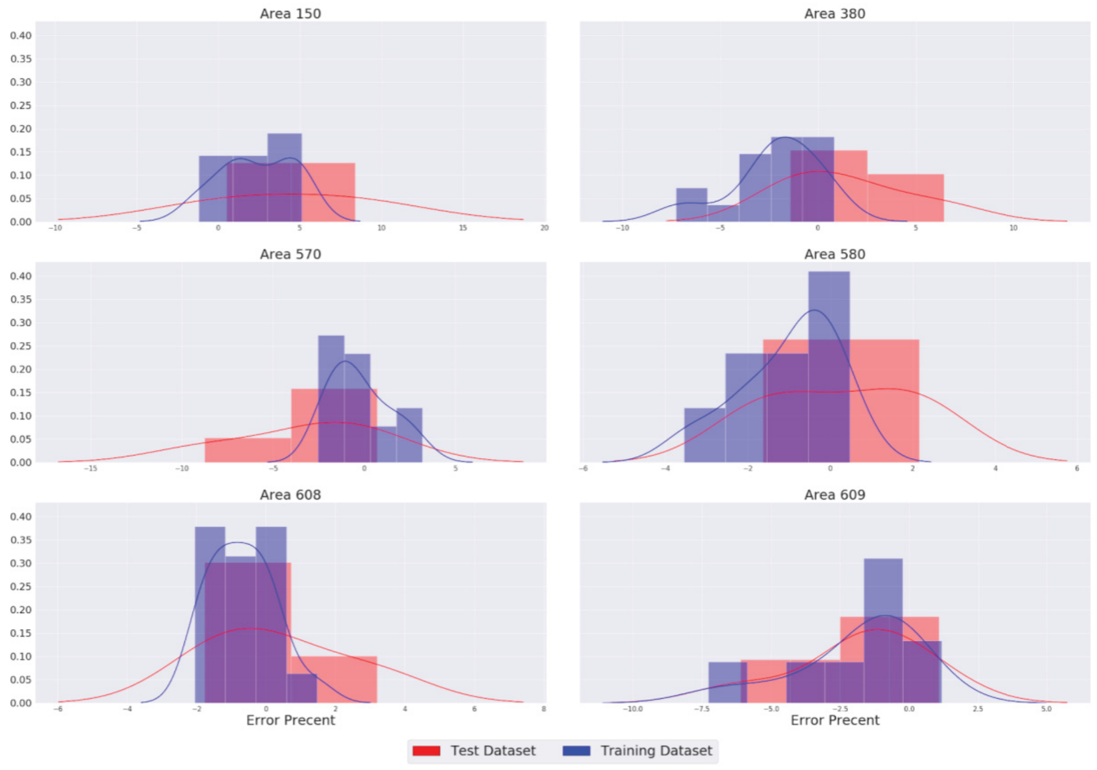


Figure 3.26: Error percent for each area

that the training dataset has an average equal to -2.18%, but the testing dataset's mean 1.45%; Which implies some training dataset for this area estimated less than expected, and some testing dataset images higher than real area.

However, looking at the other plot, they show a similar outcome. But the mean for those disruptions is more closer to zero from the previous ones. Also, the difference between the training dataset and the testing dataset is smaller. For example, the mean for area 580 is -0.23% and -3.21% with this order training and testing. And for area 608, they are -0.69% and 0.34%.

Moreover, we check if there is a relation between the object depth and estimated error. Therefore, we plot Figure 3.27, where the X-axis is the absolute error, and the Y-axis is the depth from the camera to the center of the block. As is shown in Figure 3.27, the regression line is horizontal, which means increasing the distance from the camera is not growing the error of estimating the area.

Furthermore, we checked the effectiveness of the depth input. To know how much the depth information assists the model. To do that, we trained the VGG19-R model

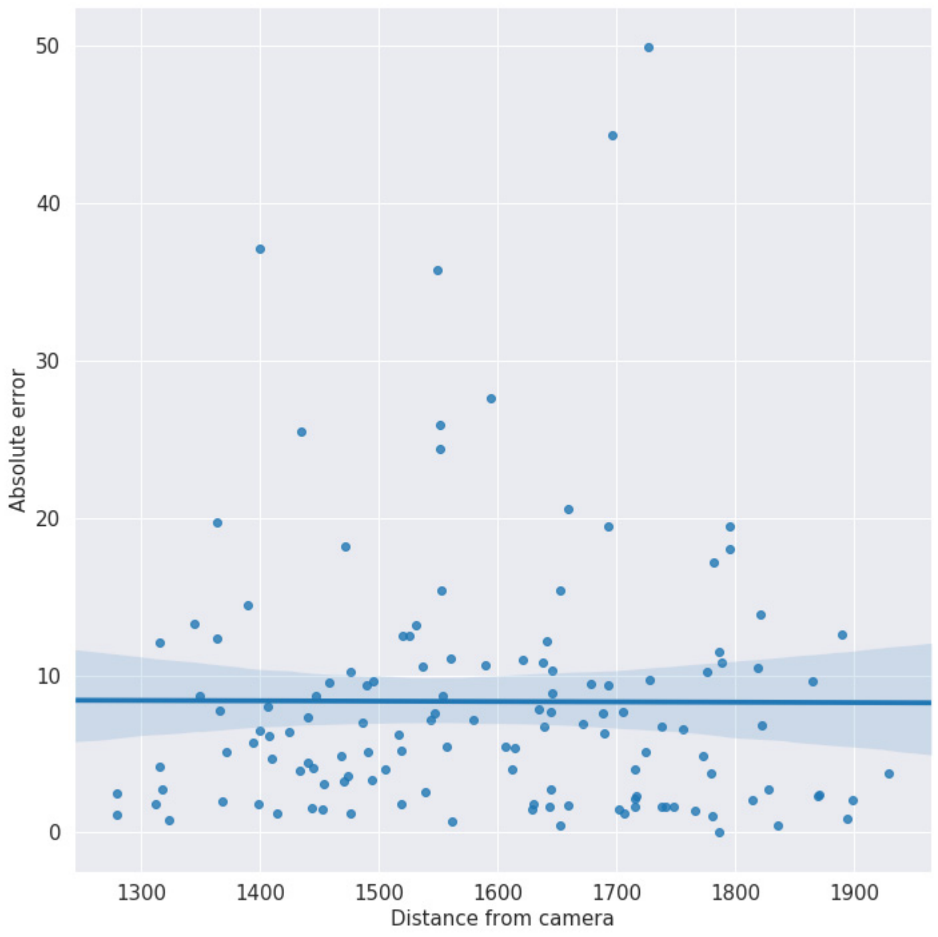


Figure 3.27: Absolute error to Error Percent

with the same dataset but without the depth map. The model fed with right and left masks. After training the model, the outcome of this training is 12% MAPE and for testing 12.7% MAPE, which means the depth map help to improve the model estimation area. Also, the model could not to learn estimate the area of the model just from the mask images.

The model reveals good results with the stereo dataset; even this is just a concept model, it could use for real problems like estimating biomass of fish under the water, we wish to have real data like that to test the model.

Chapter 4

Conclusion

The research study was successfully conducted in developing a deep learning model for estimating the biomass of harvested fish from images. We did this using two different models, VGG19-R and the semantic segmentation model FC-Densenet-R. The VGG19-R achieved a 2.4% MAPE in testing estimating weight and the model FC-Densenet-R revealed a 6.5% MAPE for the testing dataset.

Moreover, we tested VGG19-R to read a stereo vision dataset where the input of the model combined three grayscale images [right, left, depth]. This model estimates the area of Lego blocks even when the blocks are at different distances from the camera. The model achieved MAPE equal to 1.8% for training dataset and 2.4% for the testing dataset.

The main motivation of this study is to solve one of the problems in the aquaculture industry, which is monitoring fish weight inside the tank. Pre-harvest monitoring is a costly and routine task. By using computer vision, the process can be made easier without adding any pressure on the fish. It also reduces the cost of the method by decrease human involving [37].

Furthermore, we reviewed stereo vision, which is one of the cheapest ways to estimate a depth map. We discussed the concept of some models for generating the depth map from stereo vision such as CG-Net, MonoDepth, and Anynet [10, 18, 42, 49]. Anynet for example can estimating the depth with less resources. That makes it excellent to use on embedded chips [49]. In addition, unsupervised models like MonoDepth could be a good approach for new applications, where it is hard to get labeled depth dataset. Yet training those models needs a big dataset content thousands of images. Also, they are sensitive to the stereo vision setup, which means we cannot apply transfer learning to train the model on a new dataset. Except if the new dataset has the same max disparity [9, 26, 49].

4.0.1 Future work

This thesis investigated the conceptual use of deep networks and stereo cameras for estimating the mass of a fish. In order to apply this to commercial aquaculture operations, there are several more challenges to overcome: live fish, unclear images, and large fish density.

As we tested, the depth information should assist the model in estimating the weight of live fish. It is expensive to use the lidar device, and deep learning models like MonoDepth could be a cheap replacement [10]. Deep learning models like this need a big dataset of stereo images.

The second issue is unclear images. Recently, deep learning technology has been used to improve nightlight images, where the images are unclear because of the poor light situation or there is a high level of noise in the image. So, it will be useful to test clearing images from the fish tank by using this type of deep learning model [6].

Last, there are many fish in one tank for more profit. This is a big issue and we need to focus on one fish at a time. We need to also distinguish between the fish and not read the weight of the same fish over and over again.

Reference

[1] *Aquaculture statistics and reports*. Accessed 15. Oct. 2019. Mar. 2019. URL: `http://www.dfo-mpo.gc.ca/aquaculture/stats-eng.html`.

[2] *ARCore - Google Developers | Google Developers*. [Online; accessed 29. Jan. 2020]. Sept. 2019. URL: `https://developers.google.com/ar`.

[3] Gabriel J Brostow et al. "Segmentation and recognition using structure from motion point clouds". In: *European conference on computer vision*. Springer. 2008, pp. 44–57.

[4] Jason Brownlee. *A Gentle Introduction to Computer Vision*. July 2019. URL: `https://machinelearningmastery.com/what-is-computer-vision/`.

[5] Abhishek Chaurasia and Eugenio Culurciello. "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation". In: *CoRR* abs/1707.03718 (2017). arXiv: `1707.03718`. URL: `http://arxiv.org/abs/1707.03718`.

[6] Chen Chen et al. "Learning to See in the Dark". In: *CoRR* abs/1805.01934 (2018). arXiv: `1805.01934`. URL: `http://arxiv.org/abs/1805.01934`.

[7] Francois Chollet. *Deep Learning with Python*. 1st. Greenwich, CT, USA: Manning Publications Co., 2017. ISBN: 9781617294433.

[8] Edward Collier et al. "Progressively growing generative adversarial networks for high resolution semantic segmentation of satellite images". In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 763–769.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 3354–3361.

[10] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 270–279.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[12] Kenji Hata and Silvio Savarese. *CS231A Course Notes 1: Camera Models*. [Online; accessed 9. May 2020]. May 2017. URL: `https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf`.

[13] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[14] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[15] Ramesh Jain et al. *Machine Vision*. McGraw-Hill, 1995. ISBN: 0-07-032018-7.

[16] Simon Jégou et al. "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation". In: *CoRR* abs/1611.09326 (2016). arXiv: `1611.09326`. URL: `http://arxiv.org/abs/1611.09326`.

[17] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding". In: *arXiv preprint arXiv:1511.02680* (2015).

[18] Alex Kendall et al. "End-to-End Learning of Geometry and Context for Deep Stereo Regression". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017). DOI: `10.1109/iccv.2017.17`. URL: `http://dx.doi.org/10.1109/ICCV.2017.17`.

[19] Dmitry A Konovalov et al. "Automatic Weight Estimation of Harvested Fish from Images". In: *arXiv preprint arXiv:1909.02710* (2019).

[20] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[21] Todd Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[22] P. Lottes et al. “Joint Stem Detection and Crop-Weed Classification for Plant-Specific Treatment in Precision Farming”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 8233–8238. DOI: `10.1109/IROS.2018.8593678`.

[23] JR Martinez-de Dios, C Serna, and Anbal Ollero. “Computer vision and robotics techniques in fish farms”. In: *Robotica* 21.3 (2003), pp. 233–243.

[24] *MATLAB Documentation.* https://www.mathworks.com/help/vision/ug/camera-calibration.html. Accessed: 2010-09-20. 2019.

[25] Stefano Mattoccia. “Stereo vision: Algorithms and applications”. In: *University of Bologna* 22 (2011).

[26] N. Mayer et al. “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation”. In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:1512.02134. 2016. URL: `http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16`.

[27] Moritz Menze and Andreas Geiger. “Object scene flow for autonomous vehicles”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3061–3070.

[28] Oecd. *Aquaculture production.* [Online; accessed 15. Oct. 2019]. Oct. 2019. URL: `https://stats.oecd.org/Index.aspx?DataSetCode=FISH_AQUA`.

[29] *OpenAI Five is the first AI to beat the world champions in an esports game after defeating the reigning Dota 2 world champions, OG, at the OpenAI Five Finals on April 13, 2019.* https://openai.com/five/. Accessed: 2010-09-20. 2019.

[30] Luis Perez and Jason Wang. “The effectiveness of data augmentation in image classification using deep learning”. In: *arXiv preprint arXiv:1712.04621* (2017).

[31] Gardner Pinfold. *Socio-economic impact of aquaculture in Canada: 2013 edition.* Accessed 15. Oct. 2019. Feb. 2013. URL: `https://waves-vagues.dfo-mpo.gc.ca/Library/40739016.pdf`.

[32] Simon JD Prince. *Computer vision: models, learning, and inference.* Cambridge University Press, 2012.

[33] *PyTorch at Tesla - Andrej Karpathy, Tesla*. [Online; accessed 6. Jan. 2020]. Nov. 2019. URL: `https://www.youtube.com/watch?v=oBklltKXtDE`.

[34] Md Atiqur Rahman and Yang Wang. "Optimizing intersection-over-union in deep neural networks for image segmentation". In: *International symposium on visual computing*. Springer. 2016, pp. 234–244.

[35] Sebastian Ruder. "Transfer Learning - Machine Learningś Next Frontier". In: *Sebastian Ruder* (Apr. 2019). URL: `https://ruder.io/transfer-learning`.

[36] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 1573-1405. DOI: `10.1007/s11263-015-0816-y`. URL: `https://doi.org/10.1007/s11263-015-0816-y`.

[37] Mohammadmehdi Saberioon et al. "Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues". In: *Reviews in Aquaculture* 9.4 (2017), pp. 369–387.

[38] Daniel Scharstein and Richard Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: *International journal of computer vision* 47.1-3 (2002), pp. 7–42.

[39] E. Shelhamer, J. Long, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017), pp. 640–651. ISSN: 1939-3539. DOI: `10.1109/TPAMI.2016.2572683`.

[40] Kyle Simek. *Dissecting the Camera Matrix, Part 2: The Extrinsic Matrix ←*. [Online; accessed 9. May 2020]. Mar. 2016. URL: `http://ksimek.github.io/2012/08/22/extrinsic`.

[41] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: `1409.1556 [cs.CV]`.

[42] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1007–1015.

[43] Preetham Suresh, Jürgen P Schulze, et al. "Oculus rift with stereo camera for augmented reality medical intubation training". In: *Electronic Imaging* 2017.3 (2017), pp. 5–10.

[44] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[45] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[46] *Unsupervised Monocular Depth Estimation with Left-Right Consistency*. [Online; accessed 14. Jan. 2020]. Dec. 2016. URL: `https://www.youtube.com/watch?v=go3H2gU-Zck&feature=emb_title`.

[47] Guotai Wang et al. "Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks". In: *International MICCAI Brainlesion Workshop*. Springer. 2017, pp. 178–190.

[48] Yan Wang et al. "Anytime Stereo Image Depth Estimation on Mobile Devices". In: *CoRR* abs/1810.11408 (2018). arXiv: `1810.11408`. URL: `http://arxiv.org/abs/1810.11408`.

[49] Yan Wang et al. "Anytime stereo image depth estimation on mobile devices". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 5893–5900.

[50] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[51] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (Nov. 2000), pp. 1330–1334. ISSN: 0162-8828. DOI: `10.1109/34.888718`.