Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering

Faezeh Rezaei

Fish bin picking using machine vision

Master's thesis in Simulation and Visualization
Supervisor: Adam Leon Kleppe
Co-supervisor: Ola Jon Mork
July 2023

NTNU
Norwegian University of
Science and Technology

Faezeh Rezaei

Fish bin picking using machine vision

Master's thesis in Simulation and Visualization
Supervisor: Adam Leon Kleppe
Co-supervisor: Ola Jon Mork
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

The right to food is a crucial human right, ensuring that everyone has access to sufficient and safe nourishment. Fish and fishery products play a vital role in global food security. The fishing industry has been expanding due to rising demand for fish products. To meet future food demands, it is projected that food production needs to double by 2050. Norway, a major seafood exporter, achieved a record-breaking seafood export in 2022.However, the reliance on manual labor in fish processing factories has proven suboptimal and time-consuming. In line with the prevailing trend of automation in various industries, integrating machine learning, machine vision, and robotics holds promise for enhancing work efficiency and product quality within the fish industry. Nonetheless, one challenge with implementing machine learning is the requirement for labeled data, which can be costly and time-intensive to gather and annotate. To address this issue, the utilization of digital twins and 3D models has emerged as a solution to generate synthetic yet precise labeled data. In this thesis, a 3D model of a salmon was employed to generate training data for a machine vision algorithm. This algorithm enables the prediction of the fish's location and orientation on a surface, facilitating the robotic arm's ability to pick and place the fish. A customized variant of the VGG16 model architecture was employed in the experiment. Two different camera positions were assessed, and the trained model utilizing the side camera position exhibited a lower MSE of $8.3x10^{-5}$ for location and $2.5x10^{-3}$ for orientation. Notably, the validation data yielded values of $1.5x10^{-4}$ for location and $5.8x10^{-2}$ for orientation, and for the test the values where $8.4x10^{-3}$ and $1.3x10^{-4}$.

Sammendrag

Retten til mat er en avgjørende menneskerettighet som sikrer at alle har tilgang til tilstrekkelig og trygg ernæring. Fisk og fiskeriprodukter spiller en viktig rolle i global matsikkerhet. Fiskeindustrien har utvidet seg på grunn av økende etterspørsel etter fiskeprodukter. For å imøtekomme fremtidige matbehov er det anslått at matproduksjonen må dobles innen 2050. Norge, en stor eksportør av sjømat, oppnådde en rekordstor sjømateksport i 2022. Imidlertid har avhengigheten av manuelt arbeid i fiskeforedlingsfabrikker vist seg å være suboptimal og tidkrevende. I tråd med den gjeldende trenden med automatisering i ulike industrier, har integrering av maskinlæring, maskinsyn og robotikk potensial til å forbedre arbeidseffektiviteten og produktkvaliteten innen fiskeindustrien. Likevel er en utfordring ved implementering av maskinlæring kravet om merkede data, noe som kan være kostbart og tidkrevende å samle og merke. For å adressere dette problemet har bruken av digitale tvillinger og 3D-modeller dukket opp som en løsning for å generere syntetiske, men presise merkede data. I denne avhandlingen ble det brukt en 3D-modell av en laks for å generere treningsdata for en maskinsynsalgoritme. Denne algoritmen muliggjør forutsigelse av fiskens plassering og orientering på en overflate, noe som letter den robotiserte armen i å plukke opp og plassere fisken. En tilpasset variant av VGG16-modellarkitekturen ble brukt i eksperimentet. To forskjellige kameraposisjoner ble vurdert, og den trente modellen som brukte sidekameraposisjonen viste en lavere MSE (gjennomsnittlig kvadratisk feil) på 8,3x10-5 for plassering og 2,5x10-3 for orientering. Bemerkelsesverdig ga valideringsdata verdiene 1,5x10-4 for plassering og 5,8x10-2 for orientering, og for testen var verdiene 8,4x10-3 og 1,3x10-4.

Preface

I am immensely grateful to everyone who has contributed to the completion of this master's thesis dissertation. This work represents research, analysis, and reflection, and I owe its completion to the support and guidance of many individuals.

Firstly, I express my heartfelt thanks to my supervisors for their unwavering commitment, invaluable expertise, and continuous encouragement. Their guidance and feedback have shaped this dissertation into its final form. Also want to thank the academic staff at Manulab in NTNU Ålesund, who trusted me and gave me the opportunity to work and learn from them. I also acknowledge the academic staff and professors at NTNU for providing a rich academic environment and resources that have greatly contributed to the depth of my research.

Furthermore, I am grateful to my family and friends for their unwavering support throughout this demanding journey. Their encouragement and belief in my abilities have been crucial sources of motivation.

It is my hope that this master's thesis dissertation makes a meaningful contribution to the field of study and inspires future research. I am deeply humbled and honored to have had the opportunity to undertake this research, and I sincerely thank all those who have supported me along the way.

Faezeh Rezaei

Table of Contents

List of Figures

1 Introduction

The right to food is a fundamental human right that ensures everyone has access to sufficient, safe, and nutritious food. Fish and fishery products are important for global food security and provide unique health benefits due to their high nutritional value, including essential nutrients and long-chain omega-3 fatty acids[1].

Fish are an irreplaceable source of long-chain omega-3 fatty acids, which are crucial for optimal brain and neurodevelopment, especially in children. Fish consumption is also known to reduce the risk of Coronary Heart Disease (CHD) mortality in adults due to the omega-3 fatty acids. With a growing global population, the demand for fish is expected to increase, and aquaculture is projected to be the primary source to meet this demand[1].

1.1 The state of fisheries

In 2020, global fisheries and aquaculture production reached a record high of 214 million tons, with aquaculture continuing to grow (figure 1) while capture fisheries declined[2]. Asia was the main producer, accounting for 70% of total production, with China being the largest producer. Inland aquaculture accounted for 37% of total production and was important for food security in Asia and Africa [2].

Despite the growth of aquaculture, the FAO reports that marine fishery resources have continued to decline, with only 64.6% of fishery stocks being within biologically sustainable levels in 2019. However, biologically sustainable stocks accounted for 82.5% of landings of aquatic products. Rebuilding overfished stocks could increase marine capture fisheries production and contribute to the well-being of coastal communities. Successful policies and regulations in sustainable fisheries management need to be replicated and implemented globally.

Inland fisheries, particularly in least developed and developing countries, face challenges in monitoring and managing their resources due to limited resources. The FAO has developed a global threat map for inland fisheries to provide a baseline for further research and investment in inland fisheries management [2].

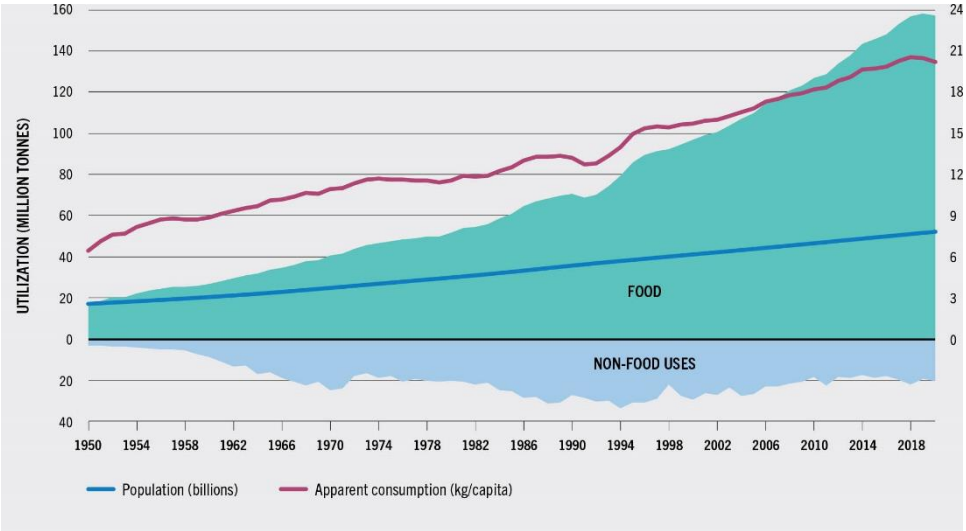


Figure 1: The State of World Fisheries and Aquaculture [2]

FAO has developed a fish price index (FPI) that offers a fresh perspective on global seafood markets and serves as a valuable tool for informing global food policy (figure 2). It complements the FAO's food price information, widely utilized by governments, NGOs, and researchers worldwide.

Figure 2: Global fish price index (FAO)

1.2 The state of fisheries in Norway

The Norwegian seafood industry has a rich history of harvesting, processing, and exporting due to the country's extensive marine resources[3]. In the 1970s, Norway became a world leader in marine aquaculture, and the industry continues to play a significant role in the country's economy. The seafood industry's impact is felt not only in the core industries of the value chain but also in supplier industries and other industries through ripple effects. The aquaculture sector is experiencing faster growth compared to fisheries, making it the dominant part of the Norwegian seafood value chain in terms of value-added and employment since 2010. Currently, having an export value of 1 billion NOK annually, the Aquaculture and Fisheries industry in Norway is significant (figure 3) [3]. Due to the growing global population and in order to meet the upcoming needs, it is anticipated that food production will need to be doubled by 2050 [4].

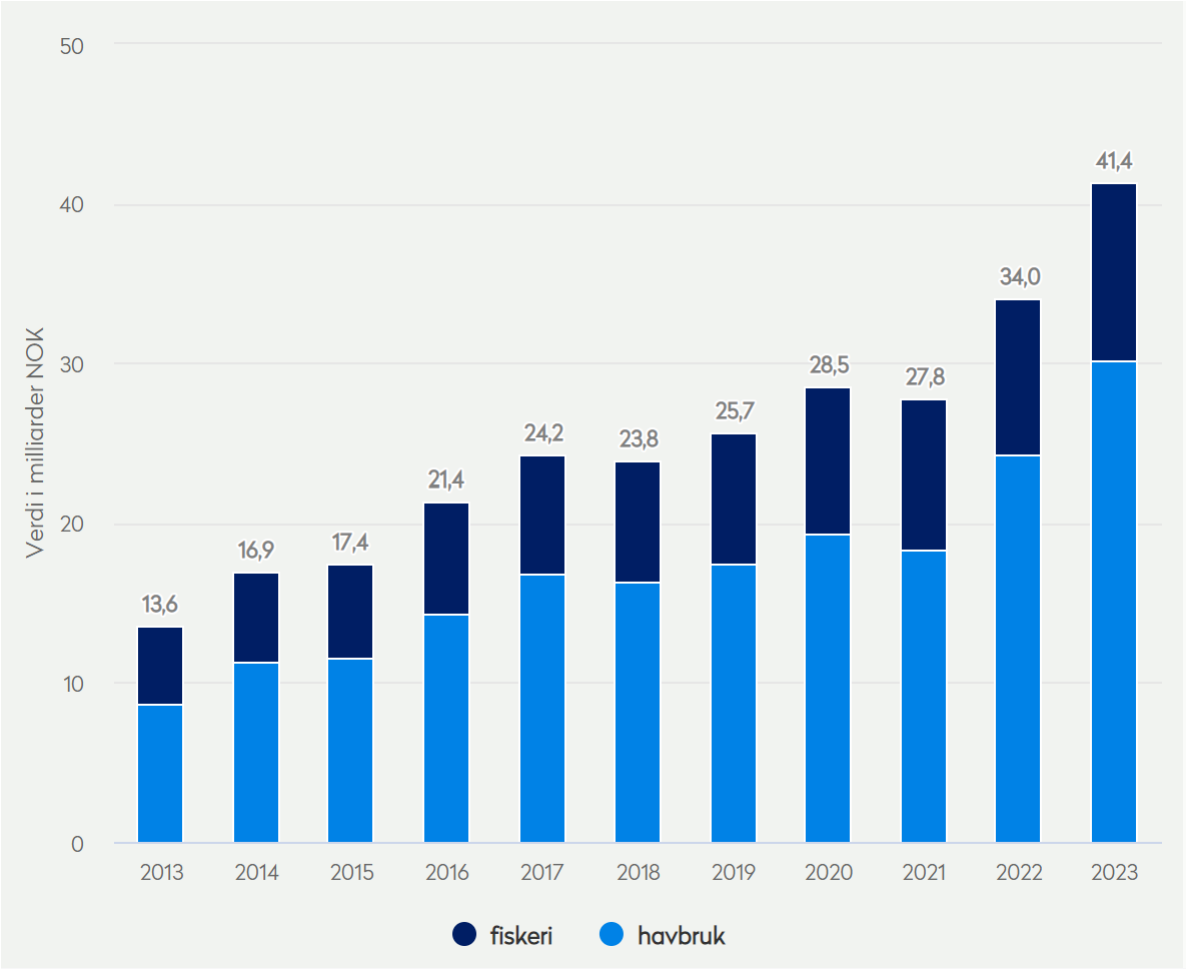


Figure 3: Export of Norwegian seafood in total divided by fisheries and aquaculture [5]

1.3 Manual labor in fish factories

For many years, fishmeal and fish oil factories have used the same old production techniques, which have led to low-quality products that are not valued highly in the market. Therefore, there is a need to enhance and revise these production methods to improve the quality of the products [6].

The seafood industry has historically been labor-intensive, with a significant portion of the world's population being employed in primary production, processing, packaging, and distribution. However, finding workers willing to work in the cold, humid, and often dangerous environment is a significant challenge, particularly with the COVID-19 pandemic and restrictions on social distancing and border travel [7]. To address this, automation within fish production has become prevalent, making it one of the most high-tech automated sectors in the protein industry alongside poultry. Although significant advances have been made in automating fish processing, manual labor is still required for tasks such as trimming fillets and quality inspection, which can be challenging due to variations in fish size, weight, color, physical shape, and state of rigor mortis. Traceability from raw product intake to end consumable product is also necessary in many cases to validate the country of origin and ensure product quality and remaining shelf life. Despite the challenges, automation has enabled a considerable increase in line speed and throughput, with the

most automated fish factories estimated to have improved fish yield and utilization from 60% to 80% in the past decade [8].

1.4 Digital twins

A digital twin is a computer-generated replica that accurately represents a physical entity, such as a wind turbine. To achieve this, sensors are attached to key functional areas of the physical object to collect data on its performance, such as energy production, temperature, and weather conditions [9]. This data is sent to a processing system, which is used to update the digital twin model. With this data-driven model, simulations can be run to analyze performance issues and identify potential improvements. The insights gained from the digital twin can then be applied to the physical object, resulting in increased efficiency.

While simulations and digital twins both employ digital models to replicate a system's processes, digital twins provide a more comprehensive virtual environment for study. In contrast, simulations are typically focused on a single process. Moreover, digital twins benefit from real-time data since object sensors feed relevant data to the system processor and insights are shared back with the physical object [9].

As a result of having access to continually updated data and the added computing power that comes with a virtual environment, digital twins can analyze a wider range of issues from multiple perspectives, giving them greater potential to enhance products and processes than standard simulations [9].

Digital twin technology is finding widespread application in manufacturing, where it has the potential to enable real-time monitoring and feedback on machine performance and production line activity. This technology also facilitates predictive maintenance and improves the reliability of devices by increasing connectivity between them. The use of artificial intelligence (AI) algorithms in conjunction with digital twins can enhance accuracy and enable detailed performance analysis.

In addition to manufacturing, the automotive and construction industries are also employing digital twins for simulation, data analytics, and real-time prediction and monitoring. One common goal across all these industries is the use of real-time simulation enabled by digital twin technology, which allows for continuous learning and monitoring. This technology has the potential to significantly enhance performance and accuracy, making it a valuable asset in various fields [10].

The popularity of Digital Twin (DT) has been steadily increasing, attracting the attention of researchers who have started to focus their studies on this topic. Figure 4 illustrates the exponential growth in the number of publications related to DT found on Scopus and ScienceDirect (limited to English-language articles) from 2011 to 2020. The significant rise in publications primarily occurred from 2016 onwards, indicating a recent surge in research activity [11].

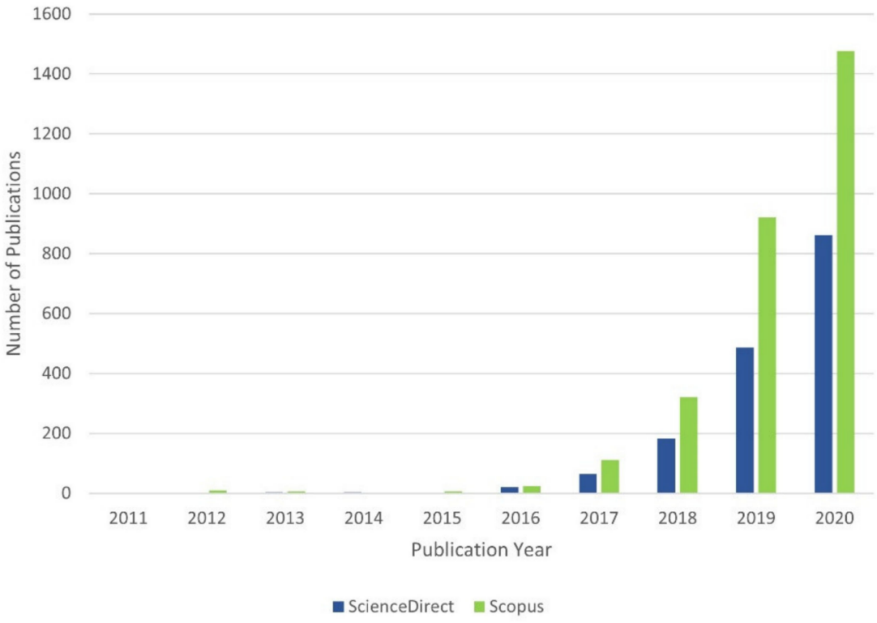


Figure 4: Increase in publications regarding digital twins [11]

1.5 Digital twins in the fishing industry

The requirements of intelligent fish farming, which aims to optimize and automate fish farm management using smart sensors, aquaculture machines, and AI processes through an AIoT (The Artificial Intelligence of Things) system [12]. The primary goals of fish farm management include reducing operation costs, maximizing profits, augmenting fish quality, and optimizing harvest efficiency. However, different environments for each stage of fish farming pose a challenge. The implementation of an AIoT system enables real-time, data-driven decision-making and provides automatic and higher-quality data collection [12].

The utilization of artificial neural networks and machine learning-based digital twins has proved to be effective in identifying damages in structures. Therefore, it is crucial to find an automated and efficient method that can swiftly and precisely detect damage in fishing nets and aquaculture cages, to save time and effort [13].

Digital twins have been used mostly in studies concerning fish farming and involve sensors and AI to optimize fish health, growth, and economic return while reducing risk to the environment. Digital twins can enhance the efficiency of fish farming, maximize production, reduce costs, and optimize the decision-making process [14]. When it comes to fish processing plants, numerous possibilities exist for leveraging digital twins to enhance operational efficiency.

1.6 Fish detection using machine vision

The use of Artificial intelligence is essential for improving product quality and production efficiency in digital aquaculture. The implementation of automatic fish detection is crucial in achieving more precise and intelligent farming. With the widespread availability of modern information technology, computer vision techniques have emerged as a powerful tool for automatic fish detection. However, fish detection using computer vision models faces numerous challenges, including poor lighting, low contrast, high noise, fish deformation, occlusion, and dynamic backgrounds [15]. When using machine vision for

detection based on the images of the real world, it is best that the image is clear with good lighting and the object can be separated from the background and the surroundings, so that the machine vision algorithm can have good results.

The same applies for the dead fish in the processing plants, though in the plants there are less challenges since the fish are usually on a conveyor belt or tables, and the lighting can be adjusted for better results.

Machine vision technology has been created and applied in numerous fields. It can capture real-time images, examine them, and use the output to address specific queries. Additionally, machine vision is being increasingly utilized in aquaculture, including for tasks such as monitoring the health of aquatic organisms, identifying diseases, detecting behavior patterns, and determining species. The measurement of fish size using machine vision has been developed and adopted widely due to its excellent efficiency [16].

1.7 Combining digital twins and machine learning

The manufacturing industry requires factories to be more flexible and adaptable to meet the demands of high product variety, uncertain demand, and strict delivery deadlines. The combination of Digital Twin (DT) and Artificial Intelligence (AI) technologies is seen as promising for addressing the emerging demands of customized production. DT models consist of the real world, virtual world, and the connections of information that connect the virtual with the real world. AI technology can support complex decision-making and business processes in production systems[17] .

There are several applications of DT in manufacturing that aim to reduce costs and improve performance. These include DT-driven frameworks to optimize planning and commissioning of human-based production processes, process parameter optimization, and physical-virtual convergence.

On the other hand, AI tools are also being used in manufacturing. Reinforcement learning has been utilized to enhance scheduling methods, and unsupervised learning algorithms, such as autoencoders, can extract features from input data without needing label information. Convolutional Neural Networks (CNNs) have been used for automated defect identification in surfaces and visual recognition of parts. However, two challenges faced by supervised ANN (Artificial neural network) are the availability of proper quality and quantity datasets and labelling the datasets for training.

Therefore, digital Twin (DT) technology can support smart manufacturing by combining the physical and virtual environments. Although Artificial Intelligence (AI) and Machine Learning (ML) are promising in manufacturing, they require significant amounts of quality training data, which can be costly and time-consuming to label. DT models can help expedite the ML training process by generating appropriate training data and automatically labeling it through simulation tools, reducing user input. These synthetic datasets can be supplemented with real-world information and validated.

The adoption and use of digital manufacturing and advanced technologies such as DT and AI are necessary for factories to become more flexible and adaptable to meet the emerging demands of customized production. The combination of DT and AI can address complex decision-making and business processes in production systems [17].

1.8 Thesis outline

The objective of this thesis is to employ a machine vision algorithm for identifying the position and orientation of fish placed on a surface, such as a table or conveyor belt. This prediction serves the purpose of guiding a robotic arm in the task of picking up and relocating the fish from the surface. Considering the laborious nature of gathering labeled data for machine vision, a digital twin of fish in Unity3D was used to generate synthetic yet dependable data, to train the machine vision algorithm.

The thesis structure is outlined as follows:

Chapter 1: Introduction, this chapter explores the significance of fish and the fish industry, highlighting their importance in various contexts. It delves into the growing trend of automation and the emergence of digital twins, elucidating their potential to enhance the fish industry.

Chapter 2: Theoretical background, provides a explanation of essential concepts in machine learning and machine vision algorithms. It explains the principles behind neural networks, deep learning, and convolutional neural networks (CNNs). Additionally, it presents a modified convolutional network specifically tailored to suit the objectives of this thesis.

Chapter 3: Methods, which details the approaches employed to generate the fish model and gather labeled data from the digital twin, utilizing the tools offered by Unity 3D. Furthermore, the chapter provides an explanation of the model architecture and the training methodology employed in this study.

Chapter 4: Results, this chapter presents the outcomes of the model training process, encompassing key metrics such as loss and accuracy for the various trained models.

Chapter 5: Discussion, It provides a comprehensive analysis of the performance of each model, highlighting their strengths and weaknesses based on the achieved results. The chapter serves to offer a clear and detailed understanding of the effectiveness and efficiency of the trained models in addressing the objectives of this thesis.

2 Theoretical background

2.1 Fish digital twin platform

A 3D fish simulator was created using Unity3D, at Manulab in NTNU Ålesund. Its purpose was to replicate the movements of a dead fish when dropped or transported on a conveyor belt. A range of fish species such as pollock, bass, and salmon were precisely modeled. Real fish data was collected and employed to generate fish of various sizes, ensuring realistic proportions in the simulation [18].

2.1.1 Fish dimentions

A dataset containing 61 fish data was collected. The dataset was completed on three different dates, each of which came from a different place. Two different kinds of fish were included in the dataset. The first one was Norwegian Sei (Pollock) with a total number of 12, and the second kind was Salmon with a total number of 49. Simple measurements of the dimensions of the fish (Length (cm), Height (cm), Width (cm)) plus its Weight were taken. Additionally, for simulation purposes and to ensure adequate accuracy and precision, the fish were cut into five different slices as shown in the figure 5, and each part was weighed separately. The parts included Head weight, Front body weight, Back body weight, Tail weight, and Tail fin weight [18].

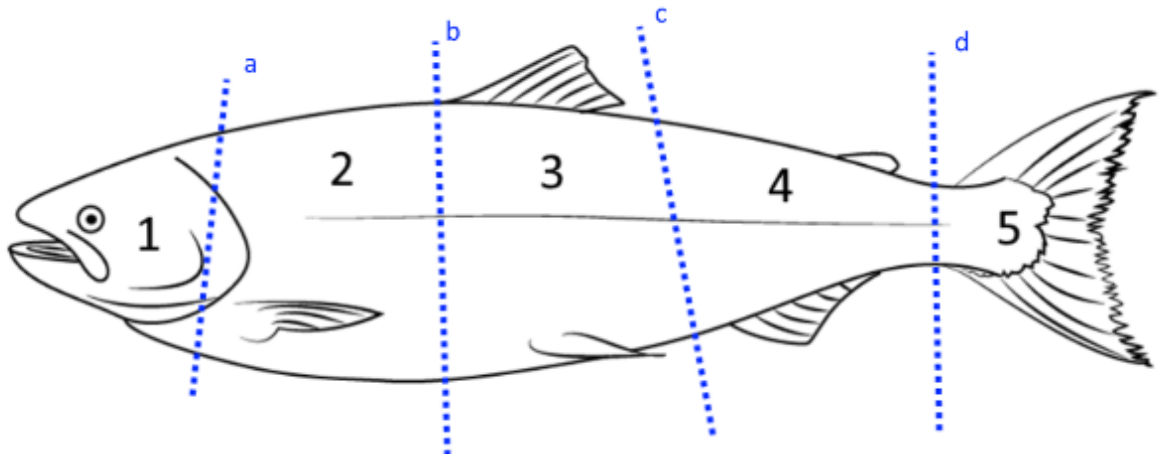


Figure 5: Positions of the slices [18]

The length of the fish was considered as the base, and the rest of the dimensions were calculated relative to the length, as they were proportional to it. The relationship between the length of the fish and its width, height, and weight was obtained using polynomial regression (figures 6 and 7) [18].

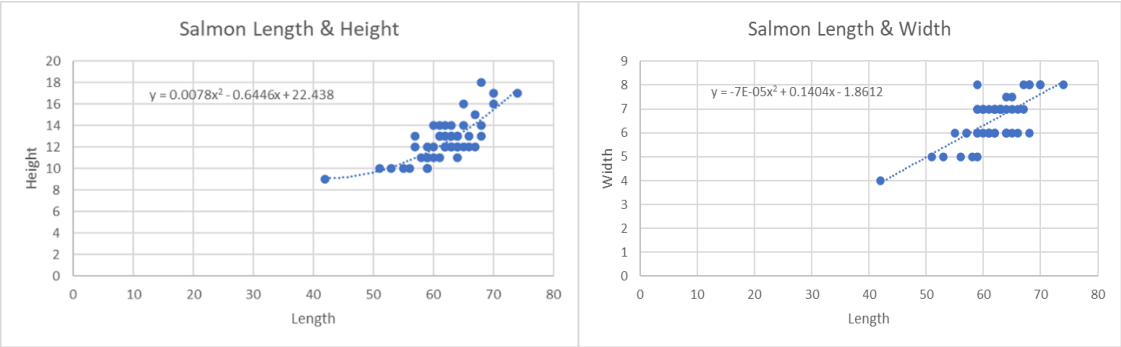


Figure 6: Relation between length, width, and height [18].

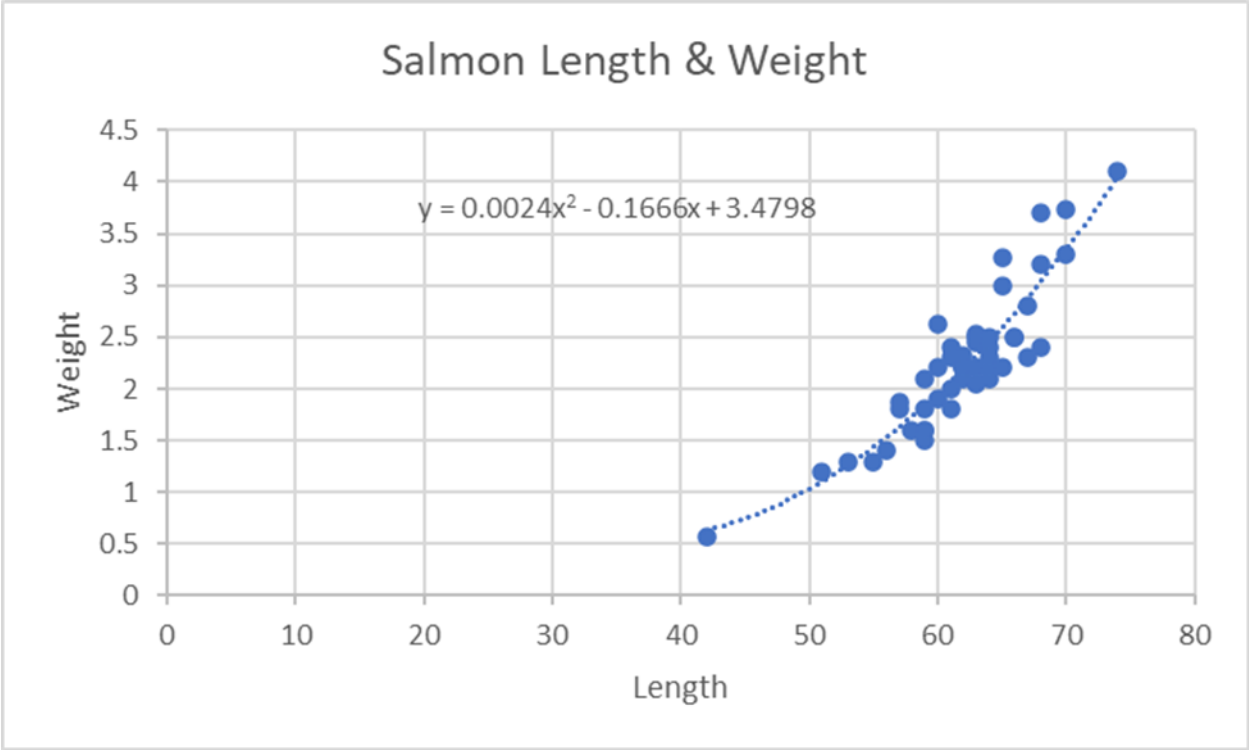


Figure 7: Relation between length and weight [18].

2.1.2 Fish structure and flexibility

To simulate realistic reactions to forces, the movement of the skeletal structure needs to be constrained. To achieve this, constraints are applied to the "Character Joint" component of the model in Unity3D. The *Character Joint* component has many configurable values and is quite complex. The most important parts of it are the *Connected Body*, both *Axis*, and all the limits. These together control the dynamics of the joint, in figure 8 the limit of bending left and right for the joint between head and body can be seen.

The *Axis* controls the twist axis of the joint, meaning its internal rotational axis. For a fish model, it must be set to the axis that is protruding from the sides of the model, since the model was positioned with Z as its forward axis. Left and right became the X axis. The *Swing Axis* controls the direction of joint movement. It must be set to the forward axis. In this case, the Z direction. By setting it to the Z axis, the joints are allowed to move about the 2D plane XY in accordance with the limits of *swing 1* and *swing 2 [18]*.

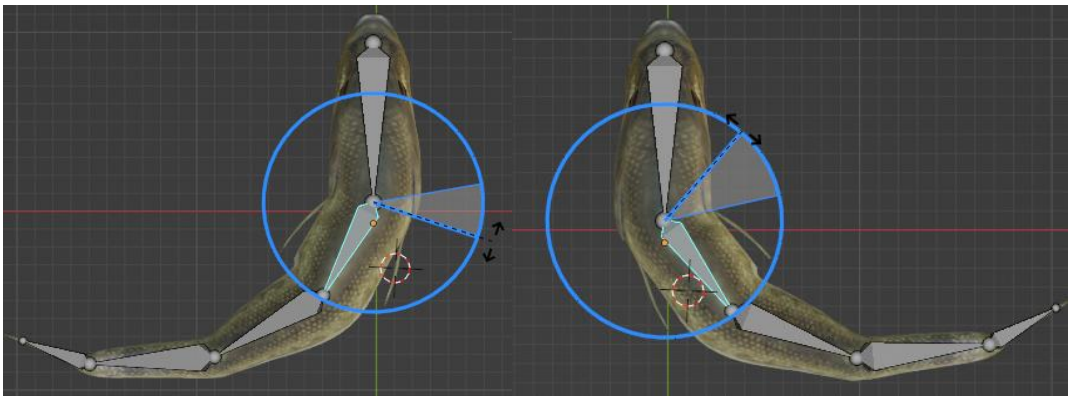


Figure 8: Bending left and right limit [18]

2.1.3 The fish simulator

By implementing the mentioned methods, a simulator of realistic fish was developed. In the simulator fish models were generated that move on a conveyor belt (figure 9). The product was primarily employed to aid in optimizing the design of conveyor belts in fish processing plants, with the aim of enhancing the efficiency of fish separation. It was tested and compered to actual fish behavior in the processing factory and yielded good results[18]. Additionally, the modeled fish and the simulator can be used to generate synthetic fish data for training machine vision algorithms. Since the data and the model proved to be good, in this thesis the salmon fish model and the parameters were used in that regard.

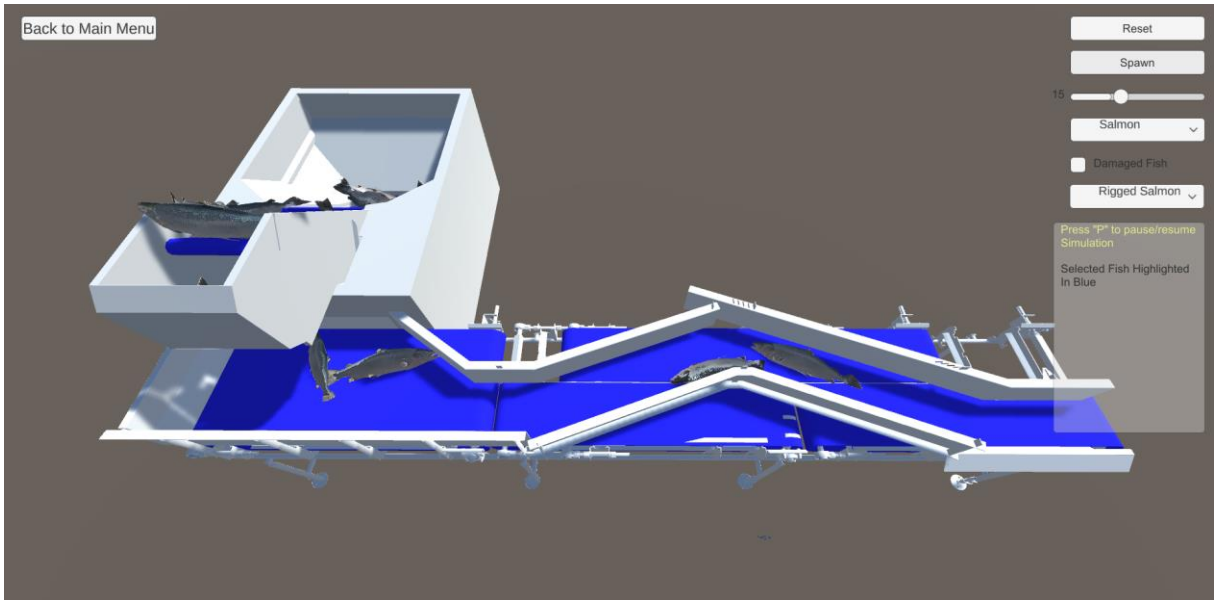


Figure 9: The developed simulator [18]

2.2 Machine learning/Machine vision algorithms

2.2.1 Machine learning

Machine learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models capable of learning from data to make predictions or decisions without explicit programming [19]. It has emerged as a powerful tool for solving complex problems and extracting valuable insights from vast amounts of information.

In traditional programming, humans define explicit rules and instructions for computers to perform specific tasks, meaning the logic is formulated and turned into code. However, in machine learning, computers learn from examples and data patterns, enabling them to generalize and make accurate predictions on new, unseen data. Therefore, the logic of the problem that is being solved is not pre-defined.

The fundamental idea behind machine learning is to build mathematical models that capture patterns and relationships within the data. These models are then trained using various algorithms to optimize their performance. The training process involves feeding the model with labeled data (that have been tagged with one or more labels identifying certain properties or characteristics), where each input is associated with a corresponding output. By analyzing this labeled data, the model learns to recognize patterns and make predictions or decisions based on new, unseen inputs.

There are several types of machine learning algorithms, with the most common being supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning involves training a model using labeled data, where the desired output is known. The model learns to map inputs to outputs by generalizing from the provided examples. For instance, in image classification, the model learns to recognize and classify images into predefined categories, such as cats and dogs. Supervised learning is widely used in various domains, including natural language processing, computer vision, and fraud detection.

Unsupervised learning deals with unlabeled data, where the model aims to find hidden patterns or structures within the data. The goal is to discover inherent relationships and groupings without predefined labels. Clustering algorithms, such as K-means and hierarchical clustering, are commonly used in unsupervised learning. This type of learning is valuable for tasks like customer segmentation, anomaly detection, and recommendation systems.

Reinforcement learning involves training an agent to interact with an environment and learn optimal actions to maximize cumulative rewards. The agent receives feedback in the form of rewards or penalties based on its actions, allowing it to learn through trial and error. Reinforcement learning has been successful in applications such as playing games (such as chess, checkers and even video games lately), robotics (including robot navigation, object recognition and manipulation), and autonomous vehicle control.

Machine learning algorithms employ various mathematical techniques to process and analyze data effectively. Decision trees, support vector machines, neural networks, and Bayesian networks are among the many algorithms used in machine learning.

- Decision trees are hierarchical structures that make decisions based on a sequence of questions or conditions. Each node in the tree represents a question or condition,

leading to subsequent nodes until a final decision is reached. Decision trees are interpretable and widely used for classification and regression tasks.

- Support vector machines (SVM) are algorithms used for both classification and regression. They find an optimal hyperplane that separates different classes or predicts continuous values. SVMs are effective when dealing with high-dimensional data and are known for their robustness.
- Neural networks, inspired by the human brain, consist of interconnected nodes (neurons) organized in layers. They are capable of learning complex patterns and relationships in data.
- Bayesian networks model probabilistic relationships among variables using directed acyclic graphs. They are valuable for modeling uncertainty and making probabilistic inferences. Bayesian networks have applications in medical diagnosis, gene expression analysis, and risk assessment.

Machine learning enables computers to learn from data and make predictions or decisions without explicit programming. It has revolutionized various fields, ranging from healthcare and finance to transportation and entertainment. By employing different algorithms and techniques, machine learning has the potential to unlock valuable insights and drive innovation across industries [20].

2.2.2 Artificial neural networks

Artificial neural networks (ANNs) consist of node layers, including input, hidden, and output layers. Each node can be imagined as an independent linear regression model, consisting of input data, weights, a bias (or threshold), and an output. The mathematical representation can be depicted as follows.

$$f(x) = \sum_{i=1}^{n} x_i w_i + bias$$

In which n is the number of inputs, x_i is the input, w_i is the weight for each input. Therefore, the inputs are multiplied by their corresponding weights and added together. Subsequently, the output is fed into an activation function, which determines the final output. Activation function helps to introduce non-linear relationships and complex mappings between inputs and outputs, enabling neural networks to learn and represent more intricate patterns in data.

There are several types of activation functions commonly used in neural networks, some of the popular ones including:

- Sigmoid: This function maps the input to a value between 0 and 1, which can be interpreted as a probability. It is often used in the output layer of binary classification problems.
$$f(x) = \frac{1}{1 + e^{-x}}$$
- Hyperbolic tangent (Tanh): Like the sigmoid function, it maps the input to a value between -1 and 1. It is commonly used in hidden layers of neural networks.
$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
- Rectified Linear Unit (ReLU): It returns the input directly if it is positive, and zero otherwise. ReLU is one of the most widely used activation[21] functions due to its simplicity and effectiveness in deep neural networks.

$$f(x) = \begin{cases} 0 \; if \; x \leq 0 \\ x \; if \; x \geq 0 \end{cases}$$

If this output surpasses a specified threshold, the node becomes "activated" and transmits data to the subsequent layer in the network. Consequently, the output of one node becomes the input for the succeeding node. The layer between the input and output layer is called a hidden layer. This sequential data transmission from one layer to another characterizes the neural network as a feedforward network, subsequently resulting in the predicted outputs (figure 10).

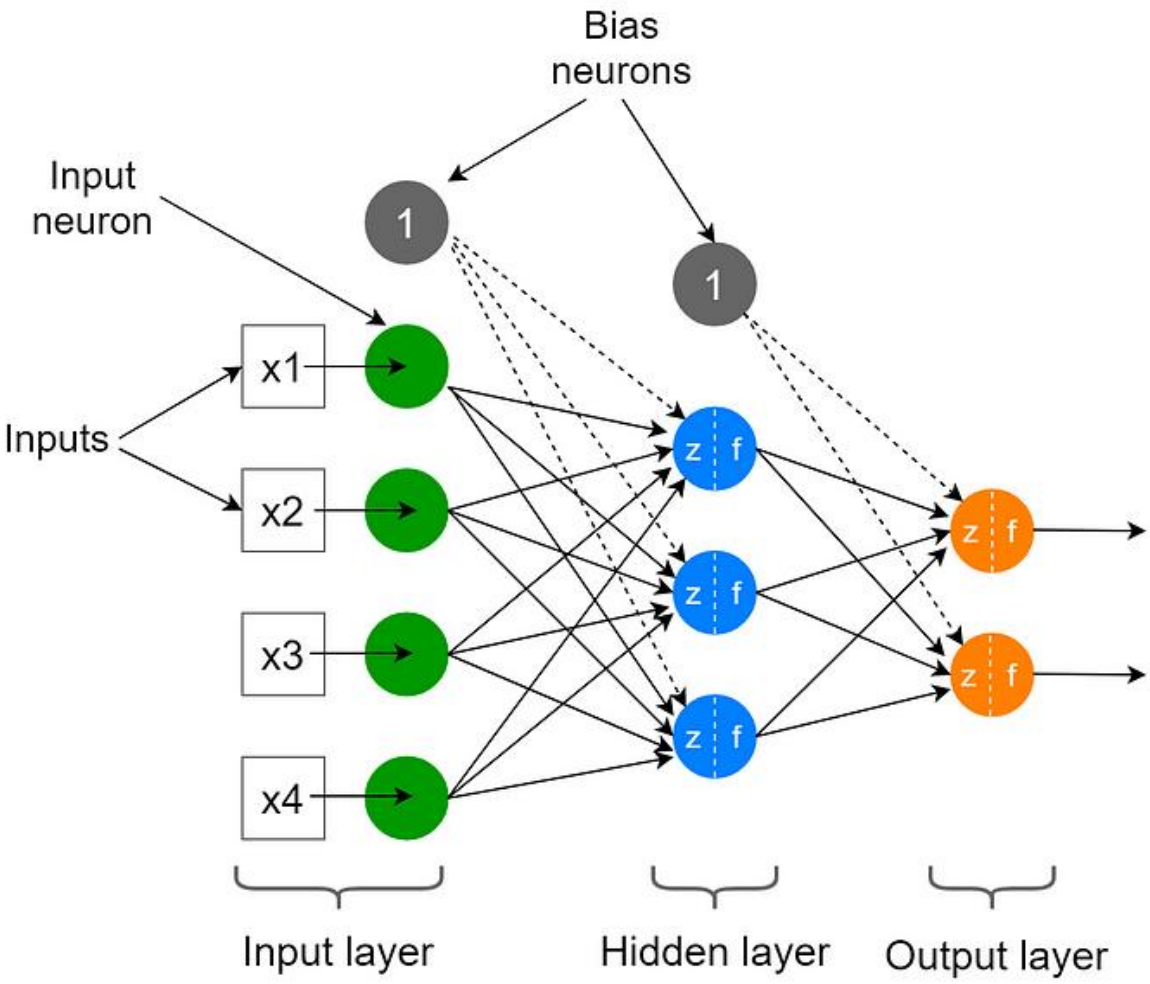


Figure 10: Structure of an artificial neural network [22]

A cost function, also known as a loss function or an objective function, is a mathematical function that measures the discrepancy between the predicted output of a machine learning model and the true output (or target) values in the training data. The cost function quantifies the error or the degree of mismatch between the predicted values and the actual values. Different machine learning tasks and algorithms may require different types of cost functions. The choice of a cost function depends on the nature of the problem being solved and the desired behavior of the model. Some common types of cost functions include Mean square error (MSE) which is common in regression problems, Binary Cross-Entropy and Categorical Cross-Entropy which are used in classification problems.

$$C = MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

In which y is the true value and $\hat{y}$ is the predicted value.

Backpropagation, or backward propagation of errors, is used to update the model's parameters based on the computed loss. The gradients of the loss with respect to the parameters are calculated, and the weights in the network are adjusted in a way that minimizes the loss. This iterative process of forward and backward propagation allows the network to learn from the data and improve its predictions over time. Since we want to update the weights with respect to the loss, we can calculate the derivate of the loss with the following formula:

$$\frac{\partial C}{\partial W_{ik}^{(i)}}$$

Which is the derivative of loos based on the weight of the network going from node j in layer i - 1 to node k in layer i

When we want to back propagate the error we use chain rule from the start of the network down to the output (loss).

2.2.3 Deep learning

Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers to learn and make predictions or decisions. It has gained significant attention and achieved remarkable success in various domains, including computer vision, natural language processing, and speech recognition [23].

At its core, deep learning leverages neural networks, which are computational models inspired by the structure and function of the human brain. Neural networks consist of interconnected nodes called neurons, organized in layers. Each neuron receives input signals, performs a computation, and passes the output to the next layer of neurons.

Deep learning networks differ from traditional neural networks in that they have multiple hidden layers between the input and output layers instead of only one or no hidden layer in traditional neural networks. These hidden layers allow the network to learn complex representations of the input data, enabling it to capture intricate patterns and relationships. The additional layers provide hierarchical abstractions, with each layer learning progressively more complex features.

Deep learning has seen tremendous advancements due to the availability of large datasets and the computational power provided by modern hardware, such as graphics processing units (GPUs). Additionally, the implementation of optimization algorithms, such as stochastic gradient descent and its variants, has contributed to the deep learning being more popular and reliable [23].

Convolutional Neural Networks (CNNs) are a popular type of deep neural network widely used in computer vision tasks. CNNs are designed to automatically learn and extract hierarchical features from images, enabling tasks such as image classification, object detection, and image segmentation (CNN will be elaborated in chapter 2.2.4)

Recurrent Neural Networks (RNNs) are another type of deep neural network that can handle sequential data, such as text or speech. RNNs have memory cells that allow them to capture dependencies and patterns over time, making them suitable for tasks like language modeling, machine translation, and speech recognition. RNNs excel in tasks that involve sequential information processing, as they can retain contextual information across different time steps.

In recent years, advancements in deep learning have also been driven by architectures such as Generative Adversarial Networks (GANs) and Transformers [23]. GANs are used for generating new data instances, such as realistic images, while Transformers have improved natural language processing tasks, including machine translation, text generation, and language understanding. Transformers, with their self-attention mechanism, can efficiently capture long-range dependencies in sequential data, making them highly effective for tasks involving large contextual understanding.

2.2.4 Convolutional neural networks

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed for processing data with a grid-like structure, such as images or time-series data. They have revolutionized the field of computer vision and achieved remarkable success in various applications, including image classification, object detection, and image segmentation.

At the core of CNNs is the convolution operation. Unlike traditional neural networks that use general matrix multiplication, CNNs employ convolution to extract meaningful features from the input data. Convolution involves sliding a small window called a filter or kernel over the input data and computing the dot product between the filter and the local receptive field at each position. This process captures local patterns and spatial relationships, allowing the network to learn hierarchical representations of the data.

One of the key advantages of CNNs is their ability to detect and extract relevant features (such as horizontal or vertical lines, shapes, etc) from the input data. Using multiple convolutional layers, the network learns to detect simple features like edges and textures in the initial layers and progressively learns more complex and abstract features in the deeper layers. This hierarchical feature learning makes CNNs highly effective at capturing intricate patterns and improving classification performance.

CNN architectures typically consist of several layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform the convolution operation, applying various filters to extract features. Pooling layers reduce the spatial dimensions of the feature maps, reducing computation and providing translation invariance. Fully connected layers at the end of the network combine the extracted features and make predictions.

2.2.4.1 Convolutional layer

The convolutional layer in deep learning performs a dot product between a learnable matrix called a kernel and a restricted part of the input image known as the receptive field. The kernel is smaller in size spatially but extends in depth, matching the number of channels in the image (e.g., RGB channels). During the forward pass, the kernel slides across the image's height and width, generating an activation map that represents the response of the kernel at each spatial position. The size of the kernel's sliding movement is called the stride [24].

If we have an input image of size W x W x D and Dout number of kernels, with a spatial size of F, a stride of S, and padding of P, the output volume's size can be calculated using the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

The convolution process can be seen in figure 11.

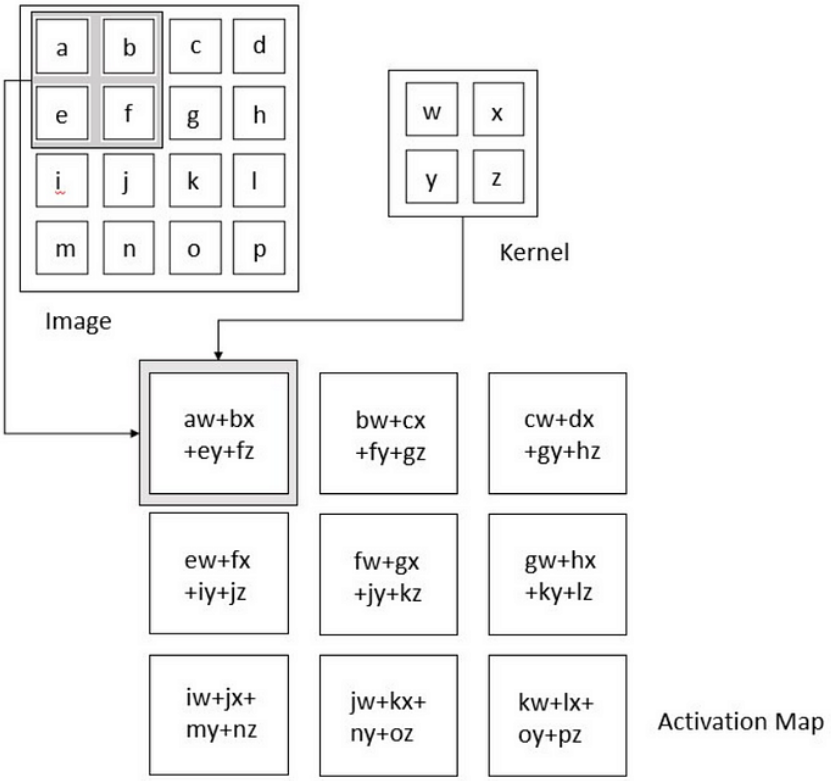


Figure 11: Convolution[23]

2.2.4.2 Pooling layer

The pooling layer in deep learning replaces certain locations in the network's output with a summary statistic of nearby outputs. This helps reduce the spatial size of the representation, resulting in less computation and weights required. The pooling operation is applied individually to each slice of the representation. There are different pooling functions available, such as average pooling, L2 norm pooling, weighted average pooling based on distance, but the most popular one is max pooling. Max pooling selects the maximum value from the neighborhood as the representative output, while average pooling selected the average value from the neighborhood [24]. The two methods are demonstrated in figure 12.

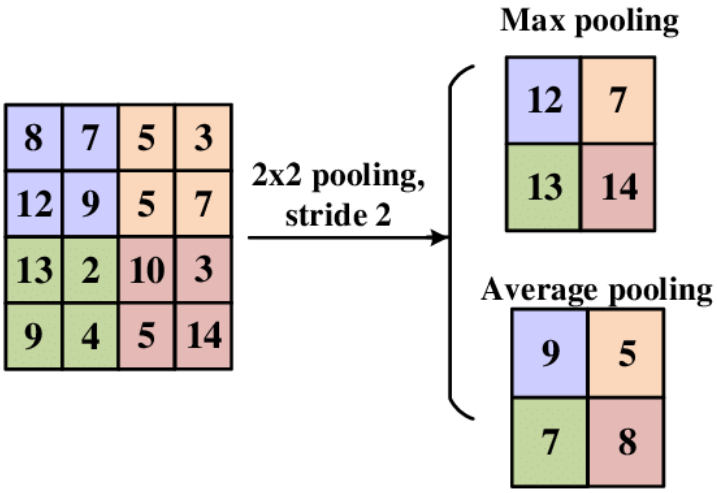


Figure 12: Average and max pooling [25]

If we have an activation map of size W x W x D, a pooling kernel with spatial size F, and stride S, the size of the output volume can be calculated using the provided formula.

$$W_{out} = \frac{W - F}{S} + 1$$

Pooling also provides some translation invariance, meaning that an object can be recognized regardless of its location within the frame.

2.2.4.3 CNN implemetations

One widely used CNN architecture is the LeNet-5 architecture proposed by Yann LeCun et al. in 1998 [26]. LeNet-5 consists of multiple convolutional and pooling layers followed by fully connected layers. It was designed for handwritten digit recognition and laid the foundation for modern CNNs. Since then, numerous CNN architectures have been developed, such as AlexNet [27], VGGNet [28] , GoogLeNet [29], and ResNet [30], each with its own innovations and improvements.

The success of CNNs can be attributed to their ability to capture local patterns and spatial dependencies, mimicking the way the visual cortex processes information. The hierarchical nature of CNNs allows them to learn increasingly abstract representations, enabling them to recognize complex visual patterns and objects.

In addition to image classification, CNNs have been extended to various computer vision tasks. Object detection algorithms, such as the region-based CNNs (R-CNN) family, utilize CNNs to detect and localize objects within images. CNNs have also been applied to image segmentation tasks, where the goal is to assign a class label to each pixel in an image, enabling detailed understanding and analysis of the image content.

Furthermore, CNNs have found applications beyond computer vision. They have been successfully employed in natural language processing tasks, such as text classification and sentiment analysis. By treating textual data as a sequence of one-dimensional inputs, CNNs can learn meaningful representations and capture local dependencies in the text.

Training CNNs typically involves large, labeled datasets and sophisticated optimization techniques. The backpropagation algorithm, combined with gradient descent optimization, is commonly used to update the network's parameters, and minimize the loss function. To

prevent overfitting, regularization techniques such as dropout and weight decay are employed.

Moreover, pretraining and transfer learning techniques have been effective in leveraging pre-trained CNN models on large datasets and transferring knowledge to new tasks with limited labeled data. This approach has significantly reduced the amount of data required to train CNNs and improved performance on various tasks.

In conclusion, Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision and achieved outstanding results in image processing tasks. By employing the convolution operation, hierarchical feature learning, and sophisticated architectures, CNNs can automatically learn meaningful representations from grid-like data. With their ability to capture local patterns, spatial dependencies, and abstract visual features, CNNs continue to drive advancements in computer vision and find applications in diverse domains [23].

2.2.5 VGG16 model

The used model is a modified version of VGG-16 architecture. VGG16 refers to the VGG model, also called VGGNet. It is a convolution neural network (CNN) model supporting 16 layers as shown in figure 13. K. Simonyan and A. Zisserman from Oxford University proposed this model [28].

2.2.5.1 Model structure

The VGG16 model demonstrates impressive performance, reaching a remarkable test accuracy of 92.7% on the extensive ImageNet dataset. ImageNet comprises over 14 million training images, covering a wide range of 1000 object classes. VGG16 stands out as one of the prominent models recognized in the ILSVRC-2014 competition.

Building upon the advancements of AlexNet, VGG16 introduces notable improvements by replacing the use of large filters with a series of smaller 3×3 filters. In comparison, AlexNet employs an 11-size kernel for the initial convolutional layer and a 5-size kernel for the second layer. To train the VGG model, the researchers dedicated several weeks of training utilizing NVIDIA Titan Black GPUs, which contributed to its exceptional performance.

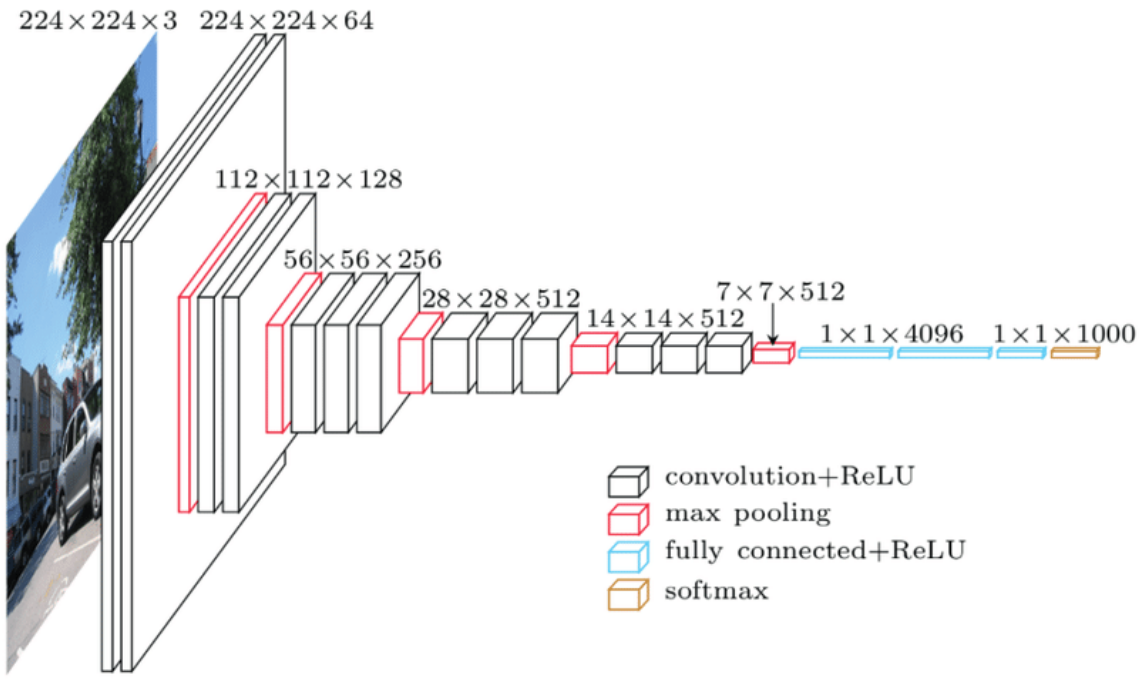


Figure 13: VGG16 Architecture[31]

VGG16, as indicated by its name, is a deep neural network consisting of 16 layers. With a staggering total of 138 million parameters, VGG16 stands out as a remarkably large network even by today's standards. However, the beauty of the VGGNet16 architecture lies in its simplicity. The VGGNet architecture effectively incorporates essential features of convolutional neural networks. A VGG network utilizes small convolution filters, and in the case of VGG16, it comprises three fully connected layers and 13 convolutional layers.

Here's a brief overview of the VGG architecture:

- Input: VGGNet takes a 224×224 image as input. During the ImageNet competition, the creators of the model maintained a consistent image input size by cropping a 224×224 section from the center of each image.
- Convolutional layers: VGG employs convolutional filters with the smallest receptive field of 3×3. Additionally, a 1×1 convolution filter is utilized as a linear transformation for the input.
- ReLU activation: The next component is the Rectified Linear Unit Activation Function (ReLU), which was a major innovation in AlexNet for reducing training time. ReLU acts as a linear function, producing an output that matches positive inputs and yields zero for negative inputs. VGG uses a convolution stride of 1 pixel to preserve the spatial resolution after convolution.
- Hidden layers: All hidden layers in the VGG network utilize ReLU activation instead of Local Response Normalization like AlexNet. This approach eliminates unnecessary increases in training time and memory consumption, with little impact on overall accuracy.
- Pooling layers: Following multiple convolutional layers, pooling layers are employed to reduce the dimensionality and number of parameters in the feature maps generated by each convolution step. Pooling plays a crucial role, especially with the significant growth in the number of filters from 64 to 128, 256, and eventually 512 in the final layers.

- Fully connected layers: VGGNet includes three fully connected layers. The first two layers consist of 4096 channels each, while the third layer comprises 1000 channels, representing one channel for each class [32].

3 Methods

3.1 Modified VGG16 for object detection

In their study titled "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," Tobin et al. employed a customized variant of the VGG16 model. The object detector employed by the researchers is parametrized with a deep convolutional neural network. Specifically, a modified version of the VGG-16 architecture (depicted in Figure 14) is utilized. This architecture was chosen due to its strong performance in various computer vision tasks and the availability of pretrained weights.

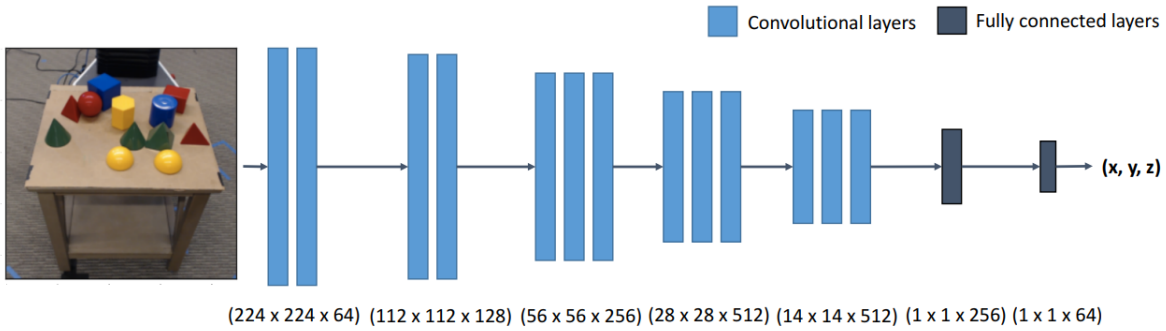


Figure 14: Modified VGG16 for object detection[33]

The convolutional layers of the standard VGG architecture are retained, while the fully connected layers are modified to smaller sizes of 256 and 64, without utilizing dropout. For most of the experiments, pretrained weights obtained from ImageNet are used to initialize the convolutional layers, as it is hypothesized to be crucial for achieving successful transfer. Interestingly, random weight initialization is found to work equally well in most cases.

To train the detector, stochastic gradient descent is employed, utilizing the L2 loss between the estimated object positions from the network and the true object positions. The Adam optimizer is utilized for this purpose. The name "Adam" stands for "Adaptive Moment Estimation," which refers to the adaptive learning rates used by the optimizer. The algorithm keeps track of past gradients and squared gradients to adjust the learning rate for each parameter individually. This allows it to dynamically adapt the learning rate during training, making it well-suited for models with large and sparse datasets or when dealing with noisy gradients. It was discovered that using a learning rate of approximately 1e−4 (compared to the standard 1e−3 for Adam) improved convergence and helped avoid a common local optimum where all objects are mapped to the center of the table [33].

3.1.1 The modified implemented model

In the context of object detection using simulation, the modified model (shown in Figure 14) is designed with only one output parameter. However, considering the requirement to predict both the position and rotation of the object, the model can be slightly adjusted to incorporate the desired outputs, as depicted in Figure 15.

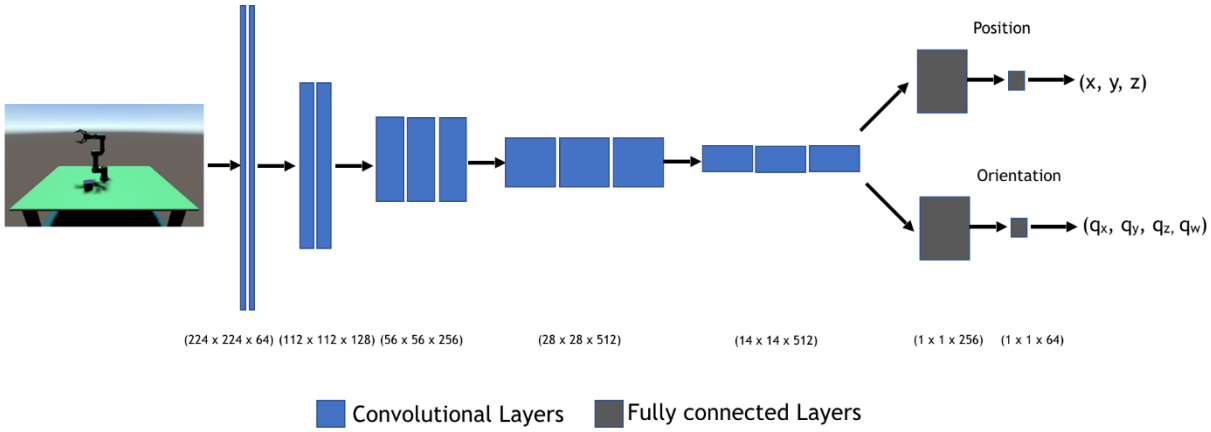


Figure 15: Modified VGG16 for multiple outputs [34]

In previous applications, the method was employed for detection and pick-and-place tasks involving a single cube. However, in this project, the scope is expanded to include a fish model. The objective is to estimate the position and orientation of the fish using the model, allowing the robotic arm to successfully pick and place the fish. This capability holds potential for real-world scenarios, such as fish processing plants, where such automation can be beneficial.

3.2 Data generation for machine vision

This project focuses on training a model to predict an object's pose using images and ground-truth labels. The trained model can then estimate the pose of unseen objects in real-time. The project utilizes Unity Computer Vision [34] to generate synthetic data, providing an efficient solution for machine learning data requirements. In this project automatically labeled data is generated in Unity, allowing the training of a machine learning model. Subsequently, the model is deployed in a Unity simulation with a UR3 robotic arm using the Robot Operating System (ROS), enabling pick-and-place tasks with an object of unknown pose.

In order to achieve this, we have the 3D model of the fish and the environment in unity (our fish digital twin), with a virtual camera and a light. The goal is for the virtual camera to capture images and label the fish, so the data can be used for training. This is done using the perception camera component. To diversify the dataset and simulate real-world scenarios, we introduce randomization to the environment. This involves varying the fish's position, rotation, and size, as well as adjusting lighting conditions and other relevant parameters. To facilitate this randomization process, we utilize a simulation scenario equipped with custom randomizers.

In the following sections, each of these components will be elaborated upon in detail.

3.2.1 Perception camera

Data collection utilizes the Unity perception package, with the virtual camera in Unity 3D configured as a perception camera. To enable object labeling, a labeling component is attached to the desired game object (in this case, the fish model). This facilitates object detection and labeling by the perception camera (figure 16).

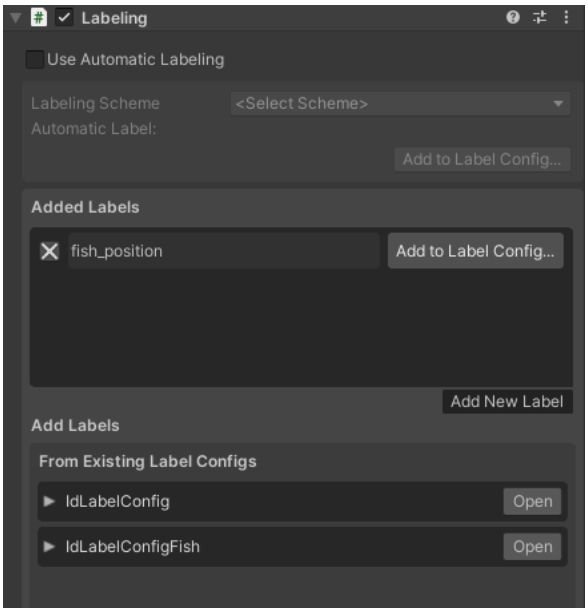


Figure 16: Labeling component for the fish object

Next, a perception component is incorporated into the camera (figure 17), enabling it to capture an image of the scene and label the objects within. Various types of labelers, such

as bounding box 2D labeler, bounding box 3D labeler, and object count labeler, are available, each serving a specific purpose.

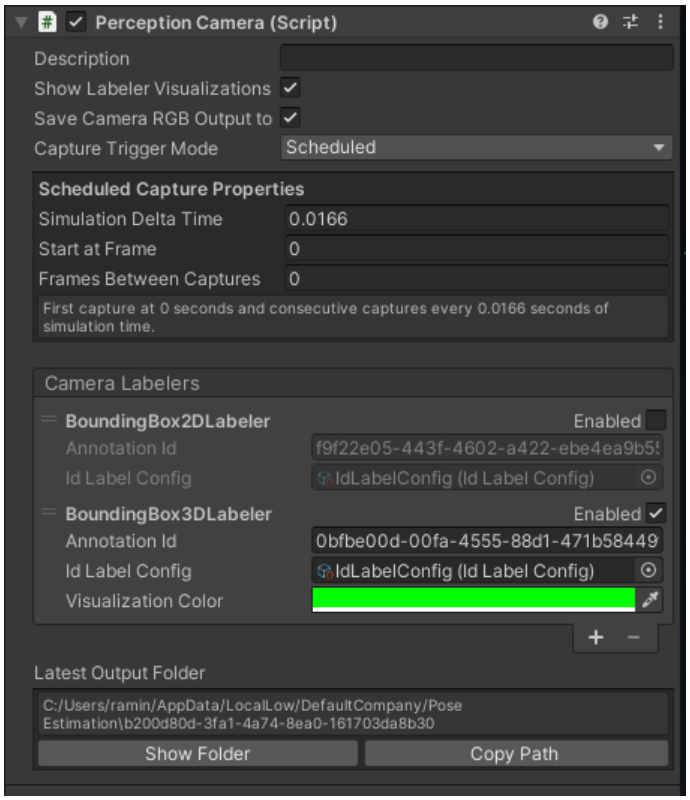


Figure 17: Perception camera component

Given that we have a 3D object, and our objective is to detect both its location and orientation, the most suitable labeler to use is the bounding box 3D labeler. This labeler captures the object by enclosing it within a 3D box as shown by the green box in figure 18.

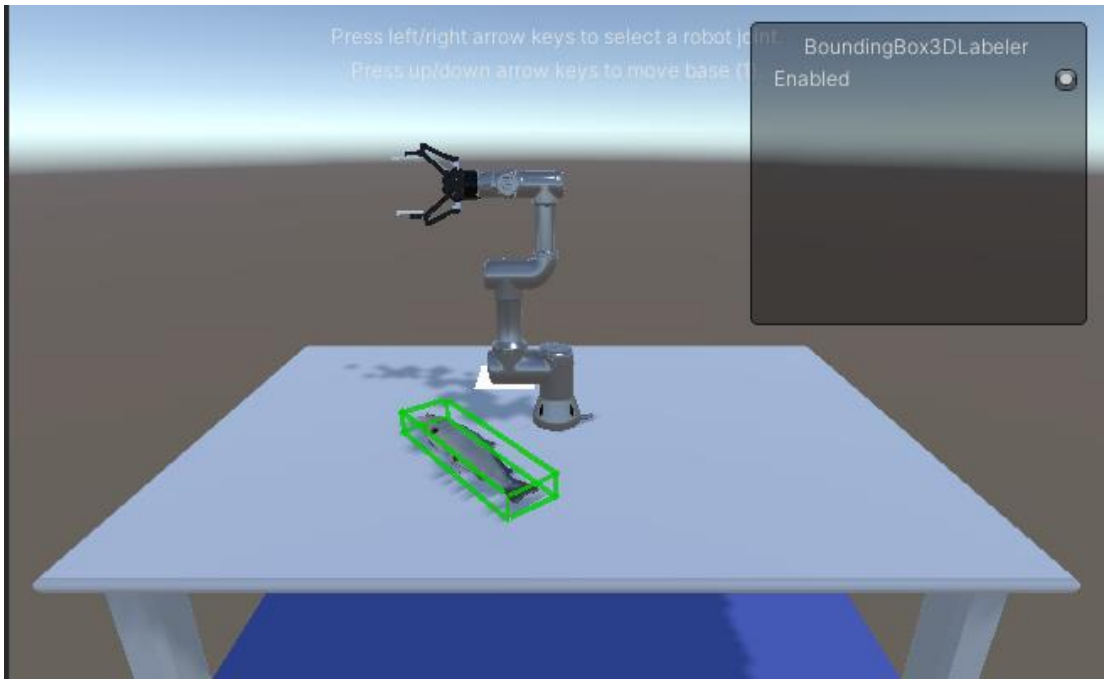


Figure 18: Bounding box 3D labeler

3.2.2 Simulation scenario

Scenario is a game object in unity that controls the execution flow of the simulation by coordinating all the randomizers that are added to it in figure 19 the scenario component and the added randomizers can be seen.

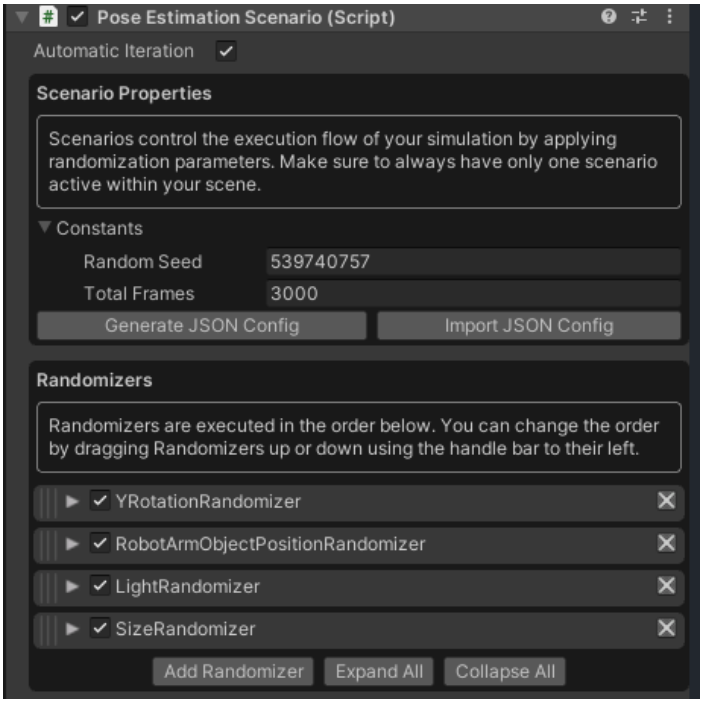


Figure 19: Pose estimation scenario component.

The perception camera component is responsible for generating various scenarios, with each scenario corresponding to a captured image. The size of the dataset can be controlled by specifying the number of frames in the total frames section. By configuring this setting, the component can be programmed to automatically execute the scenario and capture all the necessary pictures for the dataset.

In the randomizers section, various types of randomizers can be incorporated, each specifying the modifications to be made in each frame. These randomizers are implemented as scripts that alter different parameters of the objects within the scene. To ensure proper manipulation, each relevant object is associated with a tag script that enables the randomizer to identify and manipulate the desired object. The following randomizers have been included in this context:

- Rotation randomizer: This randomizer introduces rotational variations to the fish along the X and Y-axis, Y-axis representing the upward direction in Unity. By setting the minimum and maximum rotation values within the randomizer, the fish's orientation can be randomized within the specified range as can bee seen in the figure 21. The X-axis randomizer is either -90 or +90 that corresponds to the left or right side of the fish on the table. Therefore, our dataset has both sides of the fish. The Z-axis is set to 0 since the fish is on a flat surface and there is no need for this rotation (figure 20).

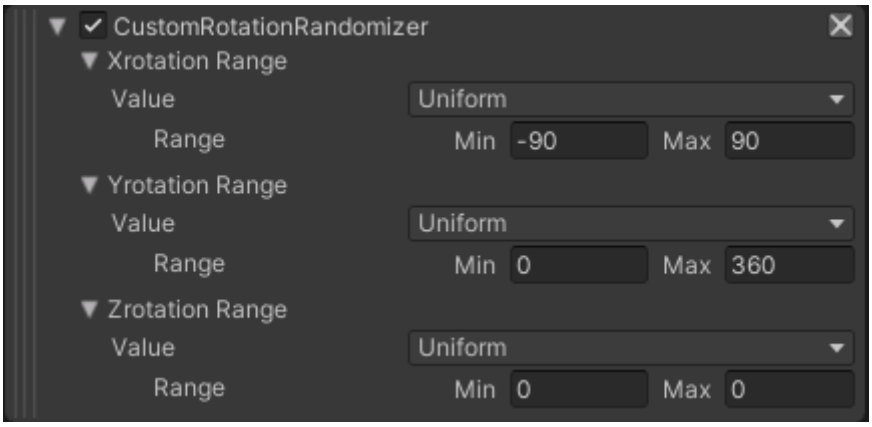


Figure 20: Rotation randomizer

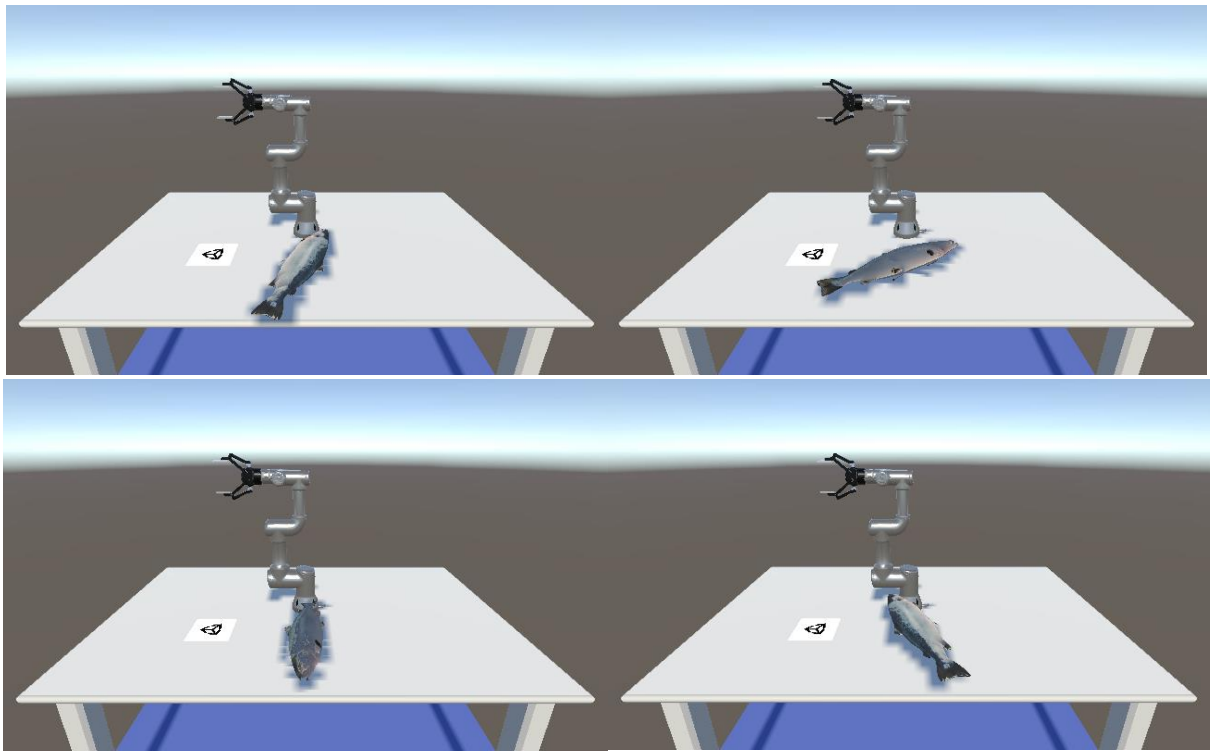


Figure 21: Results of the rotation randomizer

- Robot arm object position randomizer: This randomizer is designed to randomize the position of the fish in a manner that ensures accessibility for the robotic arm. It considers the base location of the robotic arm and allows the specification of minimum and maximum reachability parameters within the randomizer. By configuring these parameters, the randomizer ensures that the fish's randomized position remains within a range that can be reached by the robotic arm (figure 22).

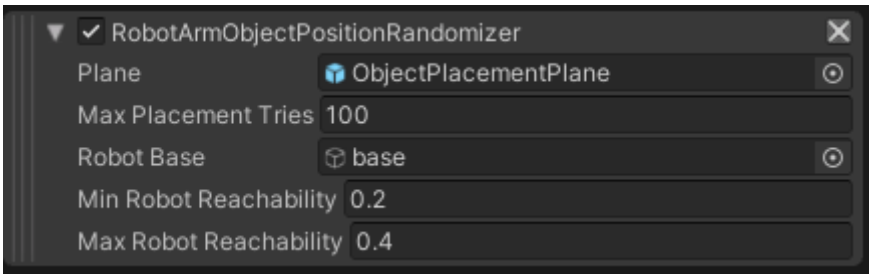


Figure 22: Robot arm object position randomizer

As a result, the fish will be positioned in different places on the table as seen in figure 23, since it is only the position randomizer, there is no rotation.

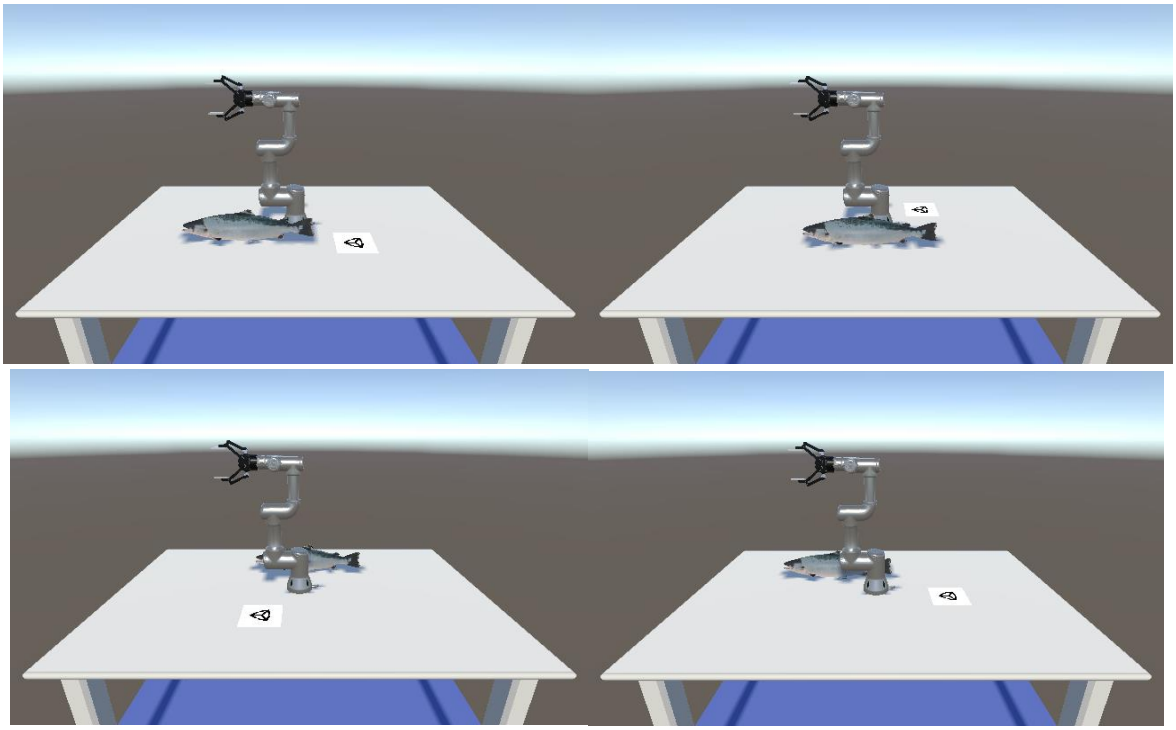


Figure 23: Location randomizer results

- Light randomizer: The light randomizer is responsible for introducing random variations to different parameters of the light source, including intensity, rotation, and color. By incorporating this randomizer, the model becomes adaptable to different lighting conditions. It allows for the creation of diverse scenarios by randomizing the characteristics of the light source, thus enhancing the model's

ability to handle varying lighting conditions (figure 24). The results can be seen in figure 25.

Figure 24: Light randomizer

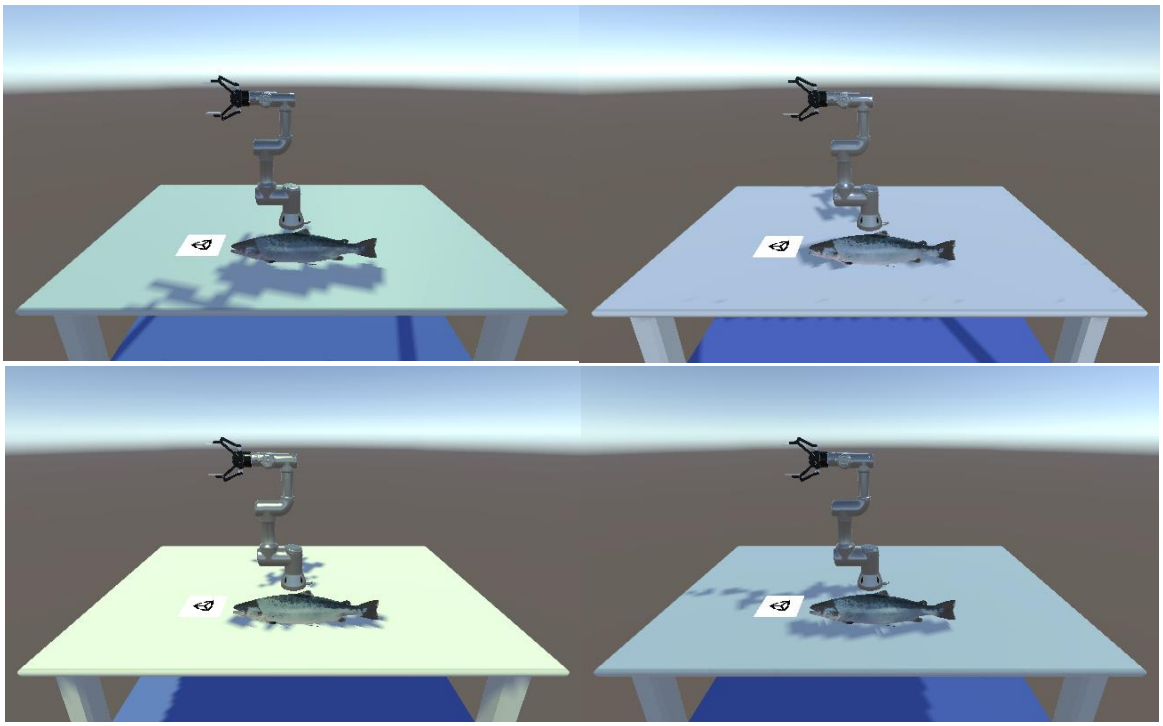


Figure 25: Light randomizer results

- Size randomizer: To avoid uniformity in fish sizes, the size randomizer is implemented to introduce variations in the dimensions of the generated fish. This randomizer samples data from a Gaussian distribution, utilizing average and

standard deviation values derived from measurements of real fish. By incorporating this randomizer (figure 26), the digital fish exhibit diverse sizes (figure 27), resembling the natural distribution observed in real-world fish populations.

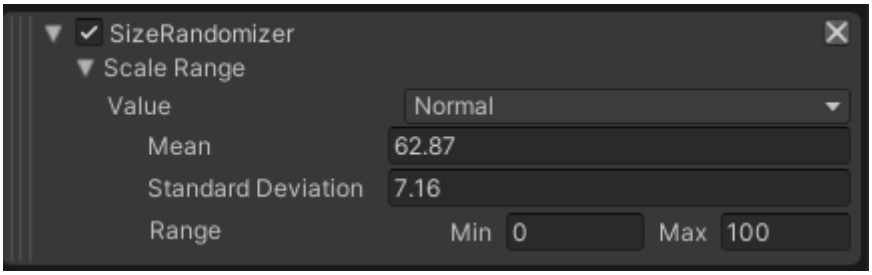


Figure 26: Size randomizer

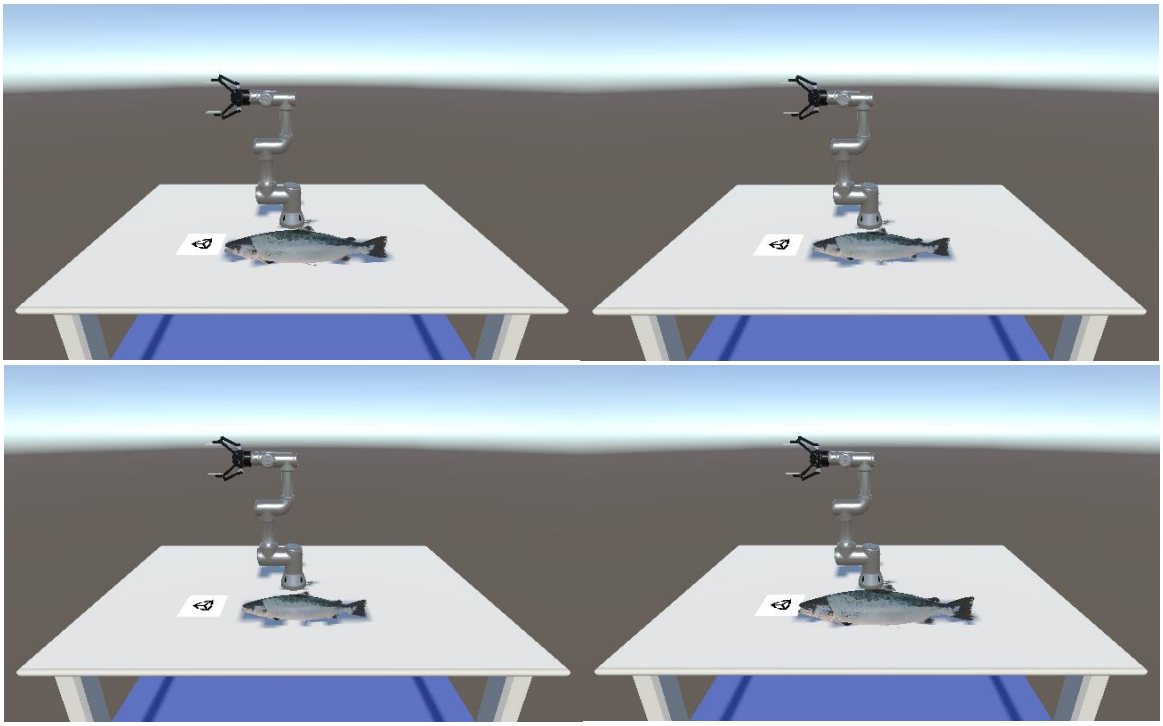


Figure 27: Size randomizer results

In all randomizers, except for the light randomizer, the fish object is associated with a tag component. And for the light randomizer it is associated with the directional light in the scene.

With the setup complete, data collection for model training can be started. The scenario can be executed multiple times, generating different frames each time. This capability allows for the generation of distinct datasets for training, validation, and testing purposes. By running the scenario with varying frames, diverse sets of data can be generated to ensure robustness and generalization in the model's training process. A sample of the generated dataset can be seen in figure 28.

4 Results

The training process involved a dataset of 30,000 images for training and 3,000 images for validation. The Adam optimizer was utilized, and the model was trained using mini batches with the batch size of 20, for a total of 120 epochs. The Adam randomizer was used, and the values were 0.0001 for alpha, 0.9 for beta1 and 0.999 for beta2.

To analyze and compare the camera's orientation and viewport, two distinct camera positions were selected: one with a side view and the other with a top view. The subsequent sections present the outcomes of training for these two models.

4.1 Side view

The sample of generated synthetic data can be seen in the figure 28 below.

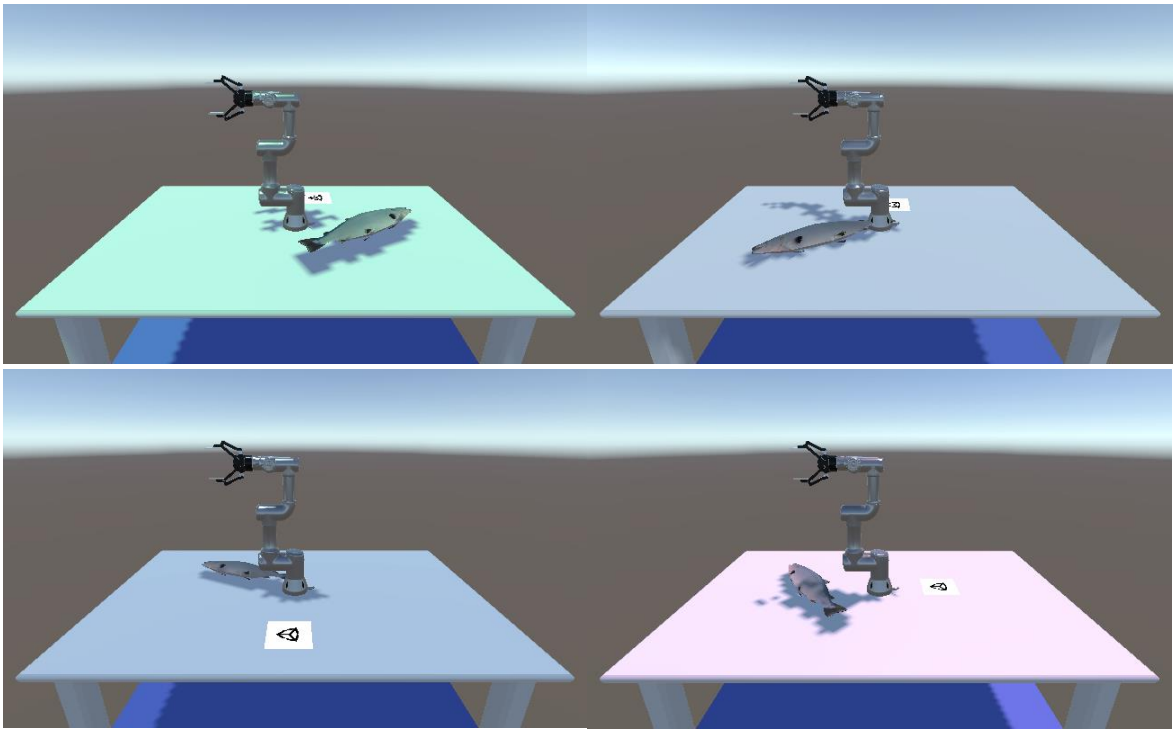


Figure 28: Sample of generated data incorporating all the randomizers (side view)

The change of loss by epoch for orientation and location for training and test data can be seen in the figures below (figures 29-32).

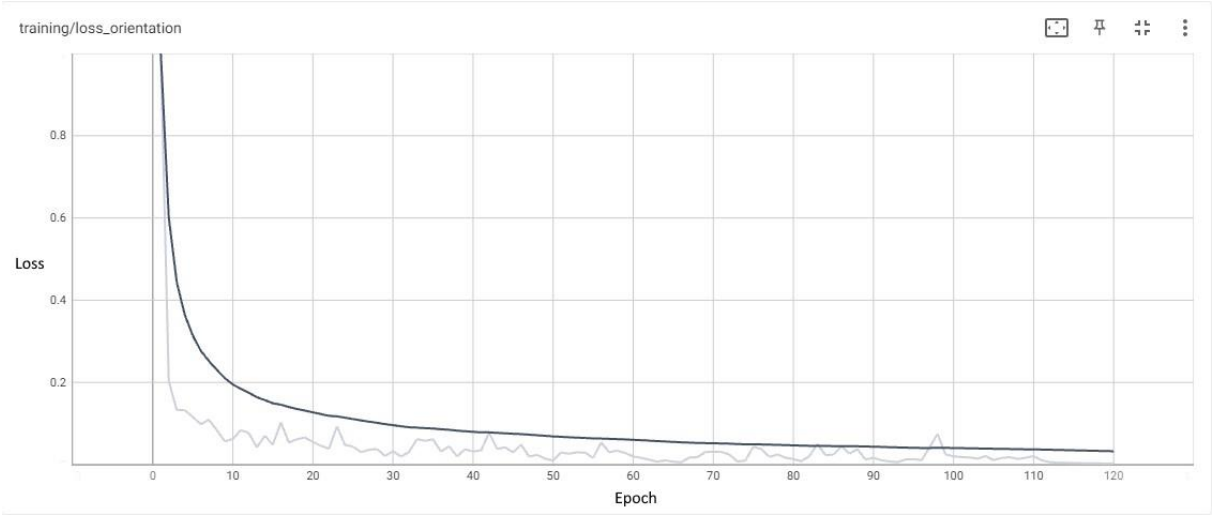


Figure 29: Orientation loss for training (side view)

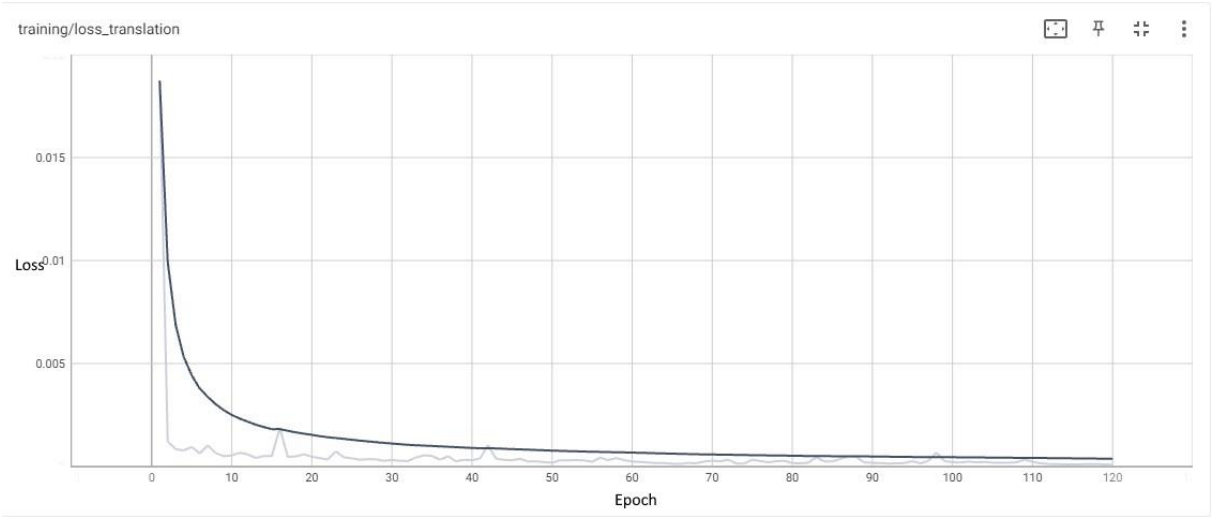


Figure 30: Location loss for training (side view)

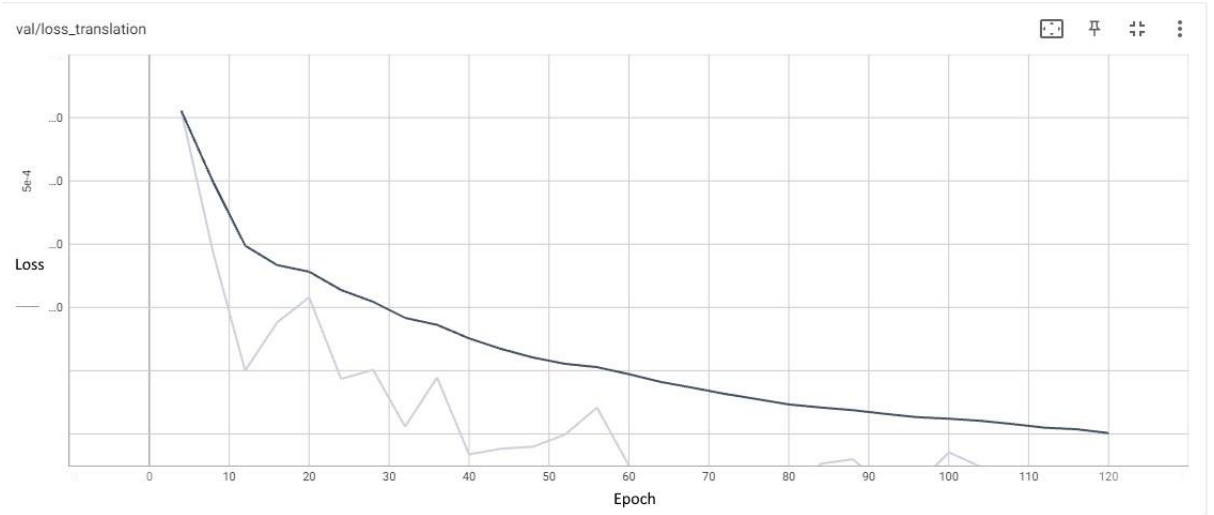


Figure 31: Location loss for validation (side view)

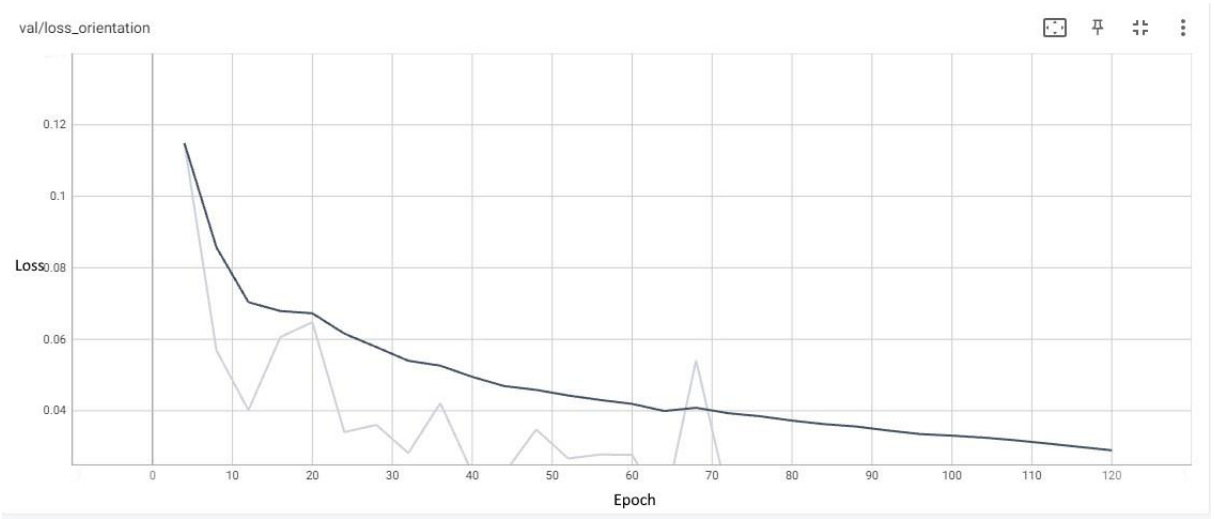


Figure 32: Orientation loss for validation (side view)

Here is the result of implementing the model to detect the fish and robot trying to pick it up (figure 33)

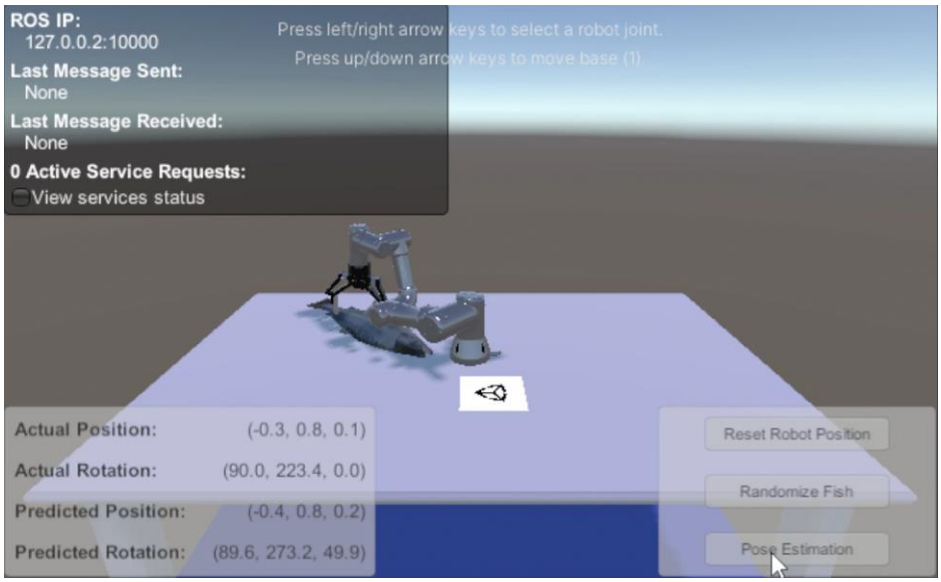


Figure 33: Robot arm detecting and trying to pick up the fish (side view)

4.2 Top View

A sample of generated fish data with the top view camera can be observed in figure 34.

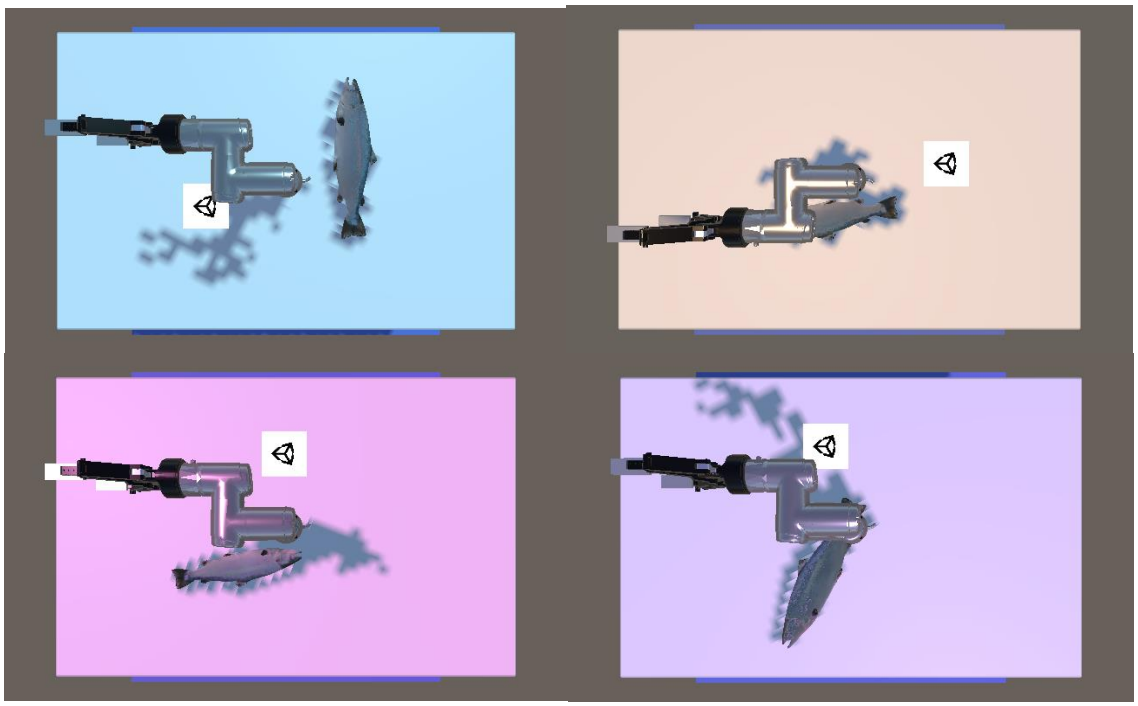


Figure 34: Sample of generated fish data incorporating all the randomizers (top view)

The change of loss by epoch for orientation and location for training and test data can be seen in the figures below (figures 35-38).

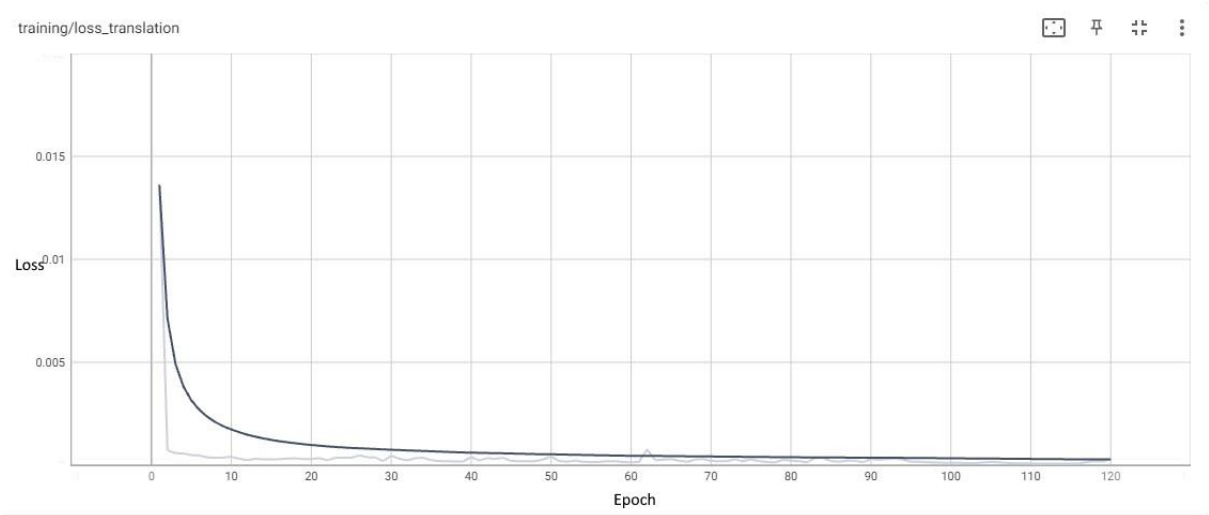


Figure 35: Location loss for training (top view)

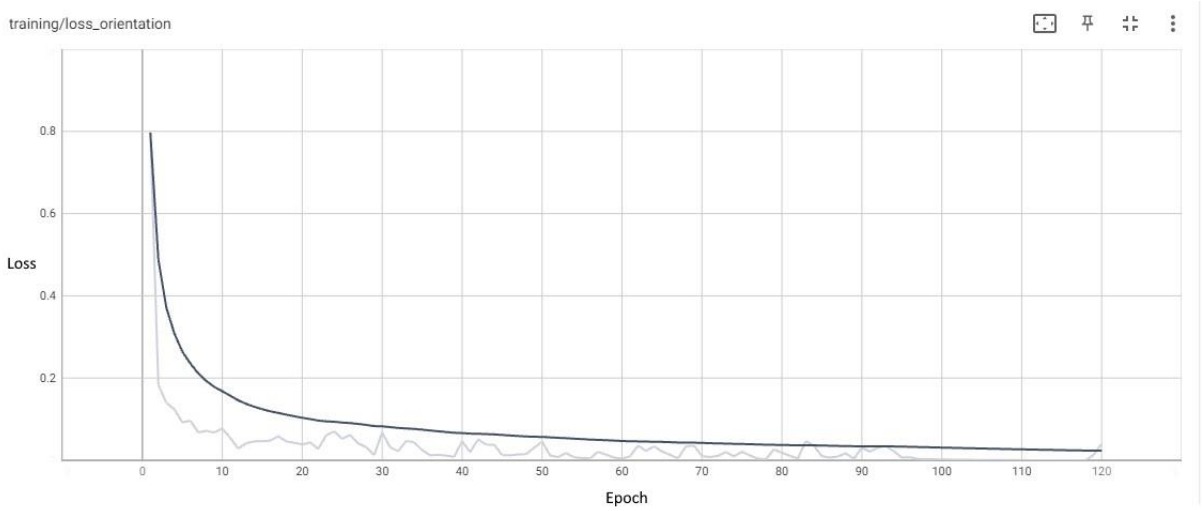


Figure 36: Orientation loss for training (top view)

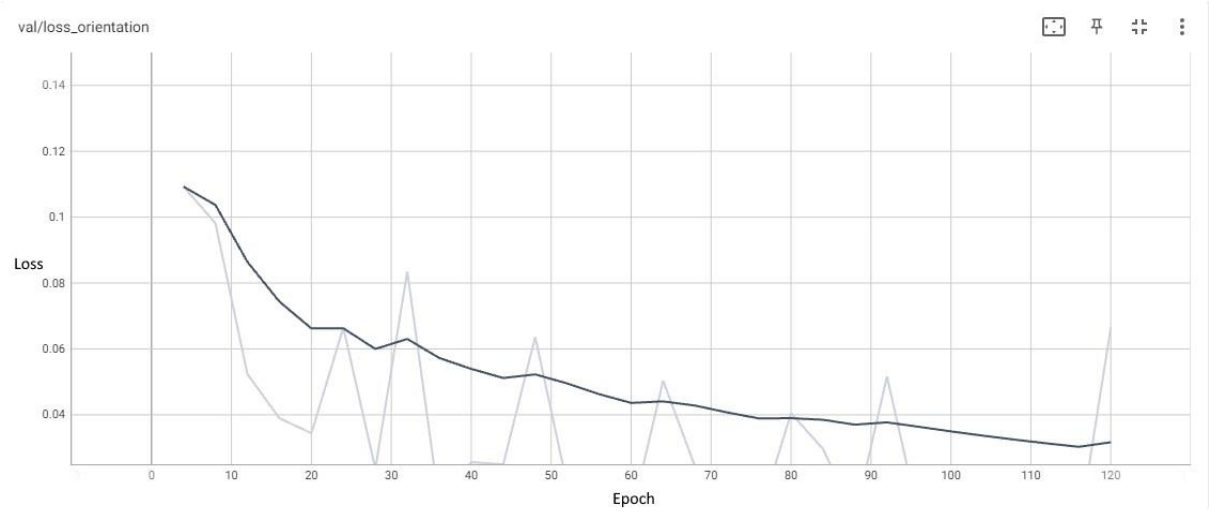


Figure 37: Orientation loss for validation (top view)

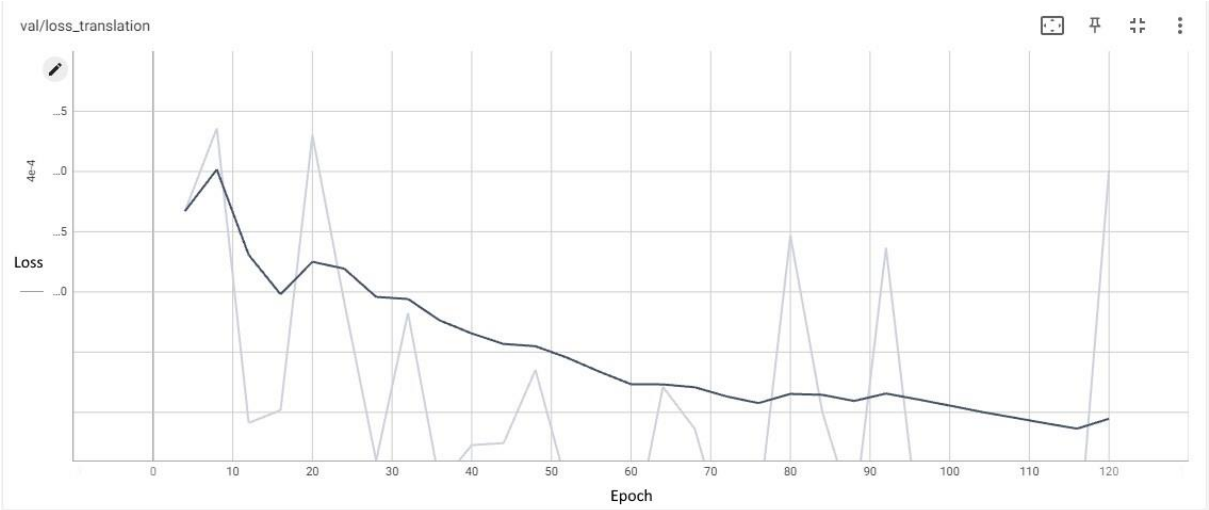


Figure 38: Location loss for validation (top view)

A picture of the model prediction and guiding the robotic arm to pick up the fish is shown in figure 39.

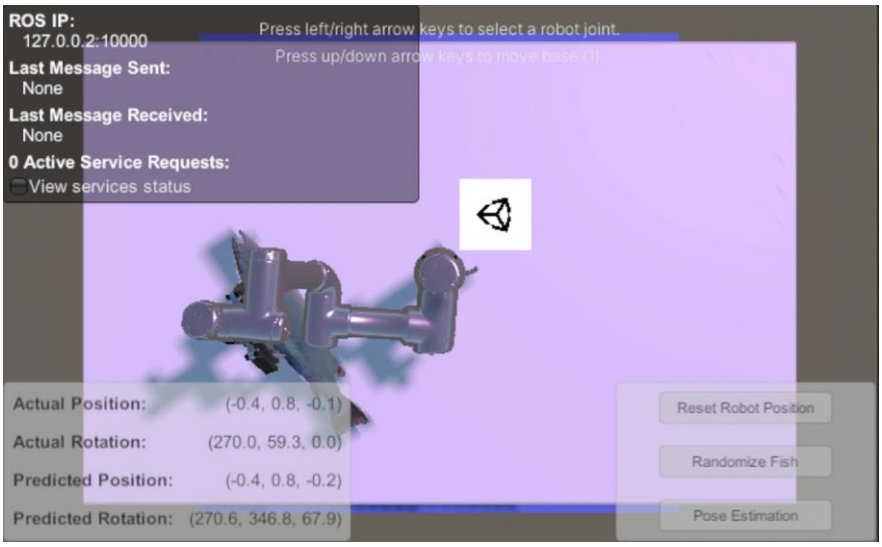


Figure 39: Robot arm detecting and trying to pick up the fish (top view)

4.3 Comparison

The comparison between the two camera positions for the training, validation and test data can be seen in the figures 40 and 41.

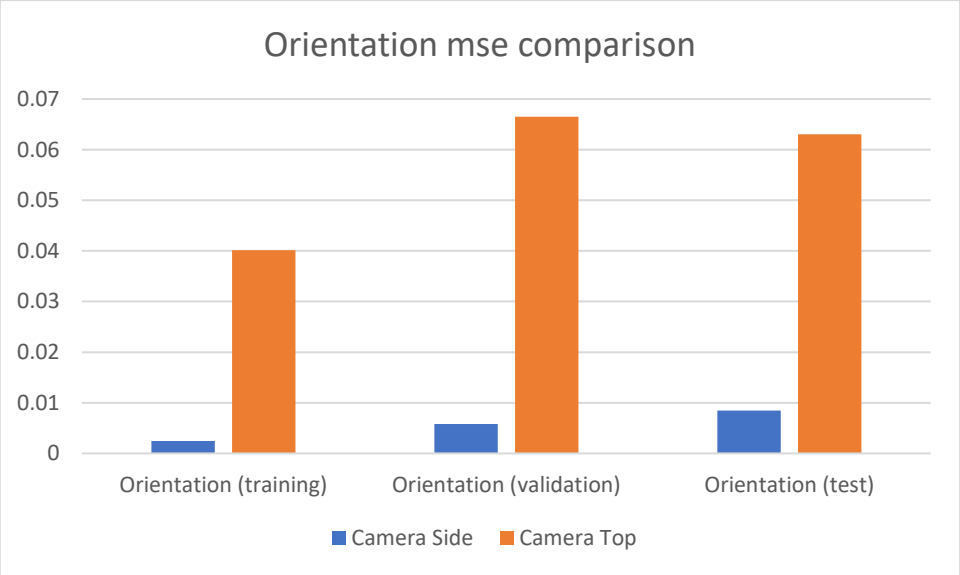


Figure 40: Comparison of the orientation MSE between side and top view

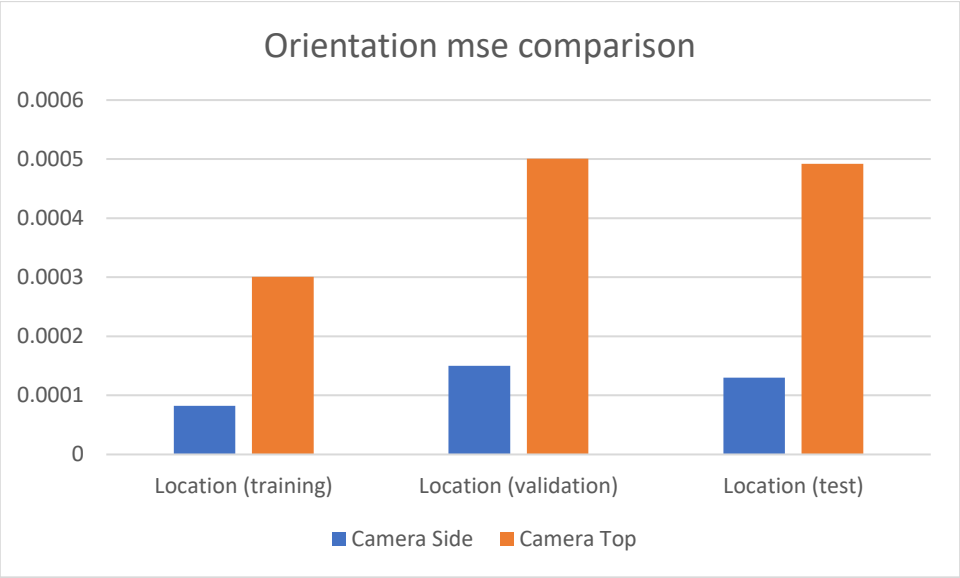


Figure 41: Comparison of the location MSE between side and top view

5 Discussion

The model demonstrates good performance in both location and position, as indicated by the low Mean Squared Error (MSE) values. It is evident that in both scenarios, the accuracy of predicting the fish's position is significantly higher compared to predicting its orientation. Which persists for training, validation, and test data.

As it is expected, for the location, it is mostly the matter of detecting the object on the scene. Since the location is reported and the center of the fish, the model can estimate is better since it only needs to detect the general shape of the fish.

But regarding the orientation of the fish, it should detect the different part of the fish (head, tail) to be able to predict the rotation, also the rotation along the x axis which determines which side of the fish is on the table, is much harder to predict. Since the model needs to consider more details.

To assess the impact of camera positioning, two models were compared: one with the camera placed on the side of the table and the other with a top-down view. The model utilizing the side camera exhibited better results.

This can be expected by comparing the generated data in figures 28 and 34. It can be observed that in the top view the robotic arm obscures a large portion of the picture and therefore there are many more situations where the fish is behind it. Therefore, in many situations this gives much less information to infer the position and orientation from what the camera sees.

The side view camera has a better view of the table that is less obscured. Even though the perspective view makes it harder to determine the location and rotation, the less obscured fish makes up for this, thus the model yields better results.

This is due to the position on the robotic arm, which is placed in middle to have more accessibility on the table, if the robotic arm is to be placed on the side of the table it loses half of its reachability which is another matter to be considered.

In the implementation within the digital twin environment, the fish's location is randomized, and based on the image captured by the virtual camera, the model predicts both the position and orientation. These values are then transmitted to the robot's controller, enabling it to pick and place the fish. While the robot successfully detected the fish's location, it encountered difficulties in grasping the fish. This can be seen in figures 33 and 39 in which the arm has moved to pick up the fish, but it wouldn’t be able to pick it up. This issue stems from the specific type and size of the robotic arm's gripper.

The simulated UR3 robotic arm lacked a suitable gripper size and type for this task. The simulation and model, however, can be employed to select and optimize an appropriate gripper. Additionally, the speed at which the robotic arm performs the task holds significance, and the simulator and model can aid in optimizing this aspect by addressing real-world constraints and the required speed.

6 Conclusion

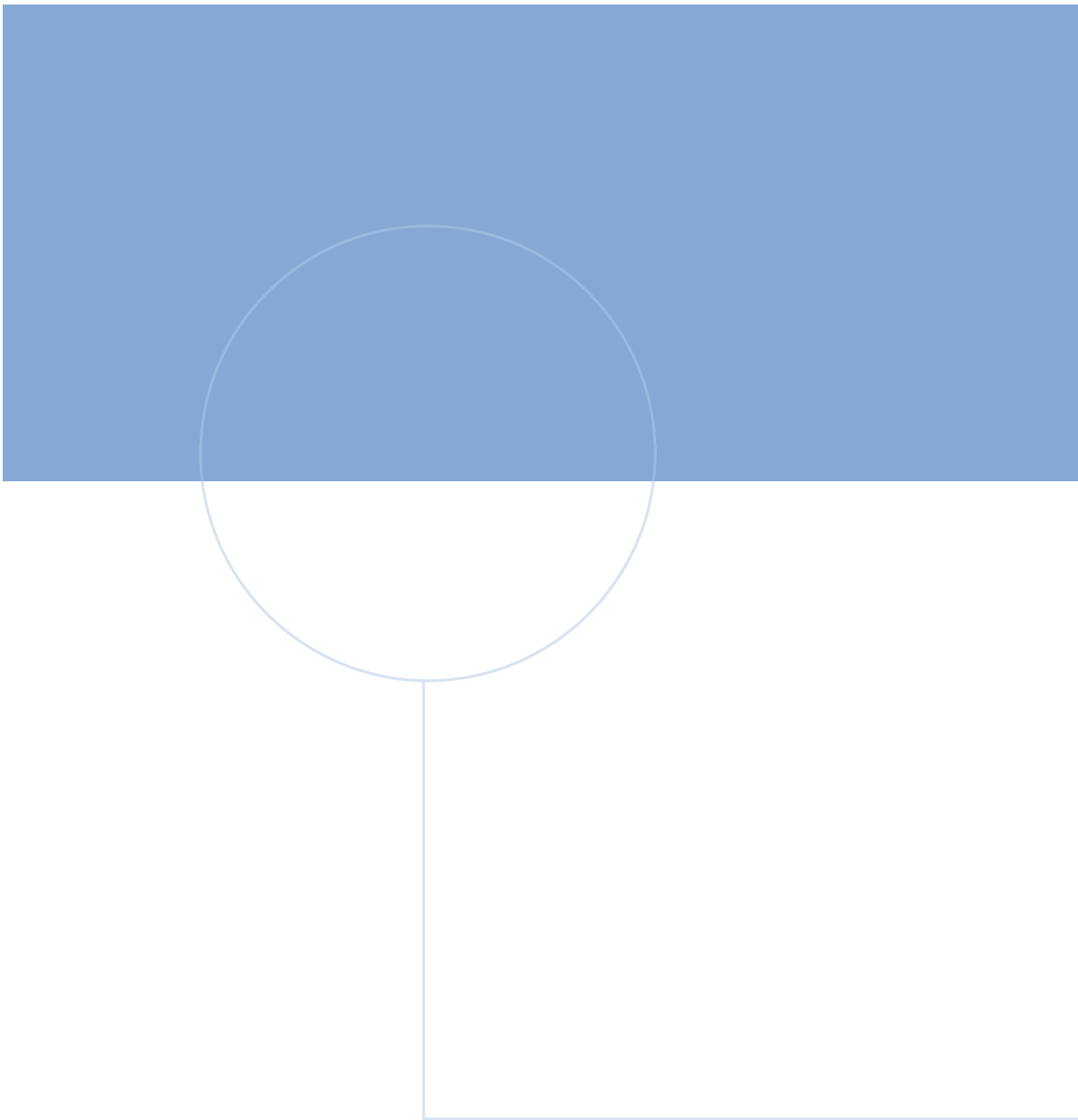
The proposed model structure demonstrates favorable outcomes for fish pose estimation problem. The side camera position proves to be superior in performance. By leveraging the trained model in conjunction with the simulator, it becomes possible to ascertain the optimal robotic arm type, gripper, or alternative mechanisms for fish retrieval based on the machine vision model.

In future research endeavors, the model can be utilized to forecast the parameters of fish movement on a conveyor belt. This application holds potential for devising a solution to identify and remove damaged fish, which can be implemented in processing factories.

References

1. Jogeir Toppe, F. *The nutritional benefits of fish are unique*. Available from: https://www.fao.org/in-action/globefish/fishery-information/resource-detail/en/c/338772/.
2. FAO, *The State of World Fisheries and Aquaculture 2022.* 2022: p. 266.
3. Johansen, U., et al., *The Norwegian seafood industry – Importance for the national economy.* Marine Policy, 2019. **110**: p. 103561.
4. Skjøndal Bar, E., *A case study of obstacles and enablers for green innovation within the fish processing equipment industry.* Journal of Cleaner Production, 2015. **90**: p. 234-243.
5. *Norges Sjømatråd, "Nøkkeltall,"* 2022; Available from: https://nokkeltall.seafood.no/.
6. Hilmarsdottir, G.S., et al., *Efficiency of fishmeal and fish oil processing of different pelagic fish species: Identification of processing steps for potential optimization toward protein production for human consumption.* Journal of Food Processing and Preservation, 2021. **45**(4): p. e15294.
7. Aday, S. and M.S. Aday, *Impact of COVID-19 on the food supply chain.* Food Quality and Safety, 2020. **4**(4): p. 167-180.
8. Einarsdóttir, H., B. Guðmundsson, and V. Ómarsson *Automation in the fish industry.* Animal Frontiers, 2022. **12**(2): p. 32-39.
9. IBM. *What is a digital twin?* 2022.
10. Fuller, A., et al., *Digital Twin: Enabling Technologies, Challenges and Open Research.* IEEE Access, 2020. **8**: p. 108952-108971.
11. Singh, M., et al., *Digital Twin: Origin to Future.* Applied System Innovation, 2021. **4**(2): p. 36.
12. Bitton, R., et al. *Deriving a Cost-Effective Digital Twin of an ICS to Facilitate Security Evaluation*. 2018. Cham: Springer International Publishing.
13. Zhao, Y.-P., et al., *Digital twin for rapid damage detection of a fixed net panel in the sea.* Computers and Electronics in Agriculture, 2022. **200**: p. 107247.
14. Lan, H.-Y., et al., *Digital Twin Architecture Evaluation for Intelligent Fish Farm Management Using Modified Analytic Hierarchy Process.* Applied Sciences, 2022. **13**(1): p. 141.
15. Yang, L., et al., *Computer Vision Models in Intelligent Aquaculture with Emphasis on Fish Detection and Behavior Analysis: A Review.* Archives of Computational Methods in Engineering, 2021. **28**(4): p. 2785-2816.
16. Hao, M., H. Yu, and D. Li, *The Measurement of Fish Size by Machine Vision - A Review*. 2016, Springer International Publishing. p. 15-32.
17. Alexopoulos, K., N. Nikolakis, and G. Chryssolouris, *Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing.* International Journal of Computer Integrated Manufacturing, 2020. **33**(5): p. 429-439.
18. Rezaei, F., et al. *Development of Digital Fish Simulation Model*. in *33th CIRP design conference*. 2023.
19. IBM. *What is machine learning?* ; Available from: https://www.ibm.com/topics/machine-learning.
20. Bishop, C.M., *Pattern recognition and machine learning*. 2006: New York : Springer, [2006] ©2006.
21. Ramachandran, P., B. Zoph, and Quoc, *Searching for Activation Functions.* arXiv pre-print server, 2017.

22. Pramoditha, R. *One Hidden Layer (Shallow) Neural Network Architecture*. 2021; Available from: https://medium.com/data-science-365/one-hidden-layer-shallow-neural-network-architecture-d45097f649e6.
23. Courville, I.G.a.Y.B.a.A., *Deep Learning*. 2016: MIT Press.
24. Mishra, M. *Convolutional Neural Networks, Explained*. 2020; Available from: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.
25. Yingge, H., I. Ali, and K.-Y. Lee, *Deep Neural Networks on Chip - A Survey*. 2020. 589-592.
26. Lecun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.
27. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, F. Pereira, et al., Editors. 2012.
28. Simonyan, K. and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition.* arXiv pre-print server, 2015.
29. Szegedy, C., et al., *Going Deeper with Convolutions.* arXiv pre-print server, 2014.
30. He, K., et al., *Deep Residual Learning for Image Recognition.* arXiv pre-print server, 2015.
31. Bezdan, T. and N. Bačanin Džakula. *Convolutional Neural Network Layers and Architectures*. Singidunum University.
32. Datagen. *Understanding VGG16: Concepts, Architecture, and Performance*. Available from: https://datagen.tech/guides/computer-vision/vgg16/.
33. Tobin, J., et al., *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.* arXiv pre-print server, 2017.
34. Unity. *Robotics simulation*. Available from: https://unity.com/solutions/automotive-transportation-manufacturing/robotics.



NTNU

Norwegian University of
Science and Technology