THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

MULTI-FINGER GRASP LOCALIZATION AND SEGMENTATION IN INFRARED IMAGES

Master's Thesis

MUSTAFA MOHAMMED AMEEN

İSTANBUL, 2021

THE REPUBLIC OF TURKEY

BAHÇEŞEHİR UNIVERSITY

GRADUATE SCHOOL OF

ELECTRICAL AND ELECTRONIC ENGINEERING GRADUATE PROGRAM

MULTI-FINGER GRASP LOCALIZATION AND SEGMENTATION IN INFRARED IMAGES

Master's Thesis

MUSTAFA MOHAMMED AMEEN

Thesis Supervisor: ASSIST. PROF. DR. TARKAN AYDIN

İSTANBUL, 2021

T.C.
BAHCESEHIR UNIVERSITY
GRADUATE SCHOOL

14/09/2021

MASTER THESIS APPROVAL FORM

Program Name:	Electric and Electronic Engineering Gradaute Program
Student's Name and Surname:	Mustafa Mohammed Ameen
Name of The Thesis:	Multi-Finger Grasp Localization and Segmentation in Infrared Images
Thesis Defense Date:	25/08/2021

This thesis has been approved by the Graduate School which has fulfilled the necessary conditions as Master thesis.

Prof. Dr. Fatma OZKUL

Institute Director

This thesis was read by us, quality and content as a master's thesis has been seen and accepted. as sufficient.

	Title/Name	**Signature**
Thesis Advisor's	Assit.Prof.Dr. Tarkan Aydin	
Member's	Assit.Prof.Dr. Ovgu Ergun	
Member's	Assit.Prof.Dr. Serkan Ayvaz	

ACKNOWLEDGEMENTS

A significant amount of time and effort has gone into the work of this thesis. While working on this topic, I gained beneficial experience and information regarding deep learning, computer vision, and image processing. When I started working on this topic, it was challenging since I did not have any background regarding machine learning and artificial neural network. However, in the end, I was delighted and gratified to myself and thankful for this opportunity to work in this exciting field.

I would like to express my sincere appreciation to my advisor, Asst. Prof. Dr. Tarkan Aydin, for his guidance and helpful support.

It would be unfair if I did not thank all the people who helped and supported me during my master's study. First, I must thank my beloved family, parents, brother, and young sister; without them, I am nothing. They always continue to support me with their prayers and encouraging words, even in my failure. I am very thankful for their support and happy to see how proud they are of their son. Secondly, I must thank my only best friend for his support during my study, and I also wish him all the luck and success during his master's study and on his road to achieving his dreams. I must also thank my beloved and my sweetheart girl, who was and still beside me during my journey, and I am very thankful for her support, prayers, and love to me.

I would like also to thank my colleagues, Alhareth Altaib, Abdullah Nazhat, and Mohammed Saeed, for their support and advice during my research.

İstanbul, 2021 Mustafa Mohammed Ameen

ABSTRACT

MULTI-FINGER GRASP LOCALIZATION AND SEGMENTATION IN INFRARED IMAGES

Mustafa Mohammed Ameen

Electrical and Electronic Engineering Graduate Program

Thesis Supervisor: Assist. Prof. Dr. Tarkan AYDIN

April 2021, 74 pages

This thesis presents a novel human contact patterns detection system that localizes and segments the grasping area (Fingers and Palm locations) using a 2D contact map for a single view for the visible part of the object captured by using a thermal camera.

In real-world robotics scenarios, mobile manipulators have often required a camera as an embedded system to grasp objects after localizing the most stable grasp contact points on them. Computer vision and natural language processing have significantly developed in deep learning. On the other hand, robotic grasp detection based on computer vision has not been widely adopted despite some success in it.

The proposed method uses deep learning models to localize and segment the fingers and palm locations using a data set that contains infrared images of the gasping area on objects used by humans hands. In each of the proposed models, we achieved positive predictions with minimum overfitting.

Keywords: Grasp Detection, Segmentation, Computer Vision, Deep Learning, Infrared.

ÖZET

ÇOK PARMAKLı KAVRAMA YERELLEŞTIRME VE SEGMENTASYON KıZıLÖTESI GÖRÜNTÜLERDE

Mustafa Mohammed Ameen

Elektrik-Elektronik Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Tarkan AYDIN

Nisan 2021, 74 sayfa

Bu tez, termal kamera kullanılarak yakalanan nesnenin görünür kısmı için tek bir görünüm için 2D temas haritası kullanarak kavrama alanını (Parmaklar ve Avuç içi konumları) yerelleştiren ve segmentlere ayıran yeni bir insan temas desenleri algılama sistemi sunar.

Gerçek dünyadaki robotik senaryolarda, mobil manipülatörler genellikle nesneleri kavramak için gömülü bir sistem olarak bir kameraya ihtiyaç duymaları gerekir. Bilgisayarla görme ve doğal dil işleme, derin öğrenmede önemli ölçüde gelişmiştir. Öte yandan, bilgisayar görüşüne dayalı robotik kavrama algılama, bazı başarılara rağmen yaygın olarak benimsenmemiştir.

Önerilen yöntem, insanların ellerinin kullandığı nesnelerin üzerindeki gaz alma alanının kızılötesi görüntülerini içeren bir veri seti kullanarak parmakları ve avuç içi konumlarını yerelleştirmek ve segmentlere ayırmak için derin öğrenme modellerini kullanır. Önerilen modellerin her birinde, minimum aşırı montaj ile pozitif tahminler gerçekleştirdik.

Anahtar Kelimeler: Kavrama Algılama,, Bölümleme Bilgisayar Görüşü, Derin Öğrenme, Kızılötesi Görüntüler.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OFABBREVIATIONS

ANN	:	Artificial Neural Networks.
AP	:	Average Precision.
C	:	Celsius.
CNN	:	Convolution neural network.
Convnets	:	Convolution Neural Networks.
IOU	:	Intersect Over Union.
K	:	Kelvin.
R-CNN	:	Region-Based Convolution Neural Network.
ReLU	:	Rectified Linear Unit.
SGD	:	Stochastic Gradient Descent.
T	:	Absolute temperature.
Tanh	:	The Hyperbolic Tangent Function.
TIR	:	Thermal Infrared.
FPN	:	Feature Pyramid Network

LIST OF SYMBOLLER

Wavelength	:	λ
Emission Constant	:	c_1 , c_2
Weight of the neuron	:	w_i
Input Feature	:	x_i
Bias	:	b
Activation Function	:	$f(z)$
Neuron Output	:	$\hat{y}$
The number of recall levels between 0 and 1	:	N
The precision at recall level r	:	P
Recall Level	:	r

1. INTRODUCTION

The goal of the computer vision area is to create systems that can process images just like humans or even better. The main tasks of computer vision can be divided into three different areas. Image classification is the ability to classify images into different classes depending on the content. Object detection is the ability to identify multiple objects and localize them in the image by putting a bounding box around it. Segmentation is the most difficult one by not only putting a bounding box around the object but also classifying every pixel in the image.

One of the essential tasks that computer vision can be used is the robotic grasp detection. Robotic grasp detection can be defined as a visual recognition problem in which the robot can use sensors to detect graspable objects in its environment. Household objects are the most used objects by humans. It is designed for use and manipulation by human hands. With these objects, humans can excel and then performing actions. If robots can enable this skill, it will potentially unlock more productive and natural human-robot interaction. The recent works for the robotic grasping of household objects focus on using large amounts of generalized data by random grasping actions; in this process, a parallel-jaw or suction-cup end effectors are used (Mahler et al. 2017) (Mahler et al. 2017) (Mahler et al. 2019). Many research papers used analytical approaches to synthesize grasping area from object shape that guarantees stability within their set of assumptions (Bicchi and Kumar 2000) (Prattichizzo et al. 2012). These researches contribute much valuable theoretical analysis of grasping; their assumptions or methods are often quite restrictive. In the real world, it makes purely analytic algorithms challenging to deploy. In contrast, data-driven applications that rely on machine learning techniques to learn features from many objects' representations (e.g., RGB image, texture, 3D mesh) can be used in newer deep learning approaches to predict grasps. For example, the grasps can be synthesized for novel objects by computing shape features for both objects, grasping hands, and using nearest-neighbor retrieval (Li and Pollard 2005). Despite this field's efficiency and reliability in the real world, it requires large amounts of data, which is hard to collect and

label. Also, these approaches are limited to regular and simple end effectors (e.g., parallel jaw grippers). In the other approach (Goldfeder and Allen 2011) (Goldfeder et al. 2009), grasping for known 3D objects generated through a grasping simulator plans grasps that maximize a given quality metric offline. After that, a scene is segmented when attempting to grasp an object in that scene. After that, we can find in the database the segmented object's closest model, and the precomputed grasps from the closest matching model are used on the new object. (Varley et al., 2015) proposed a deep learning architecture trained on RGB images patches of fingertip and palm position to detect the palm and the fingertips positions of stable grasps directly from partial objects views. These positions are generated from grasps computed on complete object models using a grasping simulator. The grasp quality metrics such as force closure were able to estimate by the architecture without the need to explicitly calculate the given metric.

(Kumra1 and Kanan 2017) In this paper, proposed a robotic grasp detection system to predict the most stable and best grasping pose of a parallel-plate robotic gripper of objects using an RGB-D image of the scene, by extracting the features from the scene by using a deep convolution neural network and then predicting the grasp configuration for the desired object by using an external convolutional neural network. (Corona et al. 2018) in this paper, a proposed method identifies the type of garment and allows a robot to bring the garment to a known pose by performing a search of two grasping points. Synthetic and real images were used to train a hierarchy of three convolution neural networks with different levels of specialization. It showed promising results of using an active search strategy to grasp directly from predefined grasping points. (Watson et al. 2017) in this paper, an investigator was made about deep learning uses to develop a real-time scheme on a physical robot. A convolution neural network was trained using a Baxter Research Robot and Kinect sensor in a supervised manner to regress grasping coordinates from RGB-D data. (Kang et al. 2020) in this paper, a proposed framework of a full convolution neural network-based visual perception framework for autonomous apple harvesting was made to determine the proper grasp pose to guide the robotic execution by including a multi-function neural network for fruit recognition and a Pointnet network. The Pointnet grasping estimation predicts the grasp pose for each fruit as output by taking the point of each fruit as input.

In this thesis, we used a dataset called ContactDB (a novel dataset for household objects containing IR images of the rich hand-object contact that occurs during grasping (Samarth Brahmbhatt et al. 2019)) to localize and segment the grasping areas (fingers and palms positions) as it is stable grasps directly from partial object views by using deep learning.

The thesis starts by defining infrared images and deep learning theory, providing the foundations for presenting the physical background of infrared images, the deep learning image classification system, and the related works. It takes from that to specify the implementation for the proposed object detection and segmentation system. Finally, the thesis ends with a conclusion of this work as well as the obtained results from it.

2. THEORY AND LITERATURE REVIEW

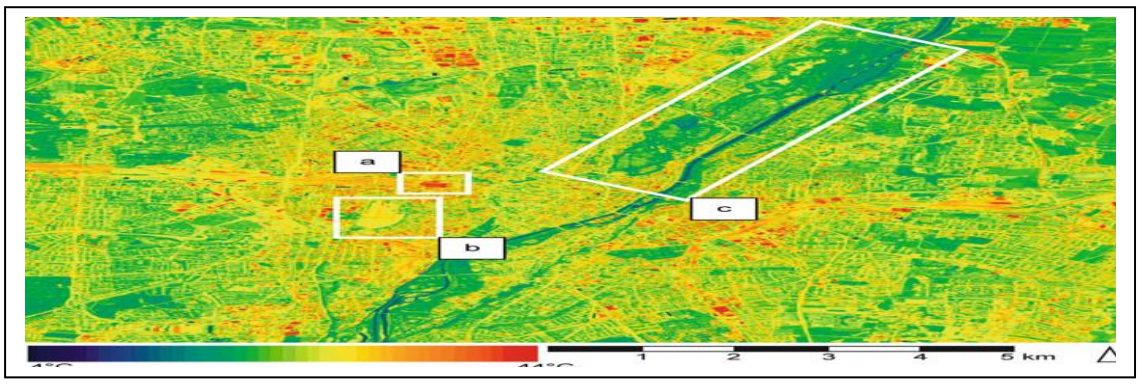
2.1 INFRARED IMAGE THEORITICAL BACKGROUND

This part begins by laying out the theoretical dimensions of the infrared images by explaining electromagnetic radiations, thermographic camera technology and the physical background behind it, camera's sensors, and at last, the infrared image classification problem.

2.1.1 Thermal Infrared Radiations

The electromagnetic radiations radiate from objects with a temperature above 0 Kelvin (273 °C). The earth's temperature is above 300 Kelvin, and the electromagnetic emittance peak is located in the thermal infrared, TIR, domain at about 9.7 µm (Tipler 2000)(Sabins 1996).As radiation is defined as transferring energy, according to (Sabins 1996), it works in/through a vacuum. By enabling the derivation, thereby TIR sensors can detect the thermal radiance images of objects located on the earth's surface.

Figure 2.1: Thermal daytime land surface temperature image of Munich 1982, based on data acquired with an airborne Bendix Scanner. Areas marked include the Munich central train station (a), the location – a large meadow – of the annual Oktoberfest (b) and the southern part of the English Garden, Munich's largest inner-city park (c) (Imagery courtesy of DLR).



Source: Kuenzer, C. et. al., 2013, *Thermal Infrared Remote Sensing*, London: Springer.

2.1.1 Thermographic Camera Technology

These cameras are commonly applied in sensing or capturing the radiometric information about temperature distribution of particular objects, for instance, building monitoring, material testing, and quality control. Furthermore, the thermographic cameras' geometric calibration is disregarded, excluding a few types of research that deal with this area (Buyuksalih and Petrie 1999) (Luhmann et al. 2010); nevertheless, due to the continuous improvement of the quality of the infrared image data, the geometric processing of these data will become more significant.

- **Physical Background**

  According to Planck (1900), Plank's emission law is the formula that describes the specific emission of a particular object as a function of absolute temperature and wavelength.

$$M_{\lambda}\left[\frac{W}{10^{-10}.m}\right] = \frac{c_1}{\lambda^5}\,\frac{1}{exp(\frac{c_2}{\lambda.T})-1} \qquad \textbf{(2.1)}$$

  Where:

  ƛ: wavelength[µm]

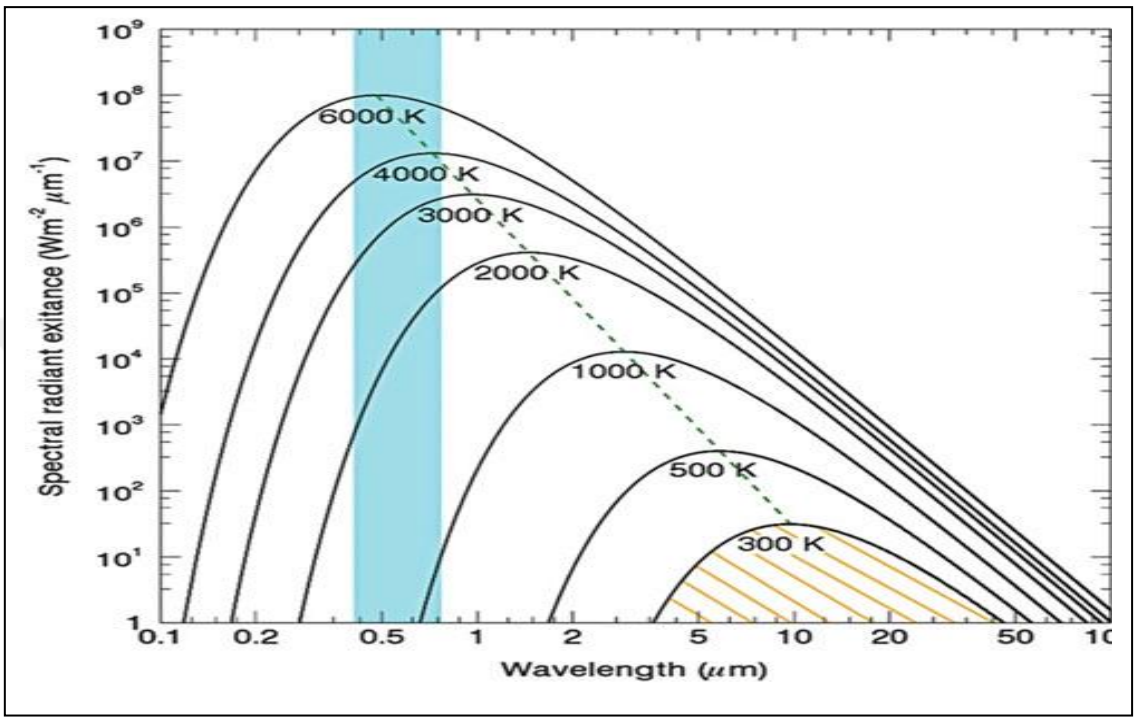
  c_1 : emission constant = 3,7418.10^{-16} W.m^2

  c_2 : emission constant = 1,4388.10^{-2} K.m

  T: absolute temperature [K]

  Figure 2.2 shows that the human eye or conventional camera can sense electromagnetic radiation radiating from objects with a temperature above 1,000 K.

Figure 2.2: Blackbody radiation curves at different selected temperatures, as derived from Eq. 2.1. The laws of Planck, Stefan-Boltzmann (marked area under the 300 K curve) and Wien (green dotted line) are depicted in this figure. The blue bar indicates the VIS region.



Source: Kuenzer, C. et. al., 2013, *Thermal Infrared Remote Sensing,* London: Springer.

2.1.2 Thermographic Camera's Sensors

According to Nolting (2007), there are two types of sensors:

1-Quantum Detectors

2-Thermal Detectors

Both sensors have different working mechanisms, In the Quantum detectors, the electrons are set free between two layers of the semiconductor device, and that is based on the inner photoelectric effect. The advantages of quantum detectors are that it is susceptible to temperatures below (-0.01 Kalvin) and above (+0.01 Kalvin) with fast performance. However, according to (Peltier or Sterling elements) (Fouad and Richter 2008), the quantum detectors' disadvantage is that they need an external cooling system.

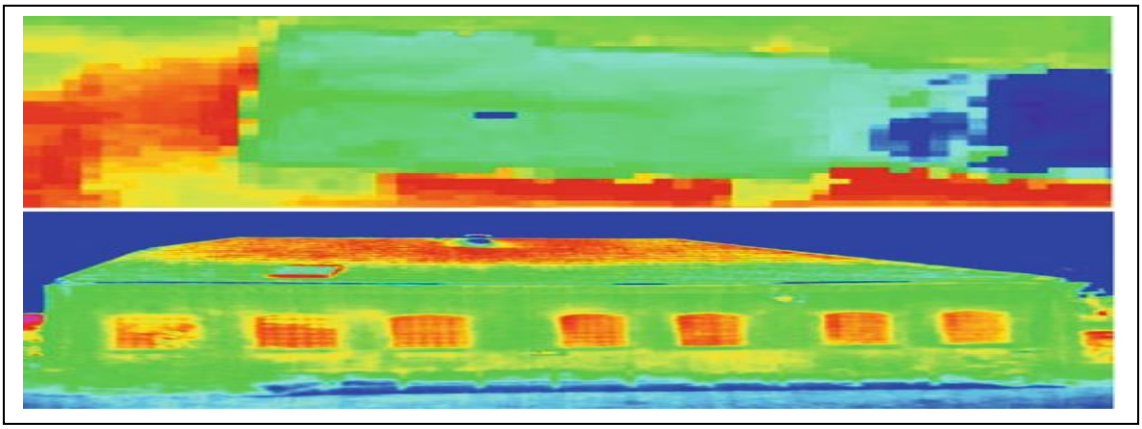
For (Hierl 2008), the Thermal detection mechanism is defined as changing the detector's electrical properties caused by changing the temperature of the detector element.

The advantage of thermal detectors is that they do not need any cooling system. However, these detectors' main disadvantages are that they are slower and less sensitive for temperatures above (+0.1 Kalvin) and below (-0.1 Kalvin).

Figure 2.3: Investigated thermographic cameras (forward looking infrared, FLIR).

Source: Kuenzer, C. & Dech S., 2013, *Thermal Infrared Remote Sensing*, London: Springer.

Figure 2.4: Original thermal images.



Source: Kuenzer, C. & Dech S., 2013, *Thermal Infrared Remote Sensing*, London: Springer.

2.2 DEEP LEARNING

This section will give a review of deep learning, starting with the definition of the artificial neural network, activation function, and finally, convolution neural network.

2.2.1 Artificial Neural Network

The term 'Neuron,' as to how we refer to the nerve cells in the brain (Rojas 1996), can be defined in deep learning as the basic computational unit that can perform a specific task. The computational steps of a single neuron can be summarized into:

1) Calculating z by combining linearly all inputs x_i :

$$z = w_1x_1 + w_1x_1 + \ldots\ldots + w_{n_x}x_{n_x} + b \quad \textbf{(2.2)}$$

2) Calculating f from z :

$$\hat{y} = f(z) = f(w_1z_1 + w_1x_1 + \ldots\ldots w_nx_n + b) \quad \textbf{(2.3)}$$

Where:

w_i: Weights of the neuron.

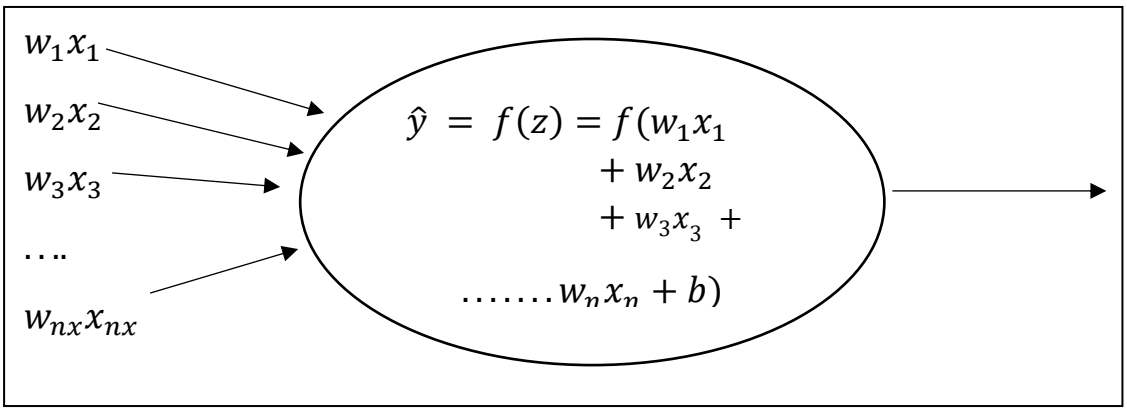
x_i: Input Features.

b: Bias.

$f(z)$: Activation function.

$\hat{y}$: Neuron Output.

Figure 2.5: The computational graph of single neuron.

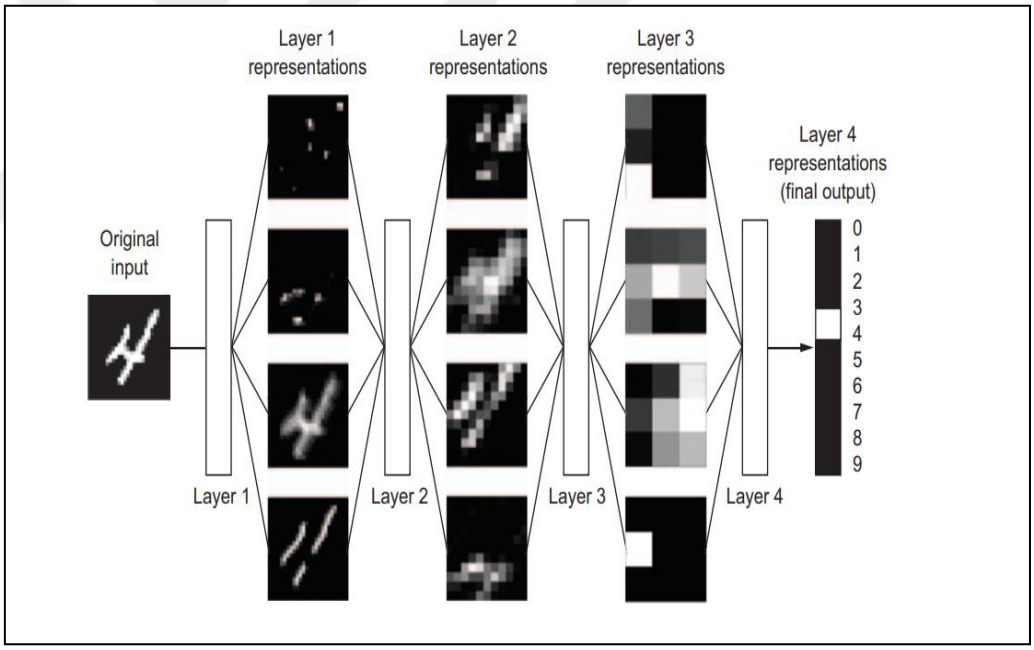


The term ‘Artificial Neural Network’ can be used to define a large set of neurons connected in parallel to perform and build complex systems. This architecture, which consists of multiple layers (input, hidden, and output layers) where each layer contains multiple neurons, can be manipulated by changing the basic units, how they are computing the results, and how they are connecting to each other.

- The Learning Process

The main idea of deep learning is to map the input data (e.g., images) to the output target (e.g., label "dog"), and that can be achieved by excessive exposure to input-to-target samples. Through this process, the network's layers transform the input data into increasingly informative representations to the final output.

Figure 2.6: Deep representations learned by a digital-classification model.



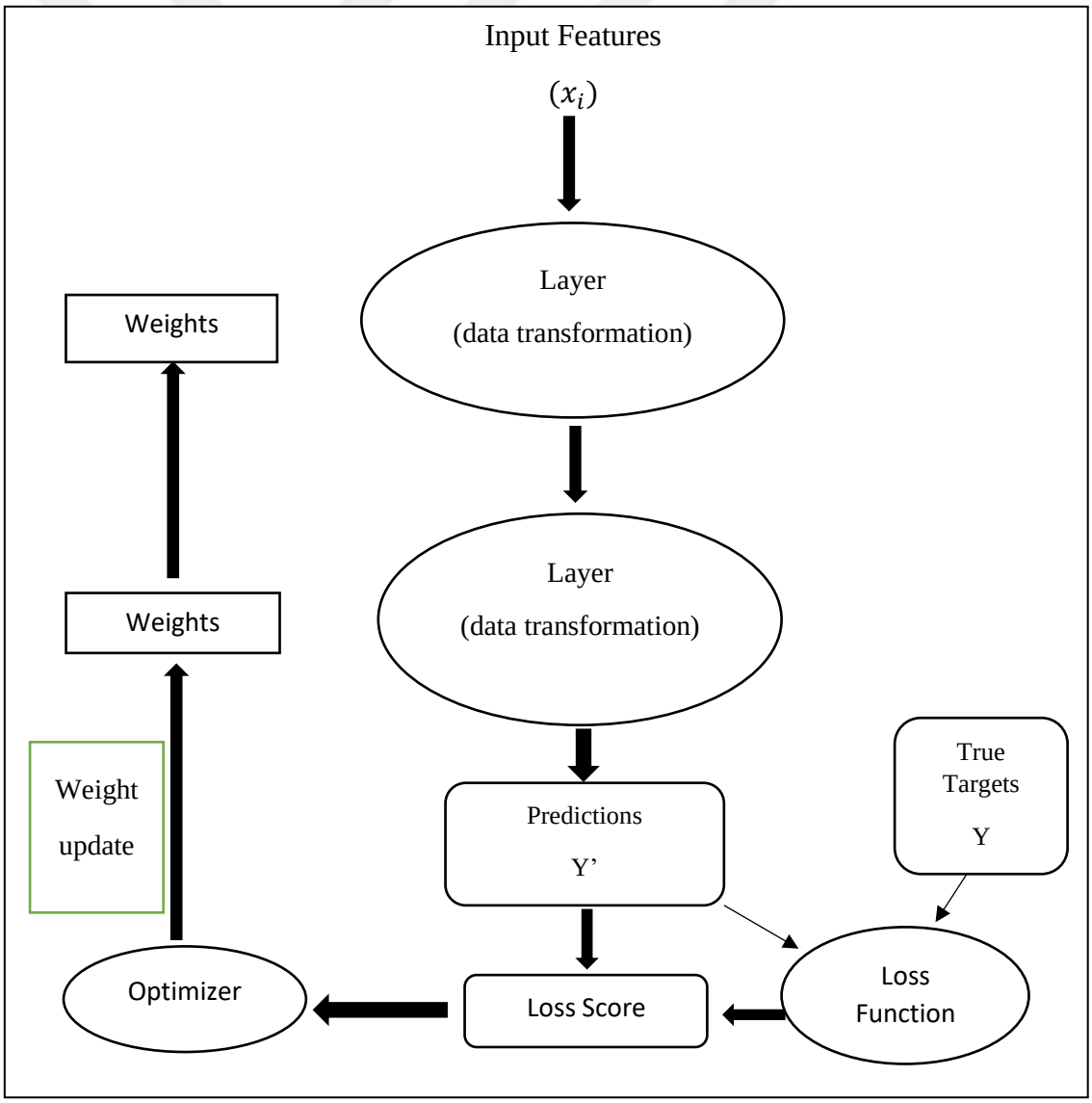
Source: Chollet, F., 2018, *Deep Learning with Python*, USA: Shelter Island.

During the learning process, each layer stores a bunch of numbers called weights. These weights (also called “parameters”) indicate the results of every neuron's computational process on the input data in a single layer. Due to that, the network can find the correct combination of the weight's values that can map the input features to the desired target.

- Loss Functions

The loss functions (also called "objective functions") are used for directing the input features to the desired targets through the learning process by computing the distance score between the output and the input to ensure the correct prediction of the network. The neural network uses the result of the loss function as a feedback signal during the training process to make the required adjustment through the optimizer of the network by implementing the Backpropagation method (which is the backbone algorithm of the deep learning) on the weights' values to minimize the distance between the input and the needed output.

Figure 2.7: The Loss function as feedback signal.



2.2.2 Activation Function

To change the neuron's output in the neural network, we must apply a mathematical function called the activation function. This function's primary purpose is to transform (z) in the neuron output ($\hat{y}$). In deep learning, there are multiple activation functions for various purposes. In the section below, we will explain the most common ones: -

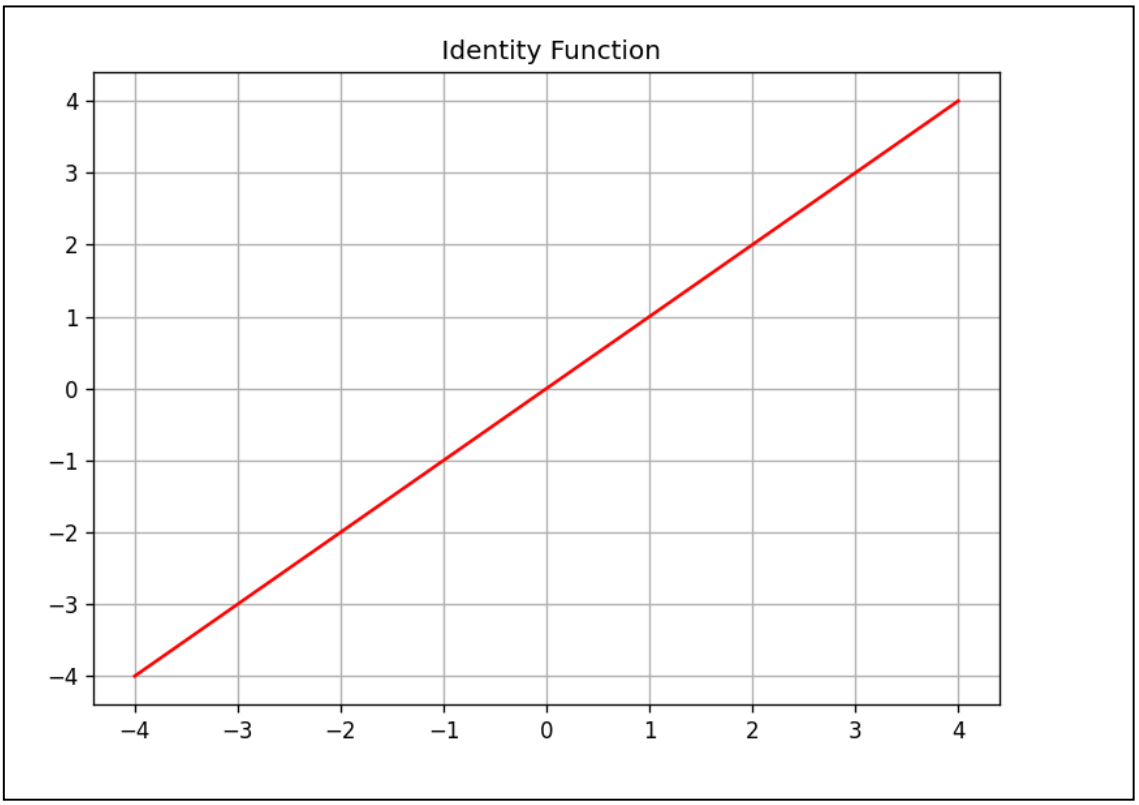
a) Identity Function

The identity function is the most basic activation function in deep learning. It only returns the same input without any change.

Formula:

$$f(z) = I(z) = z \tag{2.4}$$

Figure 2.8: Identity function.



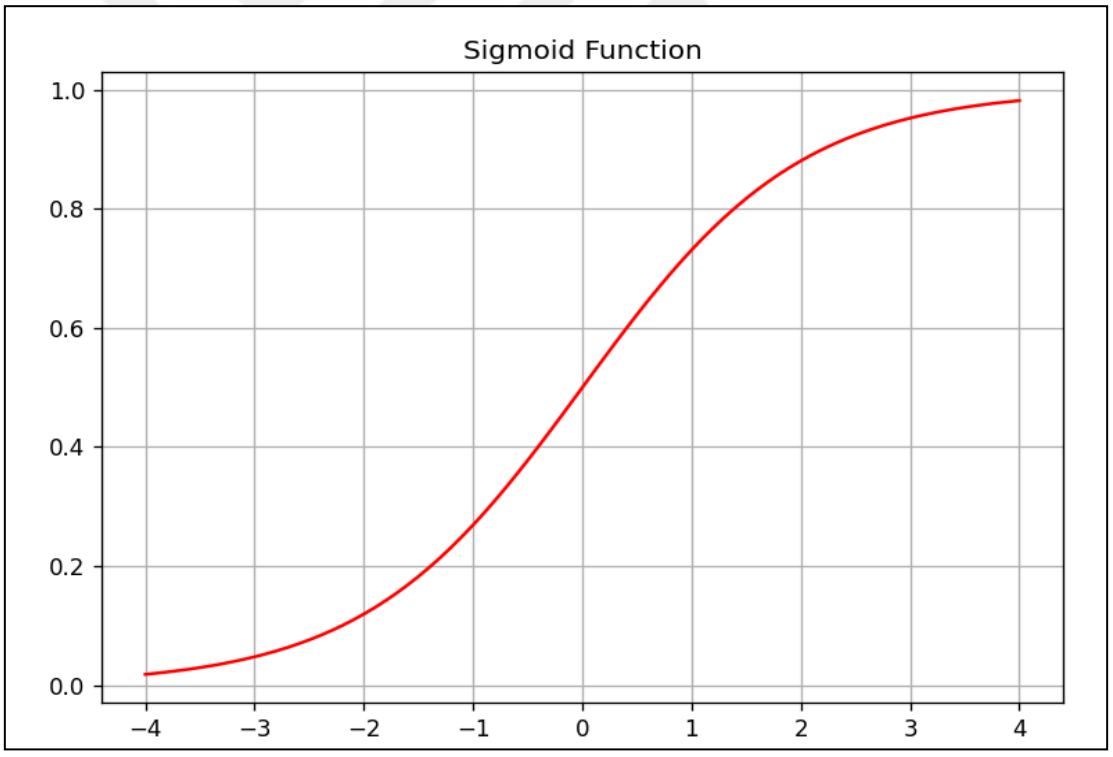
b) Sigmoid Function

The sigmoid function is the most popular and common activation function in deep learning, which only returns values between zero and one ($0 < f(z) < 1$). This function is beneficial when the neural network's output is a probability value ($0 < p < 1$).

Formula:

$$f(z) = \sigma(z) = \frac{1}{1+e^{-z}} \quad \textbf{(2.5)}$$

Figure 2.9: Sigmoid function.



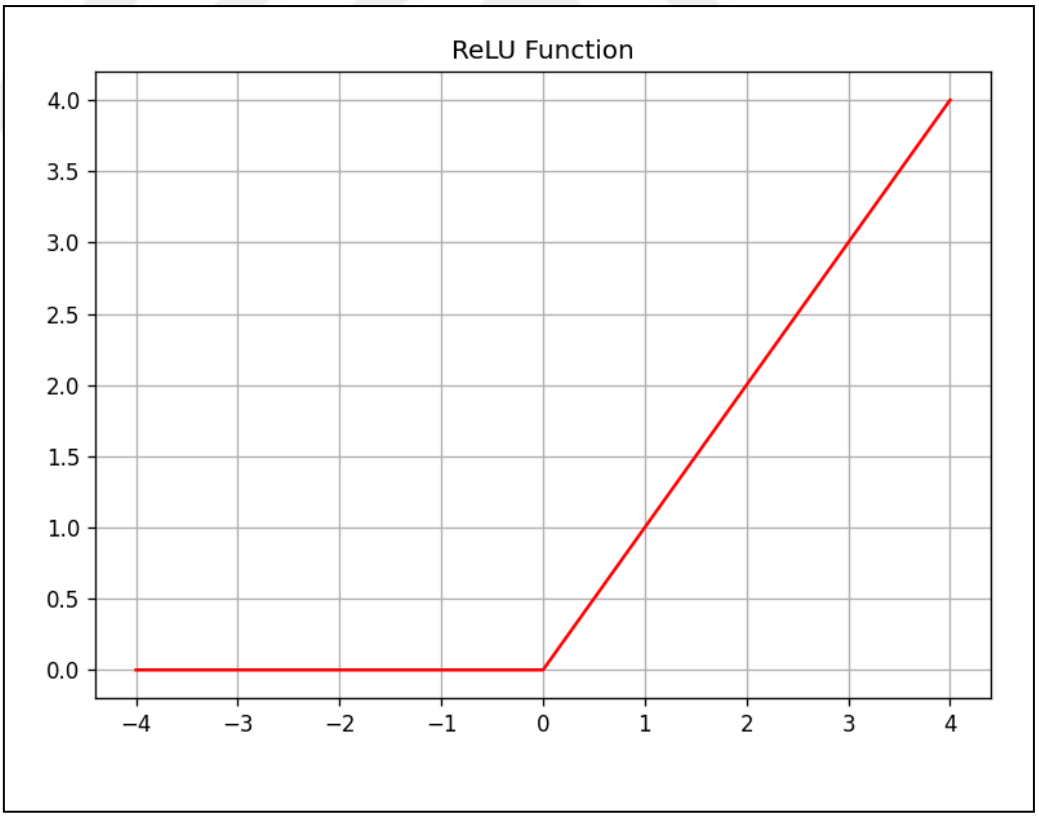
c) Rectified Linear Unit (ReLU)

This function is a popular activation function in deep learning not because it is only fast and straightforward but also because it has advantages on sigmoid function on its computational efficiency, not vanishing gradient and better convergence performance.

Formula:

$$f(z) = \max(0, z) \tag{2.6}$$

Figure 2.10: ReLU function.



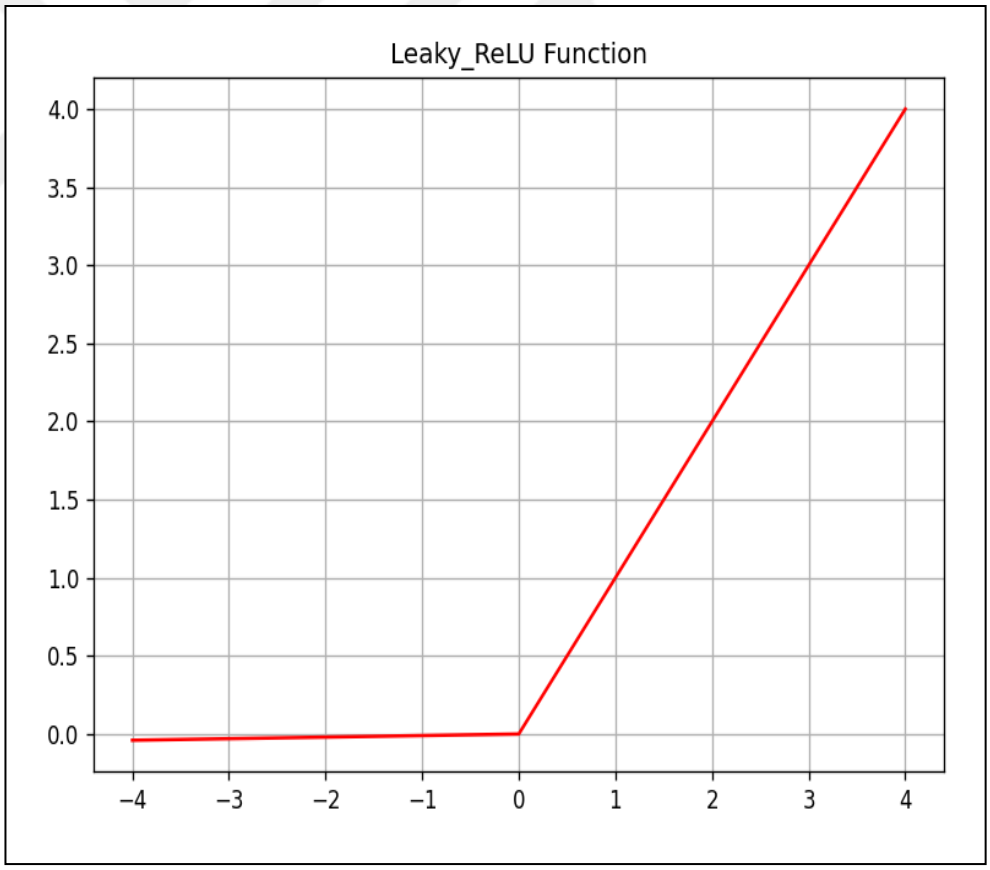
d) Leaky ReLU

This activation function, also known as (Parametric Rectified Linear Unit), has advantages on ReLU on having a slight slope for negative values and speeding up the training process as it is more balanced than the ReLU.

Formula:

$$f(z) = f(x) = \begin{cases} \alpha z, & z < 0 \\ z, & z \geq 0 \end{cases} \quad \textbf{(2.7)}$$

Figure 2.11: Leaky ReLU with $\alpha = 0.05$.



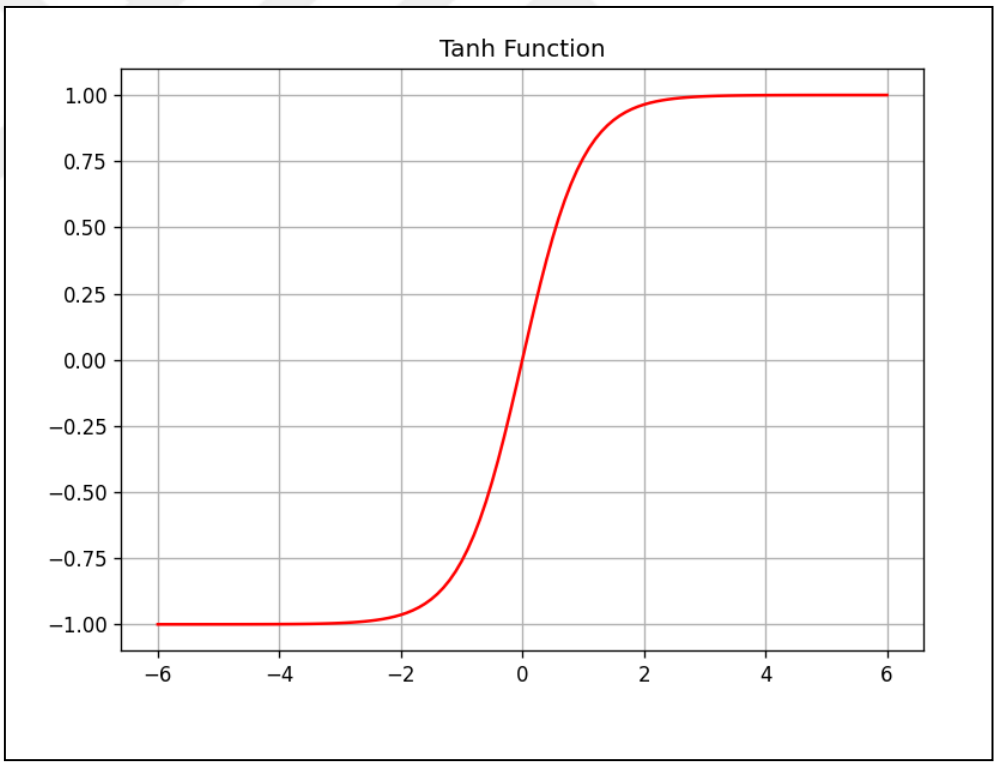
e) The hyperbolic tangent function(tanh)

This activation function is a scaled version of sigmoid, which is smooth and differentiable. This function has an advantage over the sigmoid function as it is more stable than the last one. Both tanh and sigmoid make artificial neural networks heavy as they are firing all the time. ReLu avoids this problem by not firing all the time (Shanmugamani 2018, p. 9).

Formula:

$$f(x) = 1 - \tanh^2(z) \quad \textbf{(2.8)}$$

Figure 2.12: tanh function.



In addition to these activation functions, several other activation functions have been introduced to this field, such as ArcTan, Exponential Linear Unit, Softpuls, and Swish, to serve the purpose of fast performance and achievement good prediction results during the training.

2.2.3 One-Hot Encoding

This representation method or technique is used in the classification problem when the target variables are in string form. One-hot encoding converts the string labels into one-hot encoded vectors of [0,1], where 1 at the target label index but with 0 everywhere else. For example, in the cat and dog classification problem, we use one-hot encoding to represent them by [0,1] and [1,0], respectively. In the case of more than two labels (e.g., 1,000 classes), the one-hot vectors' size will be equal to the classes' size (e.g., 1,000 integers) with all zeros but 1, no assumptions about the similarity of the target variables can be mad by one-hot encoding. By using one-hot encoding besides with softmax, multi-class classification becomes possible in ANN (Shanmugamani2 018, p.12).

2.2.4 Softmax

In the multi-class classification problem, softmax considers very useful by which we can force the neural networks to output the sum of 1. The softmax's output considers as part of a probability distribution. Softmax function can be considered as a kind of activation function with the specialty of output summing 1. By dividing the output by the summation of all other values, softmax can convert the output to the probabilities. For the optimization, the euclidean distance can be obtained from the softmax probabilities and one-hot encoding (Shanmugamani 2018, p.12).

2.2.5 Cross-entropy

Cross-entropy considers as a better cost function to optimize, by which we can compare the distance between the outputs of the softmax and the one-hot encoding. Cross-entropy considers as a loss function for which error has to be minimized. For every given data to every class, the neural networks estimate the probability. This probability has to be maximized to the correct target class. Cross-entropy considers as the summation of negative logarithmic probabilities. This value can be used for numerical stability. We can maximize the function by minimizing the negative of the same function (Shanmugamani 2018, p.12).

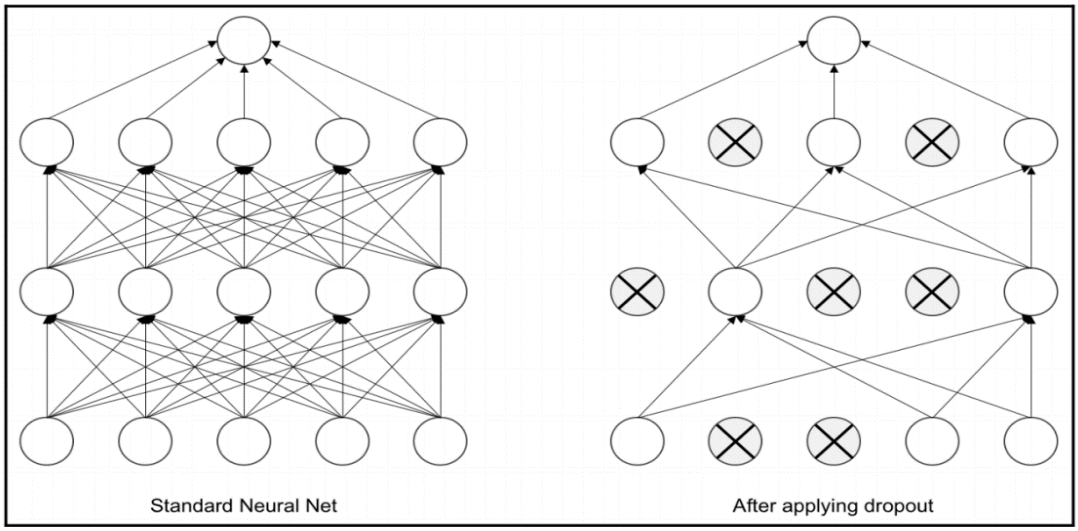
2.2.6 Regularization Methods

Regularization methods are using to avoid the overfitting of ANN:

a) Dropout

The dropout method considers as an effective way used by neural networks to avoid overfitting. The neural network, during the training, is crippled by the dropout layer by removing hidden units stochastically. In combining several neural networks, we can use dropout to randomly select a few hidden units that will lead us to different architectures for each training case. This case considers as an extreme case of bagging and model averaging. During the inference, dropout considers as unnecessary (Shanmugamani 2018, p.13). As shown in the figure below the neurons are randomly trainded:

Figure 2.13: The dropout.



Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

b) Batch normalization

The batch normalization method, or batch-norm, is used in the neural network to increase its stability and performance. The output of a layer is normalized using batch normalization with zero mean and a standard deviation of 1; therefore, the overfitting is reduced, and the training becomes faster. For

training complex networks, batch normalization considers very efficiently (Shanmugamani 2018, p.13).

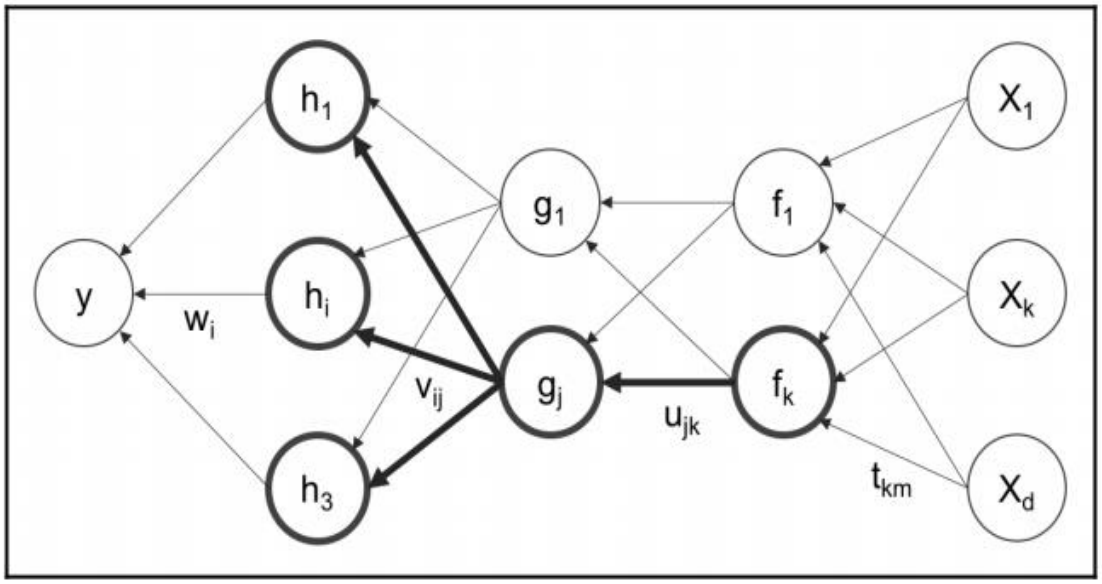
c) L1 and L2 normalization

Both L1 and L2 are used in regularization; the difference is that L1 is used to make the weights zero by penalizing its absolute value. The L2 is used to make the weights smaller during the training by penalizing the weight's squared value. Both of them assume that models with smaller weights are better (Shanmugamani 2018, p.14).

2.2.7 Backpropagation

This method is commonly used for training artificial neural networks. During the training, the weights are updated from the backward based on the error calculated (Shanmugamani 2018, p.14). Gradient descent can be used after calculating the error to calculate the weight updating.

Figure 2.14: Backpropagation.

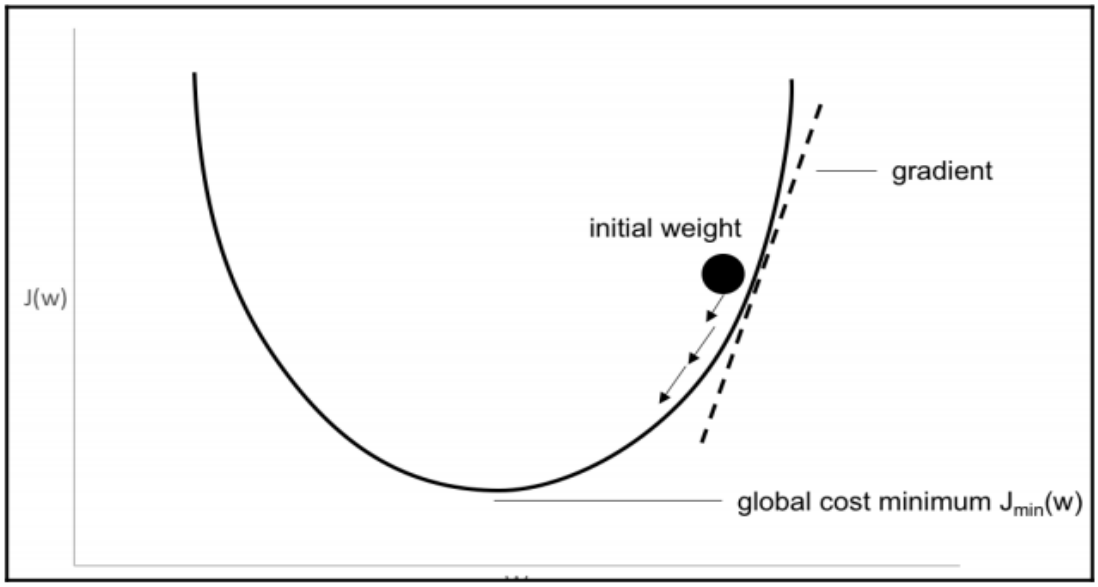


Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

2.2.8 Gradient Descent

The objective of the gradient descent method is to reach the global maximum by performing multidimensional optimization. The gradient descent method is used in many machine-learning models, as it considers a popular optimization technique. One widespread implementation of the gradient descent algorithm in neural networks is the stochastic gradient descent (SGD). The optimization aims to achieve that minimal error by calculating the error value and changing the weights. The negative of the gradient descent of the loss function considers the direction of finding the minimum. How big the step can be determined by the learning rate. Artificial neural networks with nonlinear activations will have local minima (Shanmugamani 2018, p.15).

Figure 2.15: The gradient descent.



Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

2.2.9 Stochastic Gradient Descent

SGD has no difference from the gradient descent, except that the stochastic gradient descent's primary purpose is to train only partial data every time. Also, it works better in practice for optimizing non-convex cost functions. For example, even one example can be used for training. The parameter's name is mini-batch size (Shanmugamani 2018, p.15).

2.2.10 Hyperparameter Tuning

Other parameters in the artificial neural network, not only the numerical parameters (e.g., the learning rate and regularization), can be tuned to get the most immeasurable results during the training. For example:

a) The number of epochs can be tried to train the model for a more extended period.
b) The choice of optimizer can be tried for different optimizers (e.g., gradient descent and Adam).
c) Varying the regularization method can be tried for applying several ways of regularization.
d) Choice of activation function can be tried for different functions (e.g., ReLU and Sigmoid).
e) The number of layers of neurons in each layer can be tried for various configurations.
f) Learning rate decay methods can be tried in case of not using optimizers that do this already.
g) Mini-batch size can be tried for a different size. For example, if the data set is small, the batch gradient descent can be used, and if the data set is large, mini batches will be more efficient.
h) Finally, weigh initialization methods can be tuned for better results.

Learning rate, regularization parameter can be classified into continuous real numbers where we can assume any value, the number of hidden layers, number of neurons in each layer, and number of epochs can be classified into discrete or theoretically an infinite number of values, and the optimizer, activation function, and the learning rate decay method can be classified into discrete with an only finite number of possibilities.

2.2.11 Convolutional Neural Network (CNN)

In this section, we will look at the main components of the convolution neural network: Kernels and Pooling layers.

The convolutional neural network is an advanced architecture in deep learning with more time and memory efficiency in analyzing multidimensional data (such a "images") than the fully connected network. The CNN base is ANN but with some changes to make it more suitable for analyzing many images. The multiple challenges in image analysis tasks such as object detection, classification, recognition, etc., make traditional image classification methods that included feature engineering are not suitable for working with high accuracy in rich environments, even with experts in the field. From these challenges, the idea of feature learning by ANN or CNN came out (Fawzy 2018).

The main problem with using the regular ANN with images is that all neurons within a single layer have no connections among them. Therefore, the number of neurons in the network will increase, causing overfitting.

The convolution neural network use transformation in its filters for encoding to detect features and patterns in images. Therefore, the deeper the layer, the more abstract the pattern is. Because the images are large in size, which will increase the model's size, the Convolution neural network arranged its neurons in a volumetric fashion to take advantage of the volume. Also, CNN has fewer parameters than regular ANN, making it more efficient in the term of time and memory during the training process (Shanmugamani 2018, p.17).

a) Convolution

One of the essential techniques in image processing used for sharpening, blurring, embossing images is kernels or sometimes called filters. Kernels are one of the main components of a Convolution neural network, and it is consisting of matrices of small integer numbers (e.g., 3 or 5) with dimensions $n \times n$. There are different types of kernels used for various purposes, for example:

Detection of horizontal edges filter

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad \textbf{(2.9)}$$

Detection of vertical edges filter

$$\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \quad \textbf{(2.10)}$$

Detection of edges when luminosity changes drastically filter

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad \textbf{(2.11)}$$

Blur edges in image filter

$$-\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \textbf{(2.12)}$$

Convolution is defined as the process of multiply two input tensors, one of them is the image (A matrix), which usually is bigger, and the other is the kernel (K matrix), which generally has dimensions of 3 x 3 or 5x 5, to produce one output tensor. The formula:

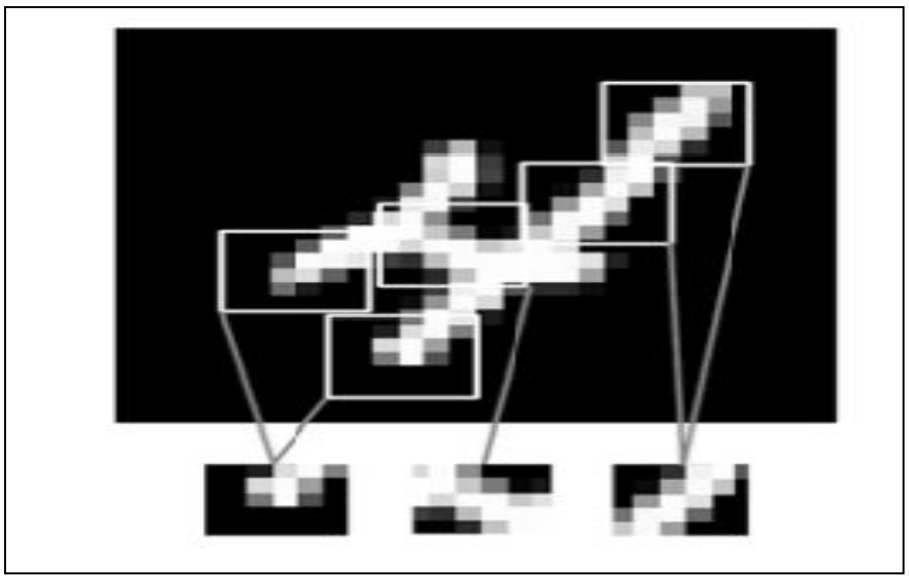
$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} * \begin{pmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{m1} & \cdots & K_{mn} \end{pmatrix} = \sum_{i=1}^{3} \sum_{j=1}^{3} a_{ij} k_i j \qquad \textbf{(2.13)}$$

Matrix(A): the image

Matrix(K): filter or kernel

In deep learning, the convolution layers are used to extract local patterns from images by using 2-D windows as inputs, as shown below:

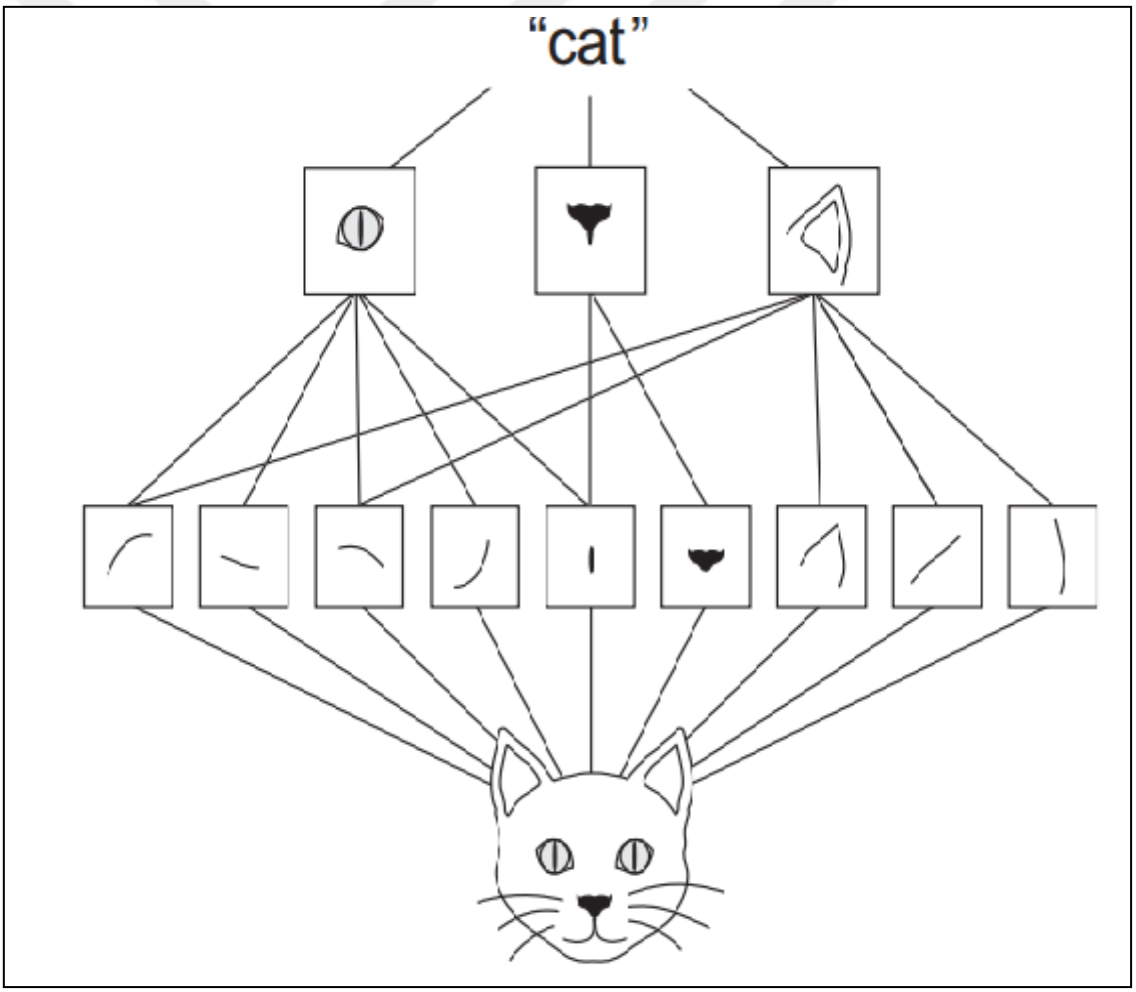
Figure 2.16: Images can be broken into local patterns such as edges , textures, and so on.



Source: Chollet, F., 2018, *Deep Learning with Python,* USA: Shelter Island.

The data produced by the convnets is more efficient in image processing because it can recognize the patterns that the convolution layers have learned in anywhere. Also, because the convolution layers can learn the spatial hierarchies of patterns, the early layers will learn the small local patterns. The last layers will learn the larger patterns. Therefore, it can be very efficient to learn complex and abstract concepts.

Figure 2.17: The visual world forms a spatial hierarchy of visual modules: Hyperlocal edges combine into local such as eyes or ears, which combine int high-level concepts such as "cat."

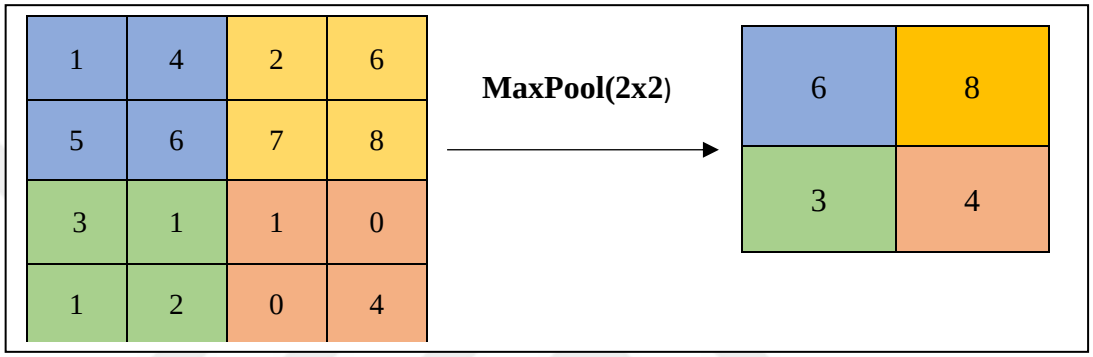


Source: Chollet, F., 2018, *Deep Learning with Python,* USA: Shelter Island.

b) Max-Pooling

Pooling layers introduce two additional hyperparameters: n_k and stride s. The pooling operation's main job is to reduce the dimensions of the feature maps aggressively, and that is done by outputting the max value of each channel in the input feature maps from the extracting windows.

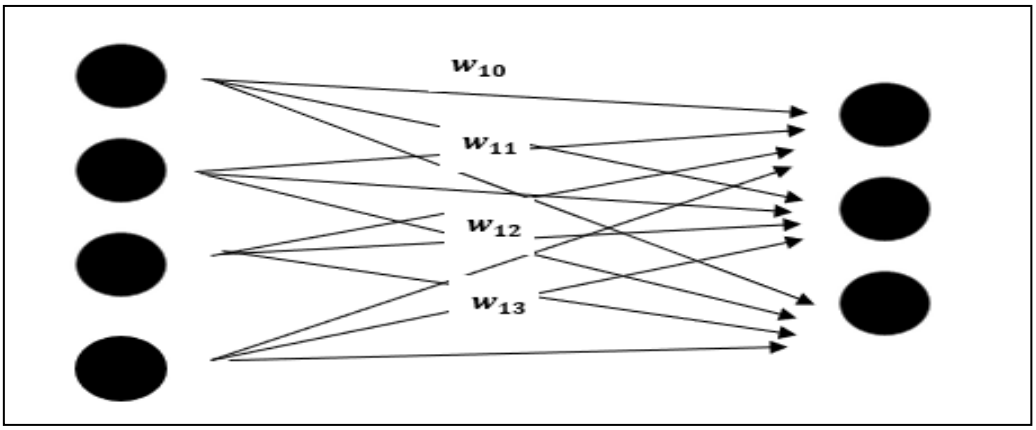
Figure 2.18: Max pooling with 2X2 filters and stride 2.



c) Fully Connected Layer

The fully connected layers are used after the convolution base (convolution and pooling layers) to obtain a score for each possible prediction of the object class by connecting all the features in an image. Each neuron is defined by a weight, which is a parameter that will be learned by the system.

Figure 2.19: Fully connected layers.



d) Classifiers

At the end of the network, a classifier is needed to classify the input and determine the system's output. There are two standard classifiers: The Support Vector Machine and the Softmax classifier.

2.2.12 Object Detection

Image classification methods are used to output what class we have in the single image (e.g., 'cat' or 'dog'). Therefore, it cannot be used as a stand-alone system to solve real-life problems regarding image processing. For example, many scenarios required us to identify the number, type, and location of objects in a single image. The term ' Object Detection is used to define methods where we can determine the instance of multiple objects with their classes by drawing bounding boxes around them.

- Intersect Over Union (IoU)

  IOU is a formula used as a metric analysis method to evaluate the prediction of the network by analyzing the overlap between the bounding boxes and the ground truth. For optimistic prediction, the value of IOU must be greater than 0.5.

  Formula:

$$IOU = \frac{Area\ of\ overlap}{Area\ of\ union} \quad \textbf{(2.14)}$$

There are various CNN methods to do object localization or detection, for example, Region-Based CNN (R-CNN), Fast R-CNN, Faster R-CNN, Mask R-CNN, and YOLO.

2.2.13 Image Segmentation

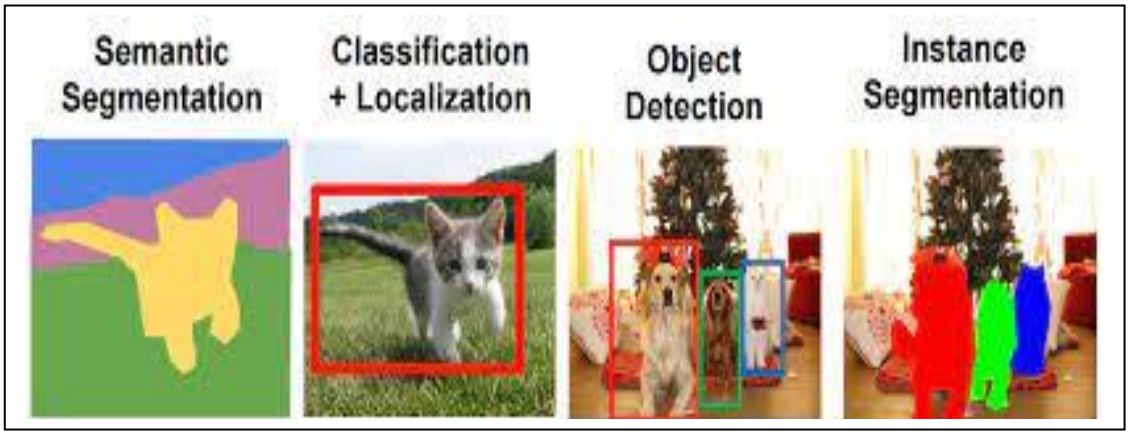
Image segmentation helps us to divide or partition into various parts called segments. By dividing the image into segments, this process helps us process only the regions in the image containing the desired information.

As an image consists of a collection or set of different pixels, image segmentation helps us group pixels with similar attributes. In object detection, we can only localize the object in the image by creating a bounding box around the object for a particular class; this process does not give us any information about the object's shape. On the other hand, image segmentation can create a pixel-wise mask for each object in the image, which gives us a more understanding of the object in the image.

There are two types of image segmentation: Instance segmentation and Semantic segmentation.

a) **Semantic Segmentation:** in this type of segmentation, every pixel that belongs to a particular class is segmented by the same color.
b) **Instance Segmentation:** every pixel that belongs to a particular class is segmented individually by a different color (e.g., object one as red, object two as green, ETC.).

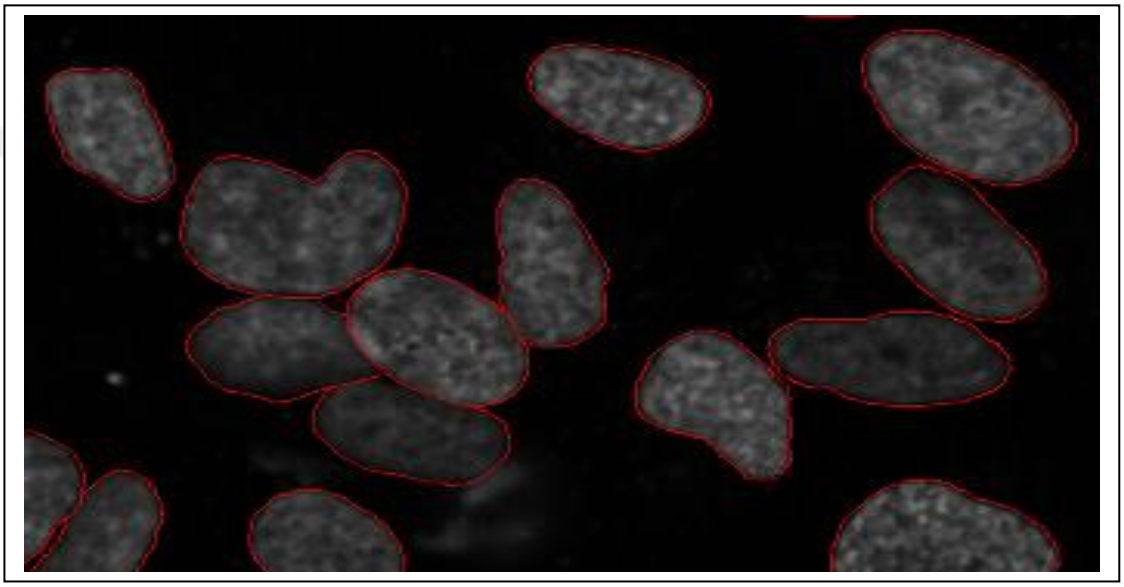
Figure 2.20: A visual explanation of the different terms describing the general task of locating one or more objects in an image.



Source: Michelucci, U., 2019, *Advanced Applied Deep Learning*, Switzerland: Apress.

There are many scenarios in which object detection will not play an important part. For example, in applications like detecting cancerous cell(s), surrounding the cells by bounding boxes will not help us define cancer severity; image segmentation will play a critical part here by defining cancerous cells' shape, see figure 2.16. There are other scenarios where image segmentation is transforming industries (e.g., traffic control system, self-driving cars, and locating objects in satellite images).

Figure 2.21: Cancer cell segmentation.



Source: Wikipedia. [online]

a) Region-based Segmentation

Region-based segmentation is another way to segment different objects based on their pixel values. If there is a sharp contrast between the objects and the image's background, the object's pixel values will be different. Threshold segmentation is a technique where we can set a threshold value, and every pixel values above or below the threshold can be classified accordingly (e.g., " object" or "background").

Global threshold: In which we can divide the image into two regions (e.g., object and background) by setting a single threshold value.

Local threshold: In which we can have multiple objects besides the background by setting multiple thresholds.

b) Edge Detection Segmentation

An edge can split two adjacent regions with different grayscale values, which can be considered the discontinuous local image features. This discontinuity can help detect the edges or the boundaries between two objects: therefore, detecting the shape of these objects in each image.

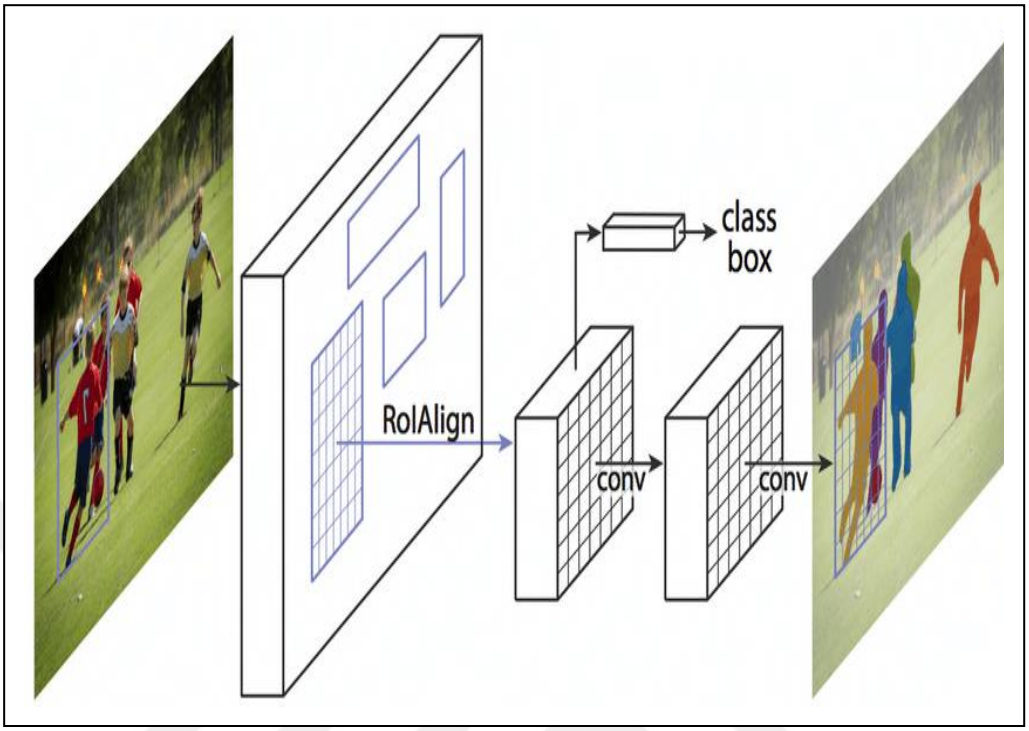
c) Image Segmentation based on clustering.

This method is based on dividing the data points into several groups. These groups have data points that are more similar to other data points in the same group than those in the other groups. K- means is the most popular used clustering algorithm, in which K represents the number of clusters without confused with the k-nearest neighbor.

d) Mask R-CNN

It was introduced by Ross Girshick et al. (2018) as a new model for object detection and object segmentation (not only the localization of the object but also specifying exactly which pixels in the image belong to the object). This model is considered an extension of the "Faster R-CNN" by Ross Girshick et al. (2016) model as it uses the same branch to predict the bounding box, except the only difference is that the last one works in parallel with the new branch predicting an object mask. Mask R-CNN has more advantages on Faster R-CNN regarding the simplicity of training, adds a small overhead, runs at 5 fps, and straightforward generalization to the other tasks, such as the estimation of the human poses within the same framework (Ross Girshick et al. 2018). As we can see in figure 2.14, The concept of the Mask R-CNN model is to build its framework on top of the Faster R-CNN model; therefore, we can predict not only the class label and the bounding box coordinates but also object mask.

Figure 2.22: The mask r-cnn framework for instance segmentation.

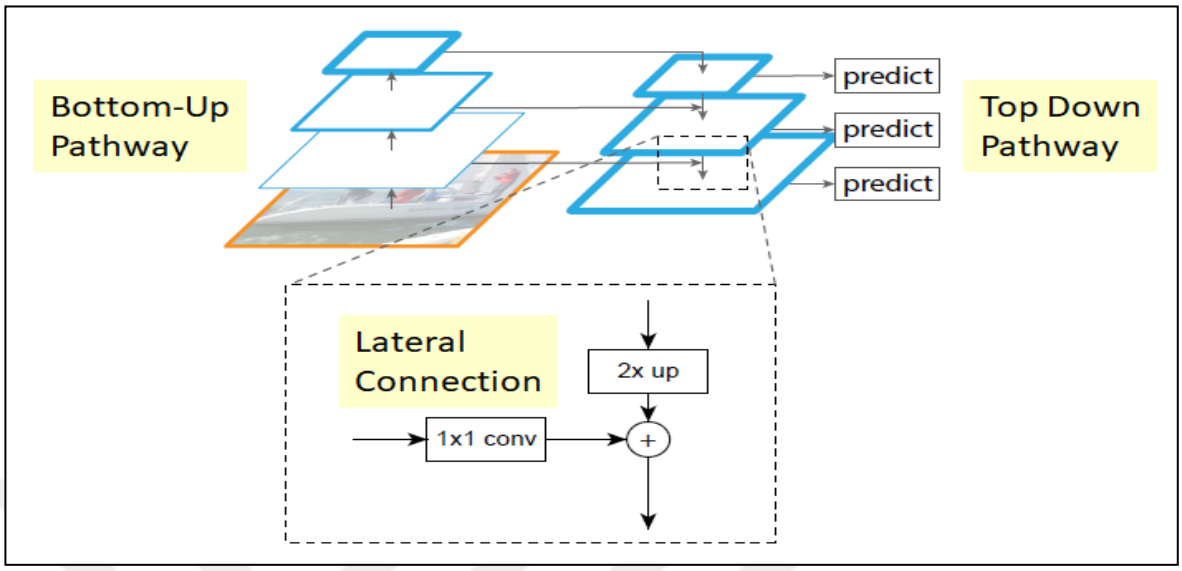


Source: Girshick, R., 2018, *Mask R-CNN*. [online]

e) Feature pyramid network

This architecture was proposed by (Yi Lin et al. 2017) for developing high-level semantic feature maps at all scales. As a generic feature extractor, this model has shown notable advancement in various applications. This model considers as a practical and accurate solution to multi-scale object detection as it can run at 6 FPS on a GPU.

Figure 2.23: Feature pyramid network.

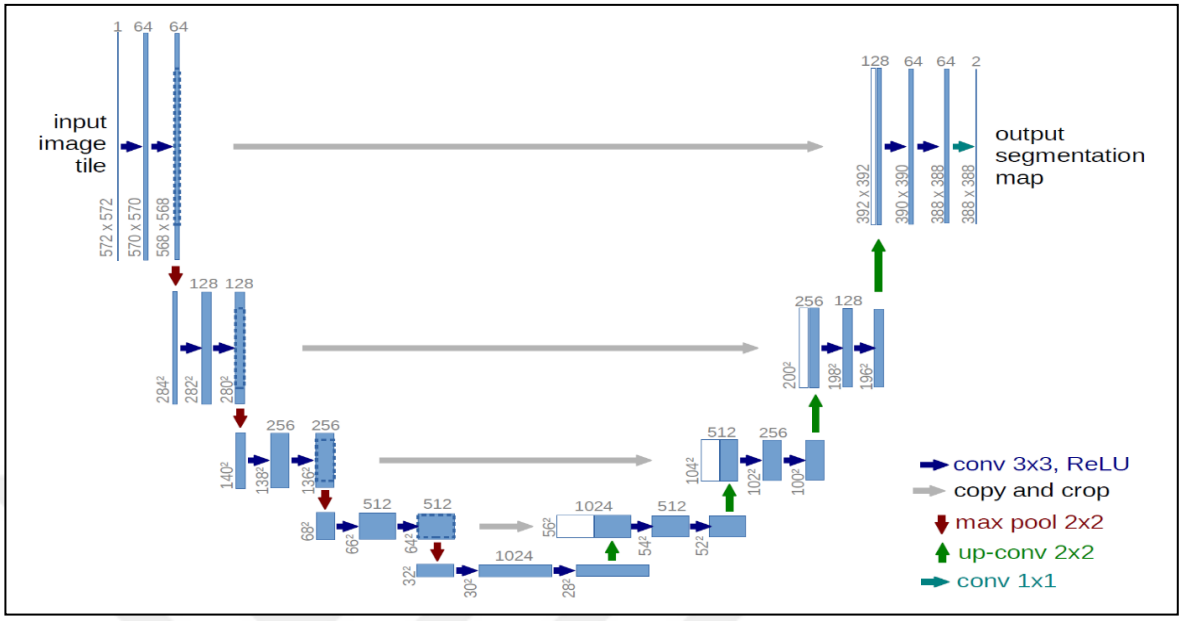


Source: Yi Lin et al. 2017, "*Feature Pyramid Networks for Object Detection*".

f) U-net

This model was introduced by (Ronneberger et al. 2015) for biomedical images segmentation. As we can see in figure 1.1, the model's architecture containing two essential parts which give it the shape of "U," the first part (left) is made for capturing context by a contracting path, and the second part (right) is made for enabling precise localization by a symmetric expanding path. The U-net model has proven that it can be trained end-to-end from significantly fewer images for segmentation of neuronal structures in electron microscopic stacks and betters the sliding-window convolution network on the ISBI challenge. Also, it won the ISBI cell tracking challenge 2015 for transmitted light microscopy images categories. In addition, this model considers fast, and it can perform segmentation in less than a second on a recent GPU on a 512x512 image.

Figure 2.24: U-net architecture.



Source: Ronneberger, O. et. al.,2015, *U-net architecture image segmentation*.

2.2.14 Transfer Learning

Transfer learning is defined as the process of learning from a model that was trained large dataset (e.g., COCO). In computer vision, the features in images are often similar (independent of what objects we want to be detected); therefore, transfer learning is used to reduce the model's trained time if the hyperparameters or weights do need to be modified. This method is popular in image classification or objection detection tasks, especially if the model was trained on similar tasks since we do not have to update the model's trained weights and only train and adjust the head layers of the model (classifier) to detect the new objects. In another scenario, the model can be fine-tuned by freezing the Convbase and training only the head layers (classifier), then unfreezing all or part of Convbase and jointly training both.

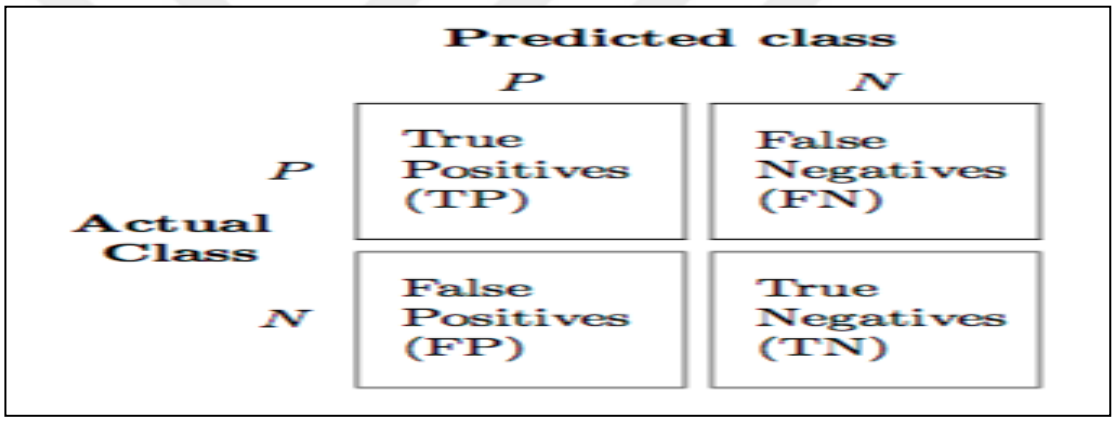
2.2.15 Performance Metrics

Different types of metrics can be used to analyze or measure the model's performance; all of them are derived from the confusion matrix.

e) Confusion Matrix

The confusion matrix is used as a general measurement to summarize the model's prediction performance by sorting them into categories based on whether the prediction is correct or false. Figure 2.16 shows the performance of a binary classification task. There are two scenarios:

Figure 2.25: A confusion matrix for a case with two possible prediction classes.



Source: Jangblad, M., 2018, *Object Detection in Infrared Images using Deep Convolutional Neural Networks.* [online]

a) **True positive & True negative:** when the predicted class and the actual class are the same.

b) **False negatives & False positive:** when the predicted class are not the same as the actual class. (Simple guide to confusion matrix terminology,2014)

f) Precision

This metric's measurement for a class shows how many percent of the predicted class is the actual class. The formula:

$$\frac{True\ Positives}{Predicited\ Positives} \quad \textbf{(2.9)}$$

g) Recall

This metric's measurement for a class shows the percent of the actual class predicted as the correct class. The formula:

$$\frac{True\ Positives}{Actual\ Positives} \quad \textbf{(2.10)}$$

h) Mean Average Precision (mAP)

The mean Average Precision was used with the COCO dataset to evaluate the COCO object detection challenge (COCO,2020). The precision/recall curve, which considers the precision of the recall's function, calculates the average precision for every object class.

The threshold score is altered for detection to get different recall levels; in the case of (threshold=0), the precision would be very low as the recall would be one. The precision is measured, and a precision/recall curve is plotted for different recall levels. The score of the average precision (AP) is measured by the average of this curve. This is then calculated for each object class, and the score of the mean AP is obtained by the mean over the AP for every object class (M.Everingham, L. et al., 2009).

Formula:

$$AP = \frac{1}{N} \sum_{r \in \{0,\ldots,1\}} p(r) \quad \textbf{(2.11)}$$

Where:

N: are the number of recall levels between 0 and 1.

P: is the precision at recall level r.

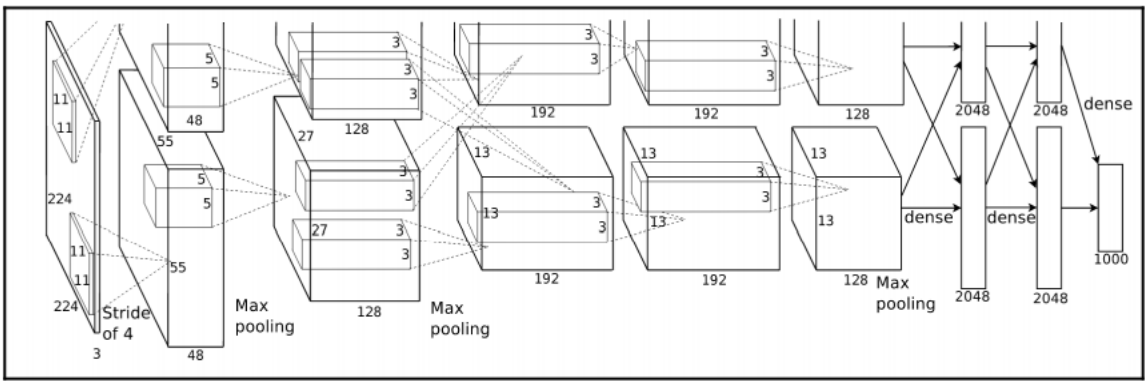
2.3 IMAGE CLASSIFICATION MODELS

This part will give an overview of various deep learning models architectures for image classification that have achieved state-of-the-art results in the ImageNet competitions.

2.3.1 AlexNet model

The alexNet model considers as the first architecture that started a broad interest in deep learning for computer vision (Krizhevsky et al. 2012). When AlexNet proposed, it has been a pioneer and influential in the deep learning field. AlexNet was tested in the ImageNet 2013, and it won with an error rate of about 15.4%, which was significantly better than others. The model consists of simple architecture with five convolution layers. The ImageNet challenge's task was to classify 1,000 categories of objects. The dataset consists of 15 annotated images with more than 22,000 categories; only 1,000 categories are used for competition. The activation that was used for the training by AlexNet was ReLU and found that the results of training are several times faster than other activation functions. It also mentioned, in the paper, that they used data augmentation techniques (e.g., image translations, horizontal flips, and random cropping), also for preventing the overfitting, they also used the dropout layer. During the training, the vanilla Stochastic Gradient Descent (SGD) was chosen. Its parameters are chosen carefully; also, the learning rate's value was changed over a fixed set of training iterations. The weight decay and the momentum take fixed values for the training. In this paper, a new concept was introduced, called Local Response Normalization (LRN). Every pixel across the filters normalized by the LRN layers to avoid massive activation in a particular filter.

Figure 2.26: The AlexNet architecture.

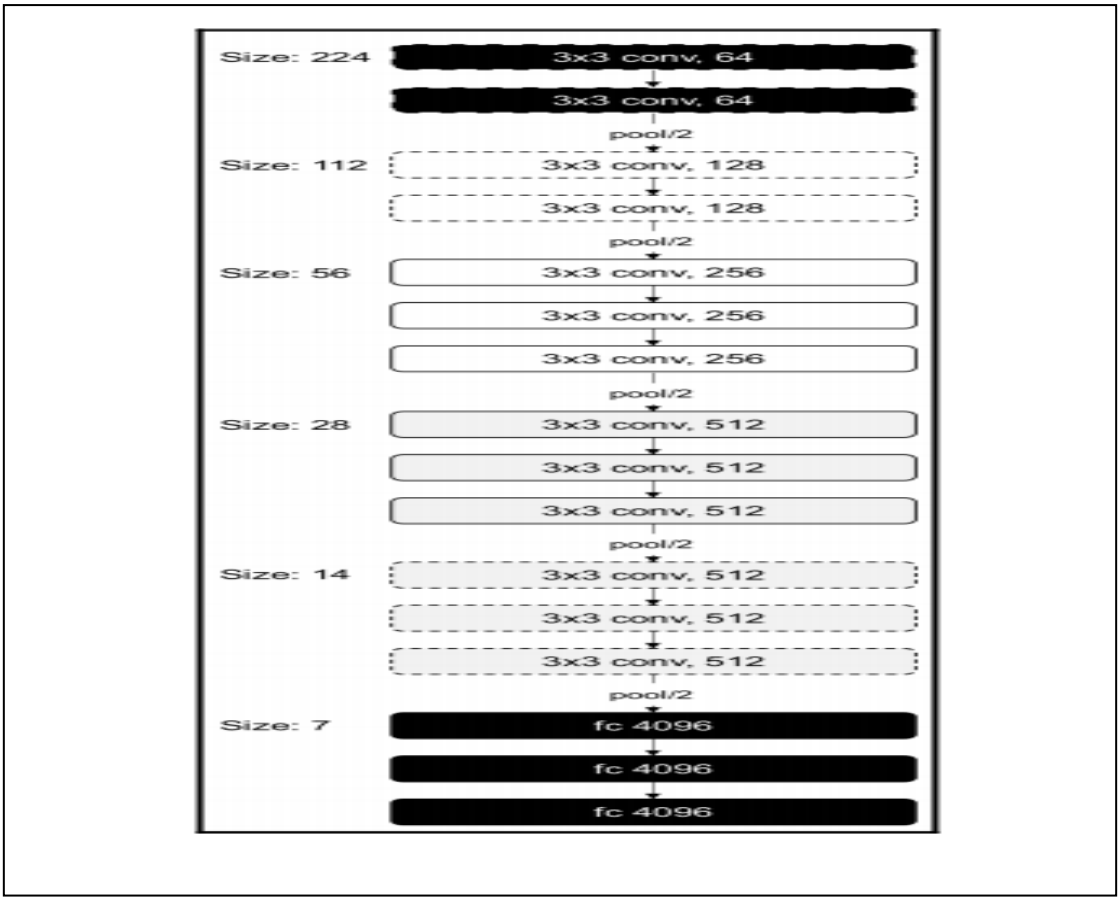


Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

2.3.2 The VGG-16 model

The Visual Geometry Group or VGG is a model created by Oxford, which is very simple and has a greater depth than AlexNet. Two models were introduced in the paper: the first one with 16 layers depth and the second one with 19 layers depth. The convolution layers in the model used 3x3 filters, stride, and pad of size 1, and a max-pooling size of 2 with stride 2 resulted in a decrease in the number of training parameters. Although the size is decreasing because of max pooling, the number of filters increasing with layers. Due to the number of parameters, it has (1338 million parameters), this model considers the largest of all the models described here; its parameters' uniformity is quite good. The most exciting feature of this model is that the smaller image is with an increased number of filters as deep as the network gets. One of the data augmentation techniques that were used in this paper is scale jittering (where a side with a random size is considered to vary the scales) (Shanmugamani 2018).

Figure 2.27: The Architecture of the 16-layer model.

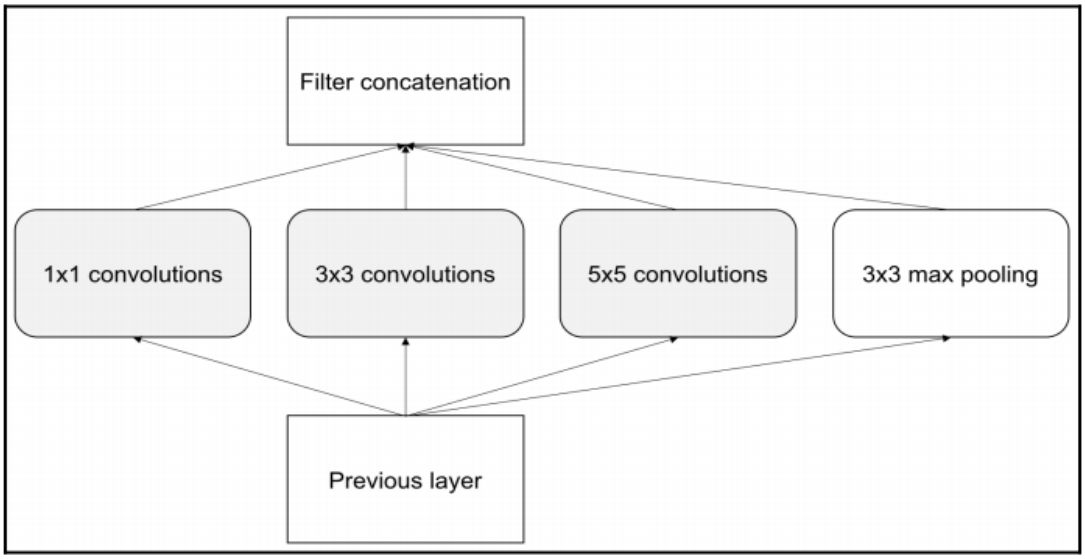


Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

2.3.3 The Google Inception-V3 model

This model was proposed by (Szegedy et al. 2014). This paper introduced the concept of inception that has a better way of generalization. In 2014, Inception-V3 won the ImageNet competition. Inception-V3 is geared towards efficiency for speed and size. This model has 12 times lesser parameters than AlexNet. It is the microarchitecture on which a macro architecture is built. In this model, each hidden layer has a higher-level representation of the image. At each layer, we can select using pooling or other layers. Inception uses several kernels instead of using one type of kernel. Various size convolutions follow an average pooling, and then they are concatenated. In this model, the kernel parameters can be learned based on the data. The model can detect small features as well as higher abstractions by using several kernels. By using 1x1 convolution, the feature is reduced and, hence, computations. This process takes less RAM during inference. This model has nine inception modules with a total of 100 layers, and they achieve good performance (Shanmugamani 2018).

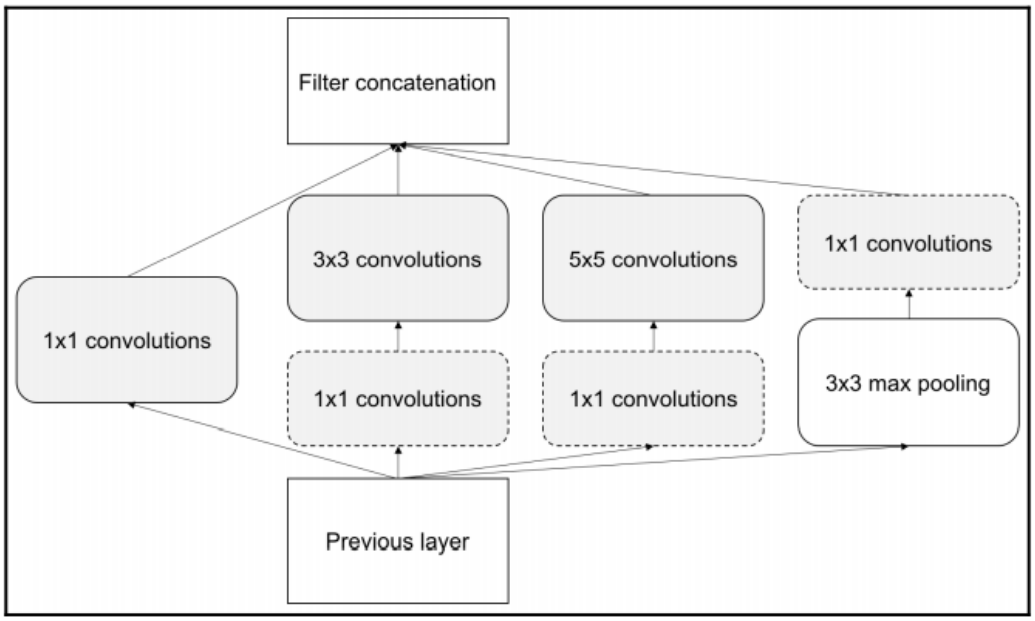
Figure 2.28: The Google Inception-V3 model architecture.



Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

In the opposite of AlexNet and VGG, the operations in this model are happening in parallel. Also, the 1x1 filters are used to reduce the dimension of the output volume. In the opposite of AlexNet and VGG, the operations in this model are happening in parallel. Also, the 1x1 filters are used to reduce the dimension of the output volume.

Figure 2.29: The Google Inception-V3 model architecture after adding the reduced dimensions.

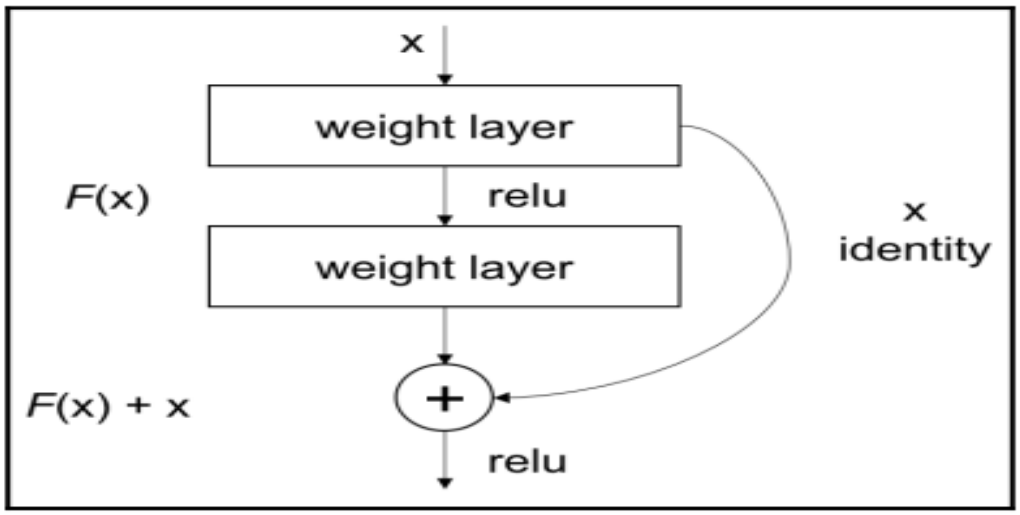


Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision,* UK: Packet

2.3.4 The Microsoft ResNet-50 model.

This model was proposed by (He et al. 2015). In 2015 it won the ImageNet competition. ResNet showed that deeper networks can be trained. The idea of this paper is that the deeper the network, the saturated the accuracy becomes. This paper proved that the accuracy can be increased not because of the overfitting or even because of the use of a high number of parameters but because of a reduction in the training error. This is because of the inability to backpropagate the gradients. As we see in figure 2.20, by sending the gradients directly to the deeper layers with a residual block, we can overcome this problem. We can see in figure 2.20, every two layers are connected, forming a residual block. The backpropagation can carry the error to earlier layers by passing the training between the layers (Shanmugamani 2018).

Figure 2.30: The Microsoft ResNet-50 technique.

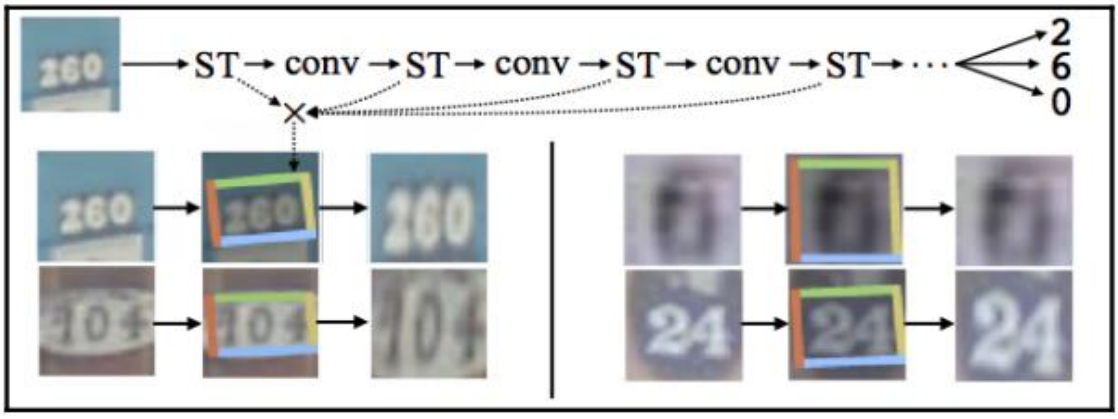


Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

2.3.5 Spatial transformer networks

This model was proposed by (Jaderberg et al. 2016). This model tried to transform the image before passing it to a convolution neural network. The difference from the other models is that this model try to modify the image before convolution. The spatial transformed network learns the parameters to transform the image. They are learned for an affine transformation. Spatial invariance can be achieved by applying an affine transformation. The difference from the previous models, spatial invariance was achieved by mas-pooling layers (Shanmugamani 2018).

Figure 2.31: The replacement of the spatial networks.

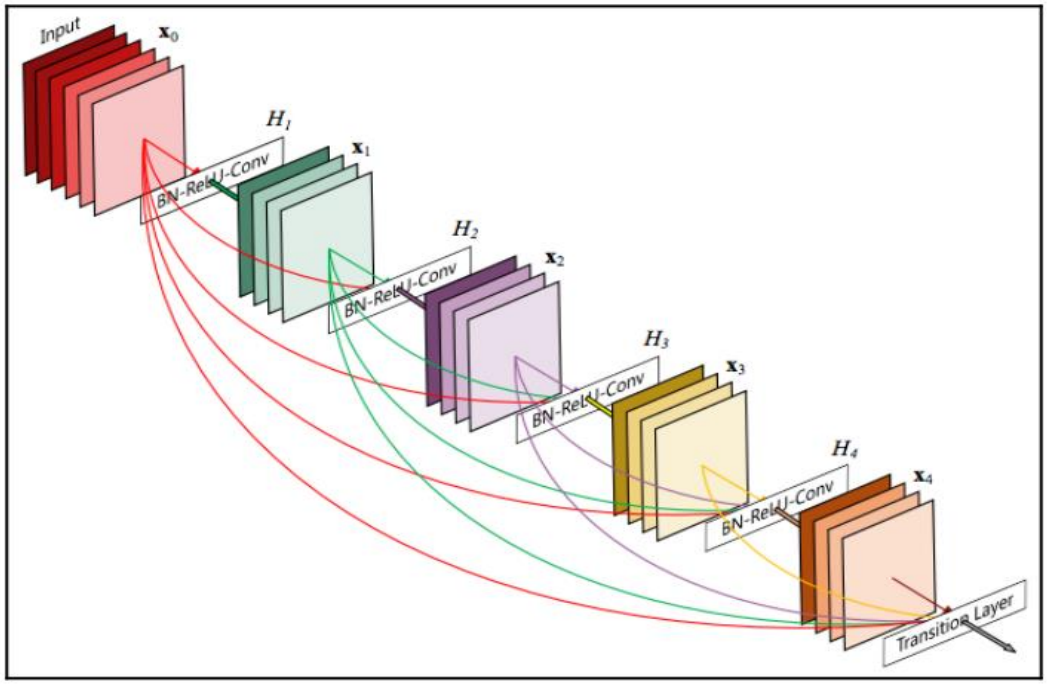


Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

2.3.6 The DenseNet model

This network was proposed by (Huang et al. 2018). The difference between the ResNet model and this model is that in the first one, the previous layer is merged into the future layer by summation. In the second one, the previous layer is merged into the future layer by concatenation. This model connects all the layers to the previous layers and the current layer to the following layers. In figure 2.23, we can see how the feature maps are supplied as input to the other layers.

Figure 2.32: The DenseNet model Architecture.



Source: Shanmugamani, R., 2018, *Deep Learning for Computer Vision*, UK: Packet

As we can see in figure 2.23, there are several advantages: smoother gradients, feature transformation, ETC; and reduces number of parameters.

2.4 INFRARED IMAGES CLASSIFICATION

This section gives an overview of infrared images classification and highlights the importance of research and applications in this field.

A considerable amount of literature has been published on infrared imaging. These studies show the importance of infrared imaging in situations (e.g., dark, and foggy conditions) where visible light cameras are useless for detecting surrounding objects. In these situations, infrared cameras can detect objects by capturing the infrared energy continuously emitting from them. Infrared cameras have been widely used in unmanned cars, drones, and autonomous ships where different targets' temperatures can be detected and converted into grayscale thermal images (Gao et al. 2017).

In recent years, researchers have been increasingly interested in infrared imaging in many applications where it used infrared images to identify input features to extract useful information in their output, for example image fusion (Liu et al. 2018) (Jin, et al. 2017), infrared image colorization, and target detection (Gao et al. 2019) (Wang et al. 2017).

For many years and before adopting deep learning, the researchers manually selected the desired features in applications such as target classification, target detection, and image fusion; for example, in pedestrian detection, the histogram of oriented gradient was widely used (Dollar et al. 2009).

In contrast to infrared images, many researchers focused on using visible light images and features like color, brightness, edge, and texture in feature extraction tasks; for many reasons related to the different imaging principles and low resolution of infrared images. Due to that, these features cannot be considered for infrared images (Fang et al. 2003). In the feature extraction tasks, infrared images are hard to be used as they contain no colors and are more blurred than visible images. In particular, the main necessary feature in infrared images is brightness.

3. DATA AND METHOD

This part presents the data sets chosen for the work and the Deep Learning proposed implementations.

3.1 DATA SETS

In order for the learning system to capture a practical model and obtain excellent results, machine learning and deep learning approaches, in particular, rely strongly on the availability of appropriately large, contextually relevant data sets with suitable variety.
Since deep learning is used for detecting the grasping area (fingers and palms positions), appropriate data sets should be chosen for this task to train, evaluate, and test the model.

The ContactDB dataset was used for this task (Brahmbhatt et al. 2019). It contains 50 household objects grasps with two practical intents by 50 participants. Heat from the hand transfers onto the objects during the grasping of the participants. If the heat does not dissipate rapidly by the object, a thermal camera is used to capture the precise locations of the contact between the human hand and the object. The pixel's intensity in the thermal image can be considered as a function of the infrared energy emitted by the corresponding world point.

As our method is based on data-driven, we used human hands' contact patterns on the objects (fingers and palms locations) as references for stable grasps. Since we are interested in a particular pose of a humane hand, we focused on using particular objects where the thermal camera clearly captured the fingers and the palms' locations. The object we have chosen are cup, bottle, wineglass, bowel, and mug. In figure 3.1, we can see the five objects that we chose. We also included three primitive objects from the dataset to increase our model's efficiency during the training. The objects are cube, pyramid, and sphere.

Figure 3.1: Cup, bottle, wineglass, bowel, and mug.

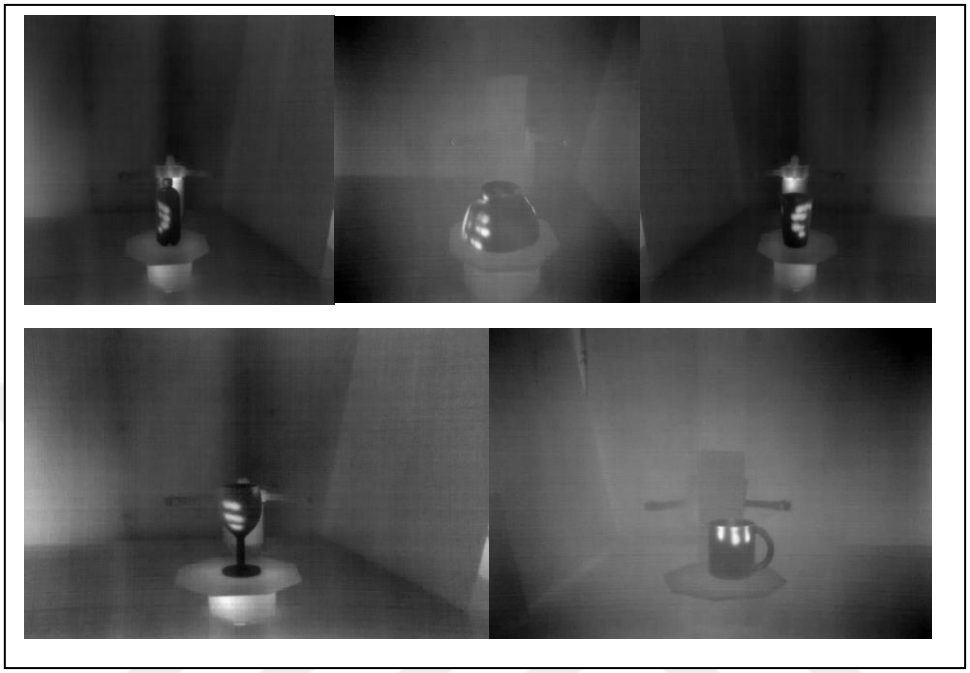


Figure 3.2: Cube, sphere, and pyramid.

3.2 GRASP LABELING

The data set is labeled using a particular tool called *LabelImag* (Russellet al. 2013). As we see in figure 3.1, we created bounding boxes surrounding the fingers and palm's locations and defined it as " Grasping Area.". Our data set contains 6037 images, each image in the data set was labeled manually.

Figure 3.3: An example of labeling the fingers and the palm locations in the image containing the object "Bottle."

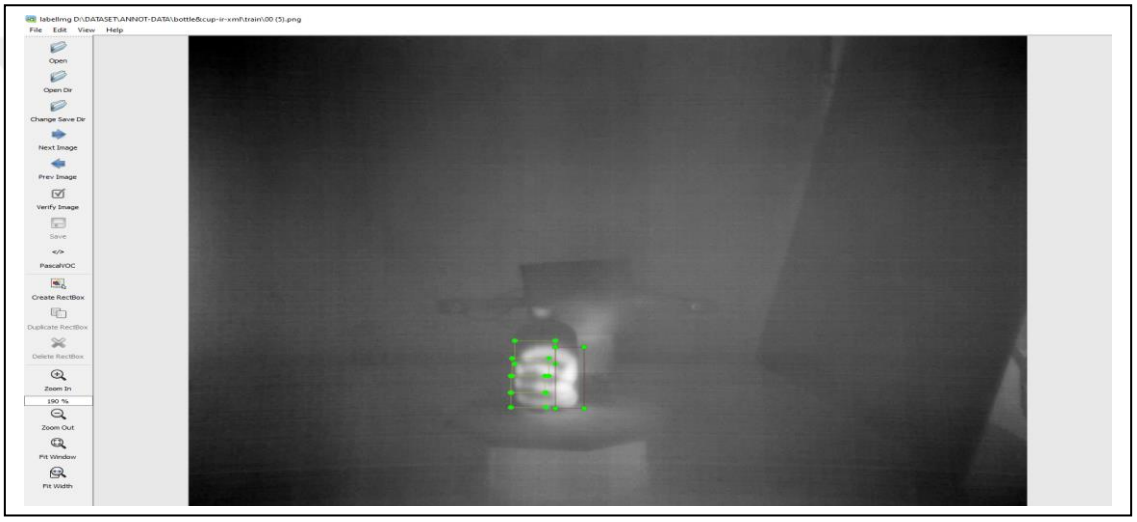


Figure 3.4: An example of labeling the fingers and the palm locations in the image containing the object "Cup."

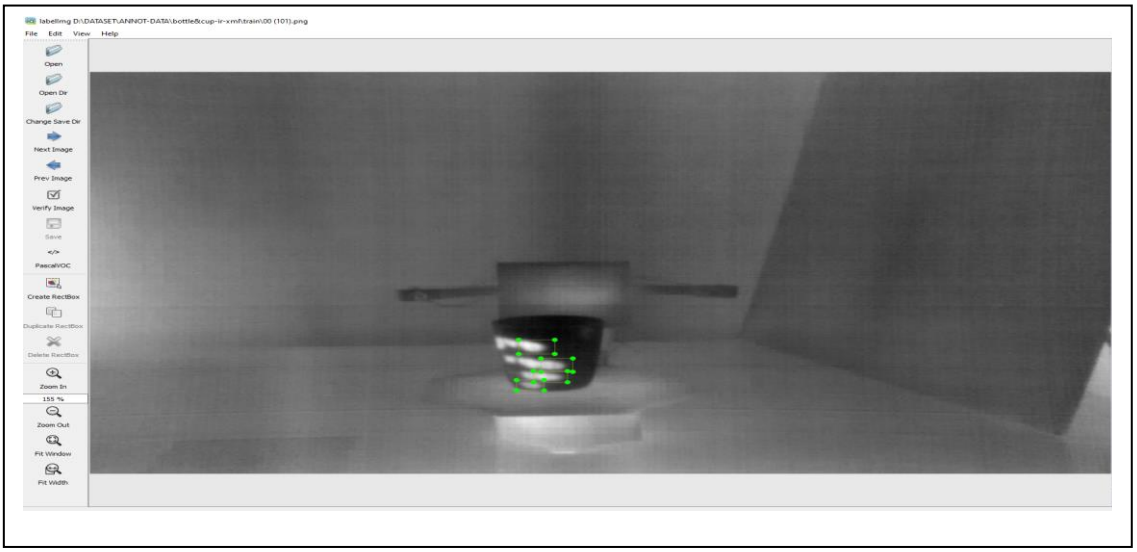


Figure 3.5: An example of labeling the fingers and the palm locations in the image containing the object "Bowel."

Figure 3.6: An example of labeling the fingers and the palm locations in the image containing the object "Wineglass."

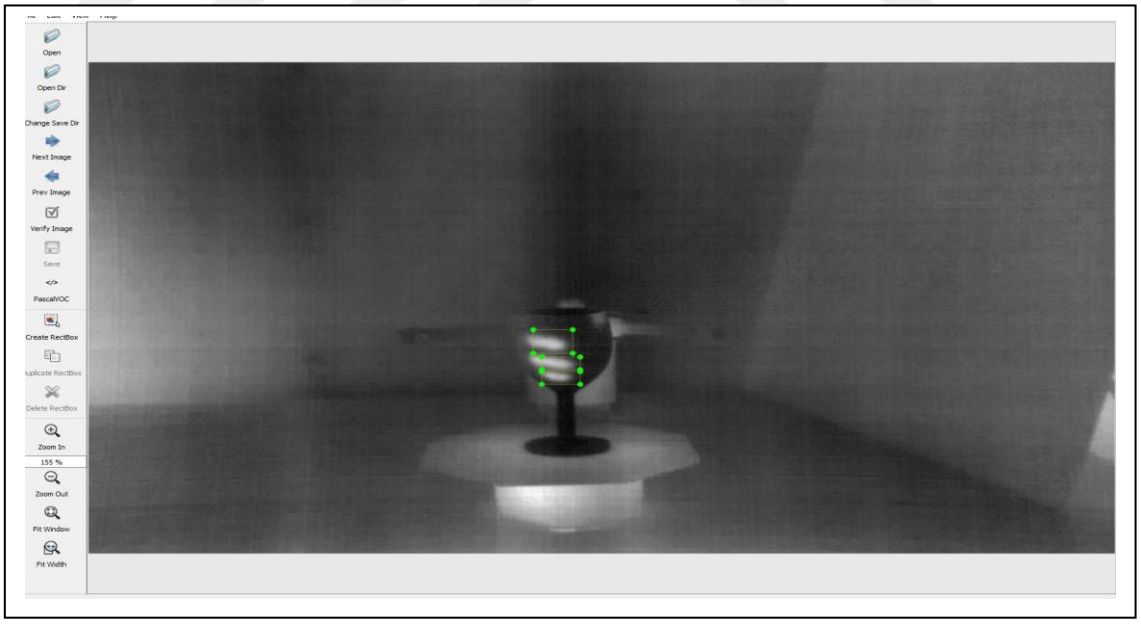


Figure 3.7: An example of labeling the fingers and the palm locations in the image containing the object "Mug."

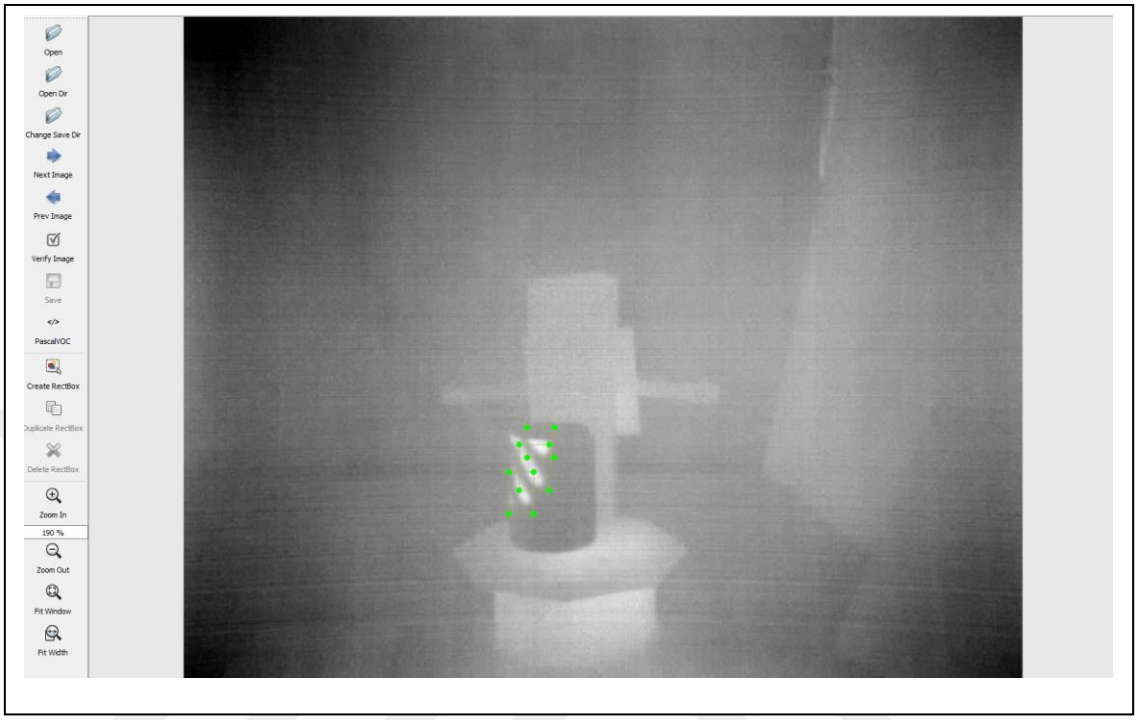


Figure 3.8: An example of labeling the fingers and the palm locations in the image containing the object "Cub."

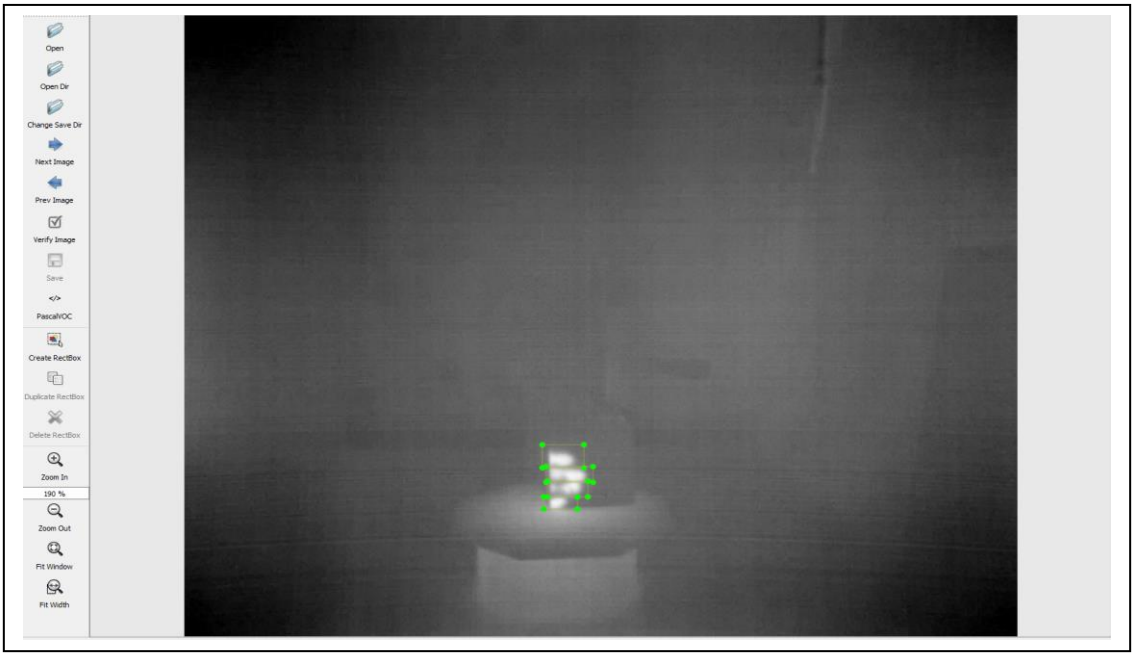


Figure 3.9: An example of labeling the fingers and the palm locations in the image containing the object "Pyramid."

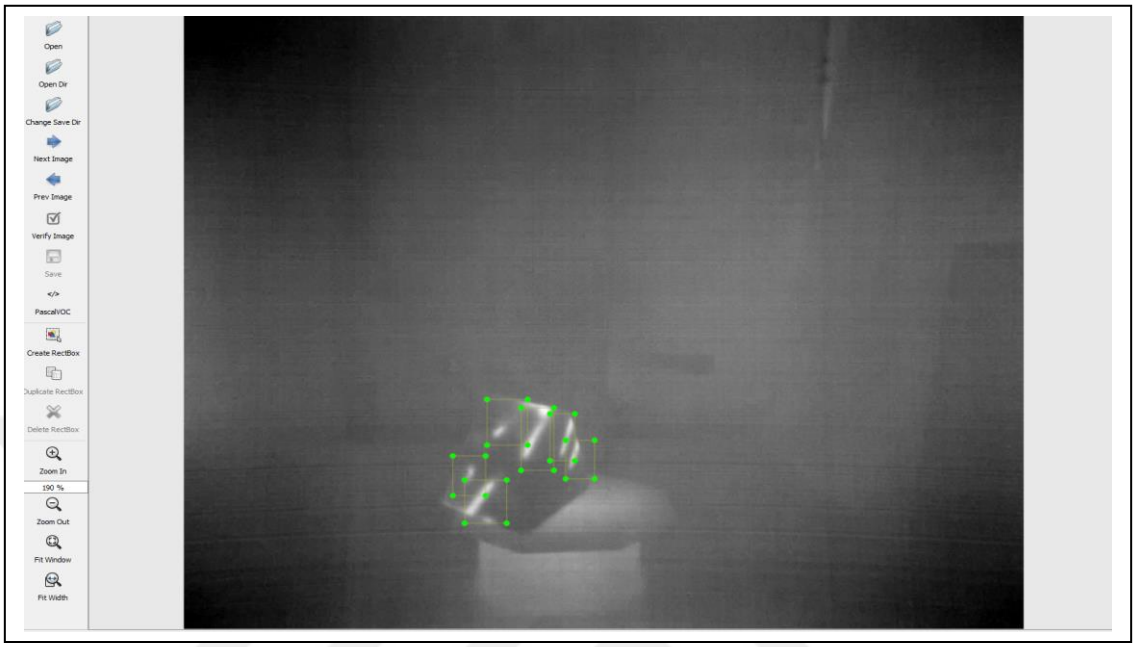
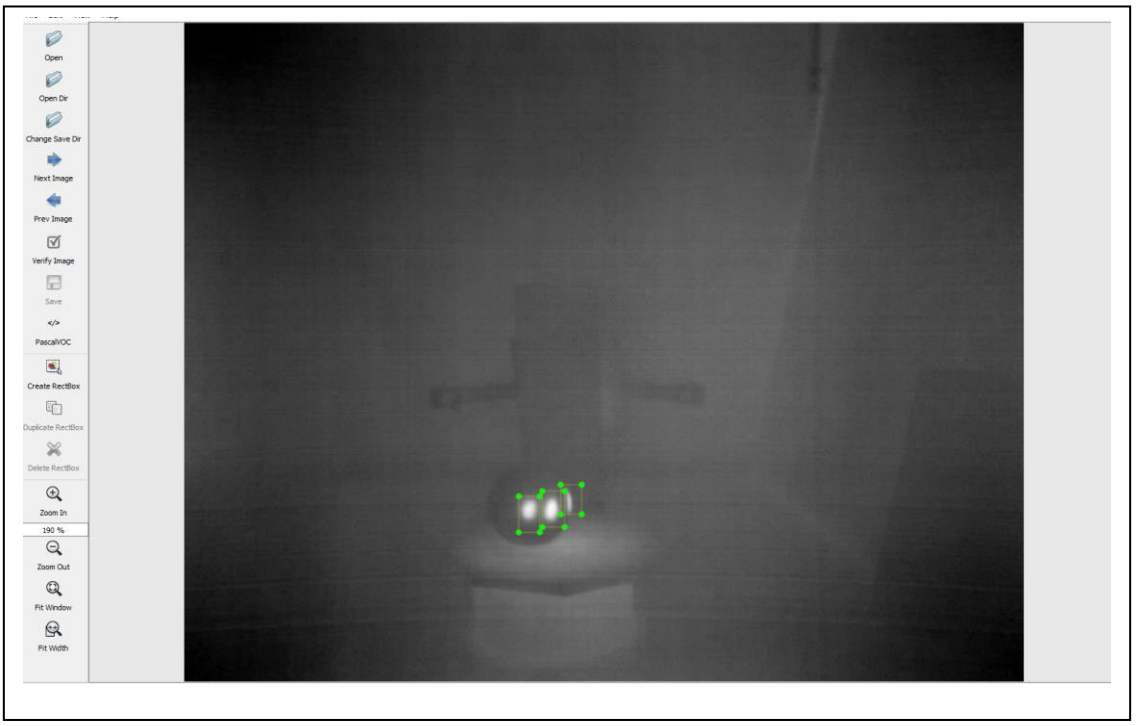


Figure 3.10: An example of labeling the fingers and the palm locations in the image containing the object " Sphere."



3.3 SUPERVISD LEARNING IMPLEMENTATION OF PRETRAINED MASK R-CNN

As our dataset size is limited, we chose to train only the classifiers of the model "Heads" instead of changing the features in pre-trained weights. We also used *mask_rcnn_coco.h5* as weight initialization. Using Mask R-CNN pre-trained model, we were able to use the annotation files containing bounding boxes to create masks for objects in the images and to implementing object detection and instance segmentation on the data set. The input images were in size 512 x 640.

Table 3.1: Pretrained layers "Heads" for mask r-cnn.

Layer (type)	Output Shape	Param #	Connected to
Input_image (Input Layer)	(None, None, None,2)	0	*
zero_padding2d_1 (ZeroPadding2D)	(None, None, None,2)	0	input_image[0][0]
fpn_c5p5 (Conv2D)	(None, None, None, 2)	524544	res5c_out[0][0]
fpn_c4p4 (Conv2D)	(None, None, None, 2)	262400	res4w_out[0][0]
fpn_c3p3 (Conv2D)	(None, None, None, 2)	131328	res3d_out[0][0]
fpn_c2p2 (Conv2D)	(None, None, None, 2)	65792	res2c_out[0][0]
fpn_p5 (Conv2D)	(None, None, None, 2)	590080	fpn_c5p5[0][0]
fpn_p2 (Conv2D)	(None, None, None, 2)	590080	fpn_p2add[0][0]

fpn_p3 (Conv2D)	(None, None, None, 2)	590080	fpn_p3add[0][0]
fpn_p4 (Conv2D)	(None, None, None, 2)	590080	fpn_p4add[0][0]
rpn_model (Model) : rpn_conv_shared (Conv2D) rpn_class_raw (Conv2D) rpn_bbox_pred (Conv2D)	[(None, None, 2), (N)]	1189394	fpn_p2[0][0] fpn_p3[0][0] fpn_p4[0][0] fpn_p5[0][0] fpn_p6[0][0]
mrcnn_mask_conv1 (TimeDistributed)	(None, 200, 14, 14)	590080	roi_align_mask[0][0]
mrcnn_mask_bn1 (TimeDistributed)	(None, 200, 14, 14)	1024	mrcnn_mask_conv1[0][0]
mrcnn_mask_conv2 (TimeDistributed)	(None, 200, 14, 14)	590080	activation_71[0][0]
mrcnn_mask_bn2 (TimeDistributed)	(None, 200, 14, 14)	1024	mrcnn_mask_conv2[0][0]
mrcnn_class_conv1 (TimeDistributed)	(None, 200, 1, 1, 10)	12846080	roi_align_classifier[0][0]
mrcnn_class_bn1 (TimeDistributed)	(None, 200, 1, 1, 10)	4096	mrcnn_class_conv1[0][0]
mrcnn_mask_conv3 (TimeDistributed)	(None, 200, 14, 14)	590080	activation_72[0][0]
mrcnn_mask_bn3 (TimeDistributed)	(None, 200, 14, 14)	1024	mrcnn_mask_conv3[0][0]
mrcnn_class_conv2 (TimeDistribued)	(None, 200, 1, 1, 10)	1049600	activation_68[0][0]
mrcnn_class_bn2 (TimeDistributed)	(None, 200, 1, 1, 10)	4096	mrcnn_class_conv2[0][0]

mrcnn_mask_conv4 (TimeDistributed)	(None, 200, 14, 14)	590080	activation_73[0][0]
mrcnn_mask_bn4 (TimeDistributed)	(None, 200, 14, 14)	1024	mrcnn_mask_conv4[0][0]
mrcnn_bbox_fc (TimeDistributed)	(None, 200, 8)	8200	pool_squeeze[0][0]
mrcnn_mask_deconv (TimeDistributed)	(None, 200, 28, 28)	262400	activation_74[0][0]
mrcnn_class_logits (TimeDistributed)	(None, 200, 2)	2050	pool_squeeze[0][0]
mrcnn_mask (TimeDistributed)	(None, 200, 28, 28)	514	mrcnn_mask_deconv[0][0]

3.4 SUPERVISD LEARNING IMPLEMENTATION OF PRETRAINED U-NET

After we extracted masks from our annotation files, we used a pre-trained U-net model with "ImageNet" weights and "ResNet34" as a backbone of the model to perform the decoding and upsampling part of the training. For the training metrics, we used The Jaccard distance to measure the intersection over the union (IoU) as our loss function and the dice coefficient function to measure the precision of the prediction during the training. We also modified the first input layer for non-RGB images.

Table 3.2: U-net model with ResNet34 as a backbone.

Layer (type)	**Output Shape**	**Param #**	**Connected to**
input_1(Input layer)	(None, 256, 256, 1)	0	*
conv2d_1(Conv2D)	(None, 256, 256, 3)	6	input_1[0][0]

Input_2(Input layer)	(None, 256, 256,3)	0	Conv2d_1[0][0]
bn_data(BatchNormalization)	(None, 256, 256,3)	9	data[0][0]
zero_padding2d_1(ZeroPadding2D)	(None, 256, 256,3)	0	bn_data[0][0]
conv0(Conv2D)	(None, 256, 256,3)	96408	zero_padding2d_1[0][0]
Continu.			
decoder_stage0_upsampling (UpSa)	(None, 256, 256,5)	0	relu1[0][0]
decoder_stage0_concat (Concaten)	None, 256, 256,7)	0	decoder_stage_upsampling[0][0]
Continu.			
final_conv (Conv2D)	None, 256, 256,1)	145	decoder_stage4b_relu[0][0]
sigmoid (Activation)	None, 256, 256,1)	0	final_conv[0][0]

3.5 SUPERVISD LEARNING IMPLEMENTATION OF PRETRAINED FPN

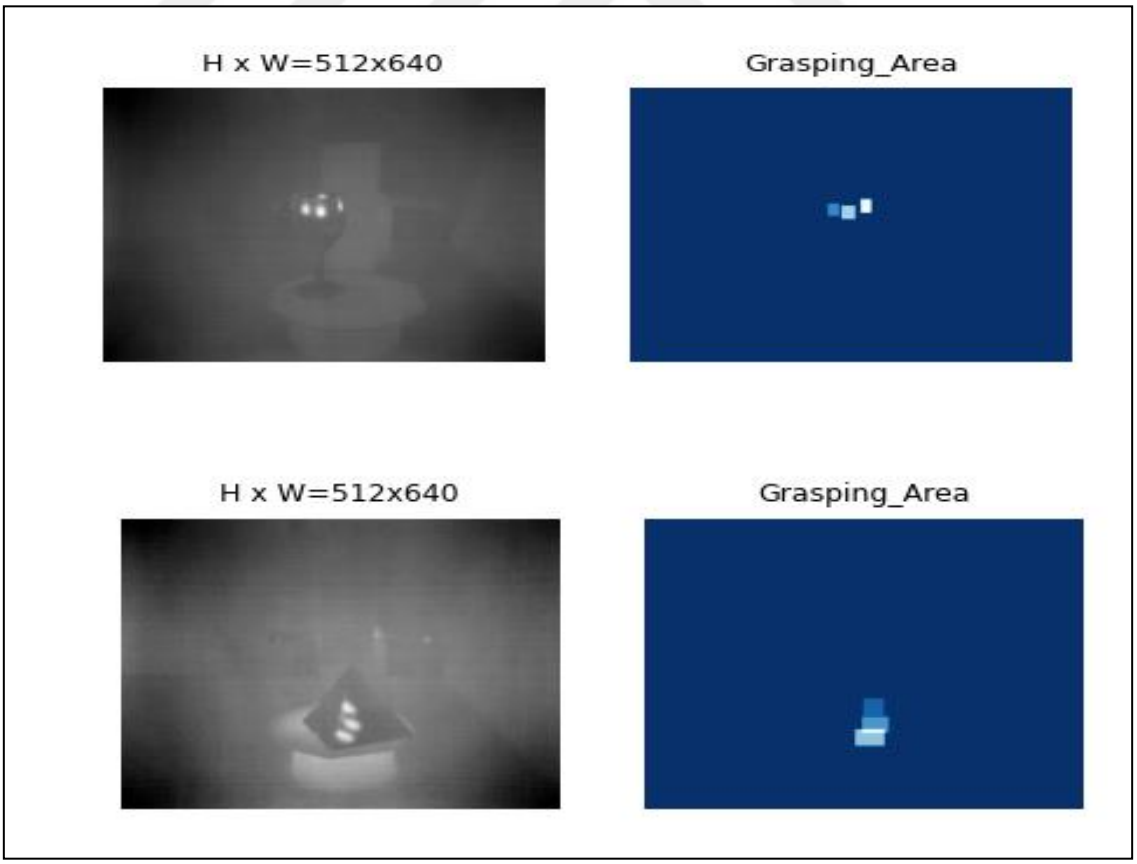
In this pre-trained model, we used the same initializing weights, loss function, accuracy function that we used in U-net to train our dataset. For the model backbone, we used " Resnet50 ", also the first layer had to be modified for non-RGB images.

Table 3.3: FPN model with ResNet50 as a backbone.

Layer (type)	**Output Shape**	**Param #**	**Connected to**
input_1(Input layer)	(None, 256, 256, 1)	0	*
conv2d_1(Conv2D)	(None, 256, 256, 3)	6	input_1 [0][0]
Input_2(Input layer)	(None, 256, 256,3)	0	Conv2d_1 [0][0]
bn_data(BatchNormalization)	(None, 256, 256,3)	9	Data [0][0]
zero_padding2d_1(ZeroPadding2D)	(None, 256, 256,3)	0	bn_data [0][0]
conv0(Conv2D)	(None, 256, 256,3)	96408	zero_padding2d_1 [0][0]
bn0 (BatchNormalization)	(None, 256, 256,6)	256	conv0[0][0]
relu0 (Activation)	(None, 256, 256,6)	0	bn0[0][0]
Continu.			
pyramid_dropout (SpatialDropout)	(None, 256, 256,5)	0	aggregation_concat [0][0]

final_stage_conv (Conv2D)	(None, 256, 256,1)	589824	pyramid_dropout[0][0]
final_stage_bn (BatchNormalizat)	(None, 256, 256,1)	512	final_stage_conv[0][0]
final_stage_relu (Activation)	(None, 256, 256,1)	0	final_stage_bn[0][0]
final_upsampling(UpSampling2D)	(None, 256, 256,1)	0	final_stage_relu[0][0]
head_conv (Conv2D)	(None, 256, 256,1)	1153	final_upsampling[0][0]
sigmoid (Activation)	(None, 256, 256,1)	0	head_conv[0][0]

Figure 3.11: Example of visualizing data set information and mask.



4. EXPERIMENTAL RESUTLS

This part presents the hardware system available to implement and run the proposed architectures and the results obtained from the execution of the implementations.

4.1 HARDWARE AND SOFTWARE

The hardware system used for thesis work is a desktop build with intel core i7 10700, 32GB of system RAM, Nvidia RTX 2070 Ventus GPU with 8GB of RAM, Cuda toolkit version 10.1.243 with cudnn version 10.1, and the operating system used is WINDOWS 10.

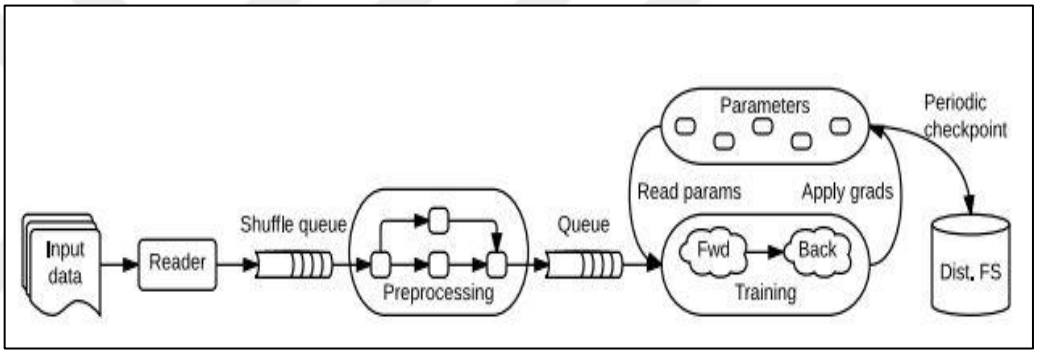
The programming platform chosen to perform the deep learning implementation is Google TensorFlow with Keras API in the Python programming language.

i) **Python programming language:** Python can be defined as an interpreted, object-oriented, high-level programming language. It is the simple, easy-to-learn syntax that emphasizes readability which reduces the cost of program maintenance. It is the de-facto choice for the deep learning application. Python has the largest community and support ecosystem of libraries. Python is the natural language of choice since TensorFlow API for python is the most complete. Python version used is computed using Anaconda package management environment 3.7.0.

j) **TensorFlow:** It is an open-source library for the development and deployment of deep learning. It implements data flow and numerical computations by using computational graphs. It is suitable for deep learning since the graph has nodes that enable any numerical computation. TensorFlow provides a single API for all kinds of platforms and hardware. ALL the complexity of scaling and optimization can be handled at the backend by TensorFlow. Also, it comes with tools for visualization and deployment in production. The TensorFlow version we used in this work is 1.15.0.

k) **Keras library:** It is an open-source library for deep learning written in python. Keras provides an easy interface to use TensorFlow as a backend. By focusing on friendliness, modularity, and extensibility, Keras is designed for easy and fast experimentation. It can run seamlessly between CPU and GPU, and it is a self-contained framework. It can be used separately or used TensorFlow as a backend by using *tf.keras* API. The Keras version we used in this work is 2.2.4.

Figure 4.1: Dataflow graph schematic of tensorflow



Source: Colyer, A., 2016, The morning paper, *Tensorflow: A system for large-scale machine learning.* [online]

4.2 RESULTS

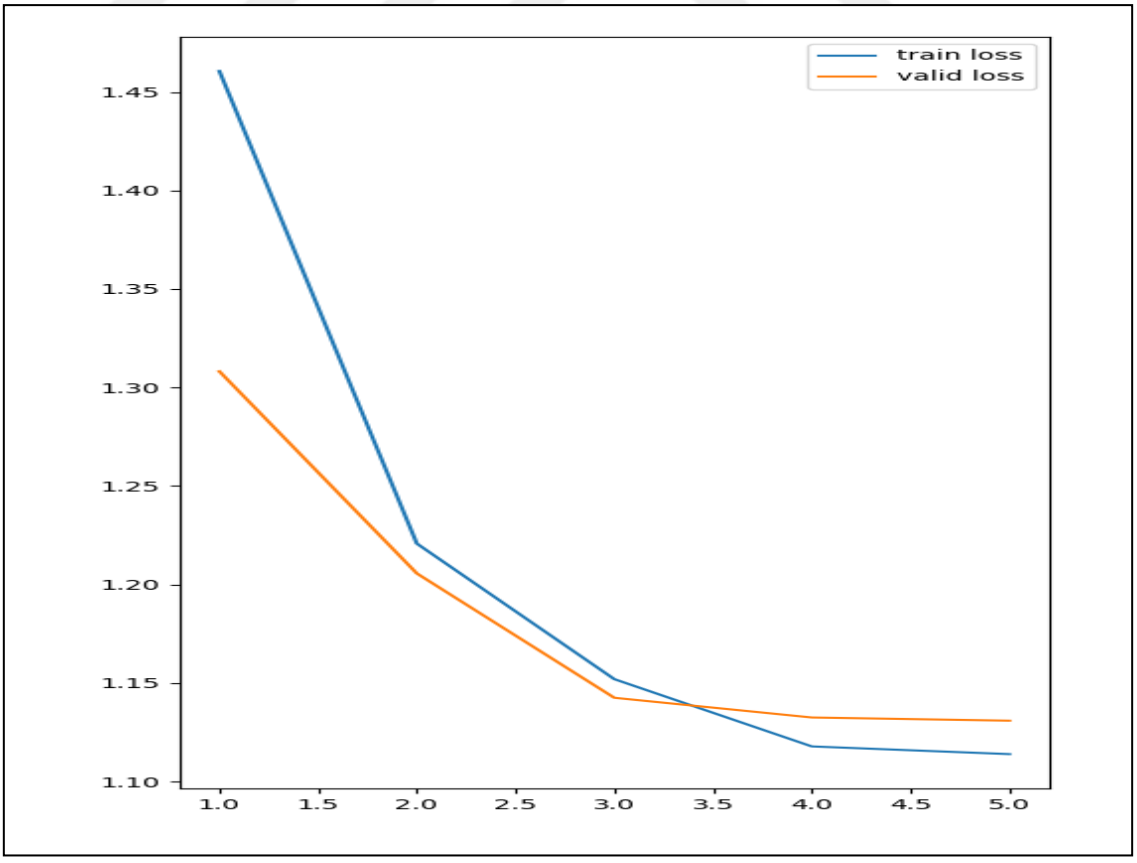
This part presents the obtained results of the proposed method from the training and evaluation with the data set on the hardware system provided. Before the training, we divided the data set into train set (5433 images), validation set (607 images), and test set (168 images). Both the training and the validation sets were training to predict one class ("Grasping_Area").

Data augmentation custom techniques (e.g., horizontal flipping, rotation, transformation, and shearing) were used in our results to overcome the overfitting. The loss and accuracy curves of the training and valuation sets were obtained.

The first pre-trained model that we trained was Mask R-CNN with Resnet101 as a backbone for bounding boxes and instance segmentation prediction with 5 epochs, 2 images per GPU, and 1000 steps per epochs. We also used *mask_rcnn_coco* as weight initialization. The maximum accuracy we have achieved through our experiments was 95% with a 1.110 loss value.

The following figure shows the training and validation loss curves for pre-trained Mask R-CNN.

Figure 4.2: The train and validation loss curves for the Mask R-CNN model.



The following figures show the bounding boxes prediction of the pre-trained Mask R-CNN model for the "Grasping_Area" class.

Figure 4.3: The result of bounding boxes prediction for the object “cup” by Mask R-CNN.

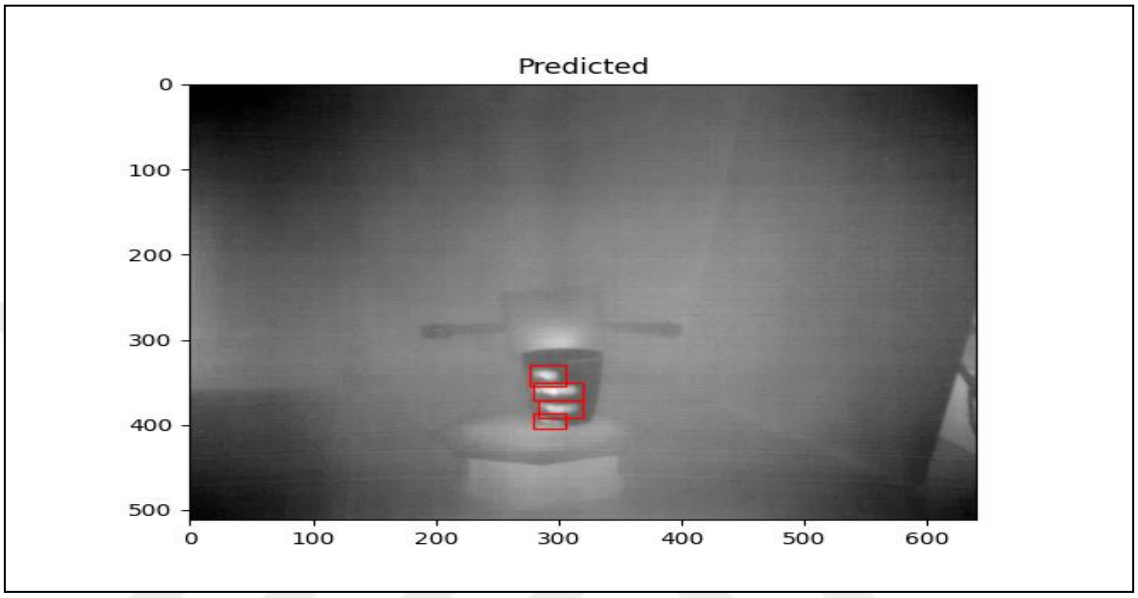


Figure 4.4: The result of bounding boxes prediction for the object “bowel” by Mask R-CNN

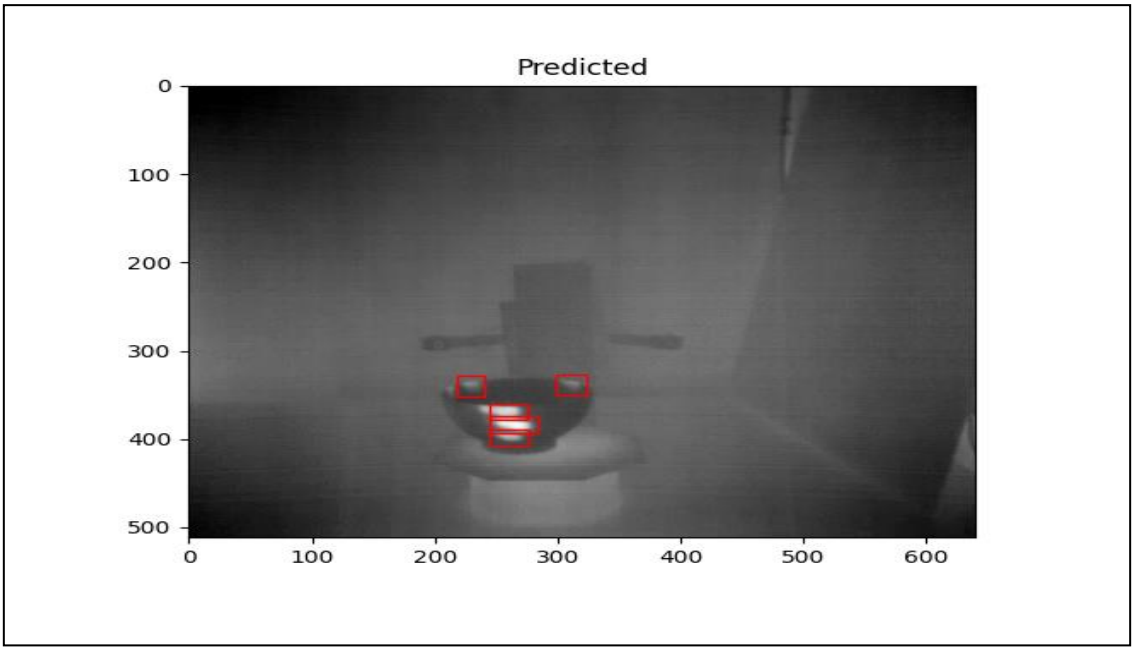
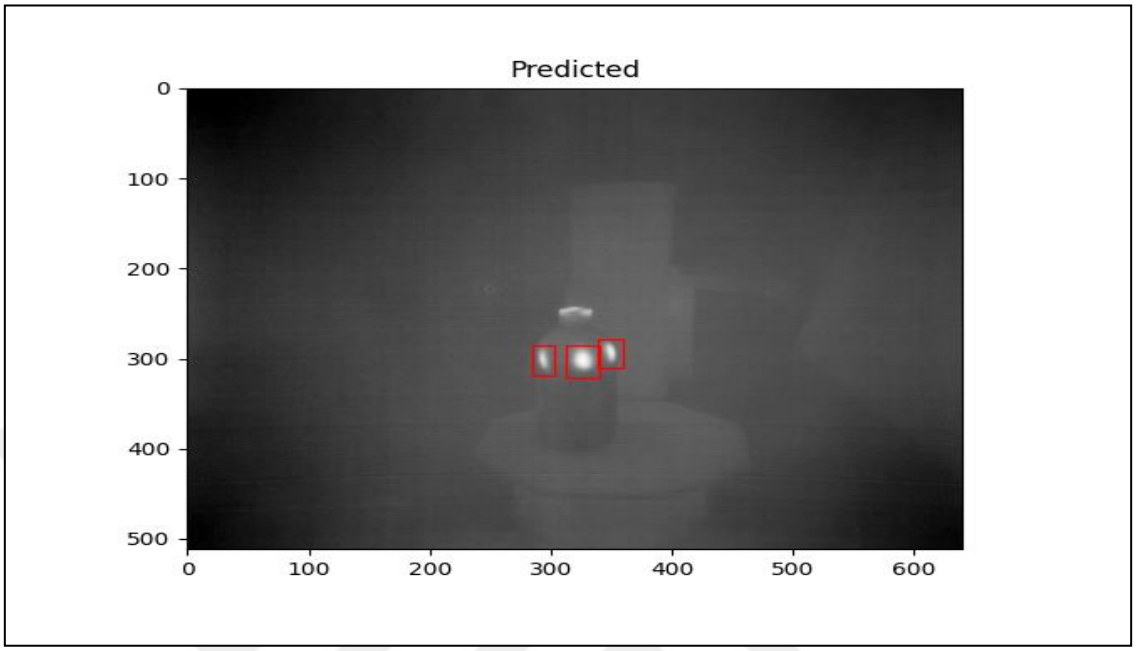


Figure 4.5: The result of bounding boxes prediction for the object “bottle” by Mask R-CNN.



The following figures show the instance segmentation prediction of the pre-trained Mask R-CNN model for the "Grasping_Area" class.

Figure 4.6: The result of instance segmentation for the object “cup” by Mask R-CNN.

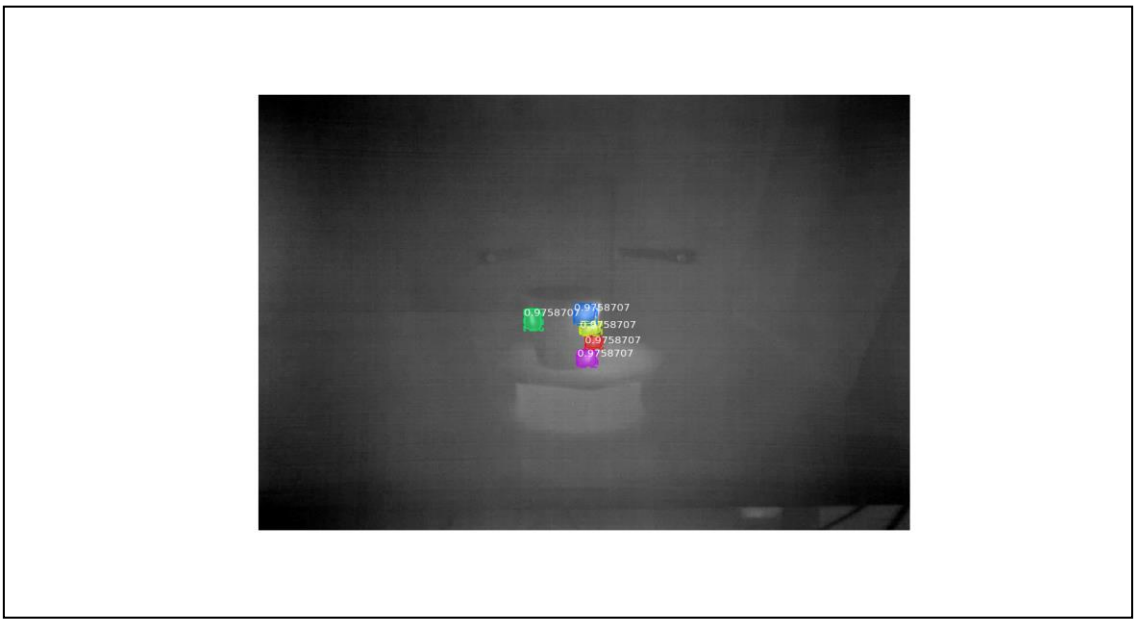


Figure 4.7: The result of instance segmentation for the object “bottle” by Mask R-CNN.

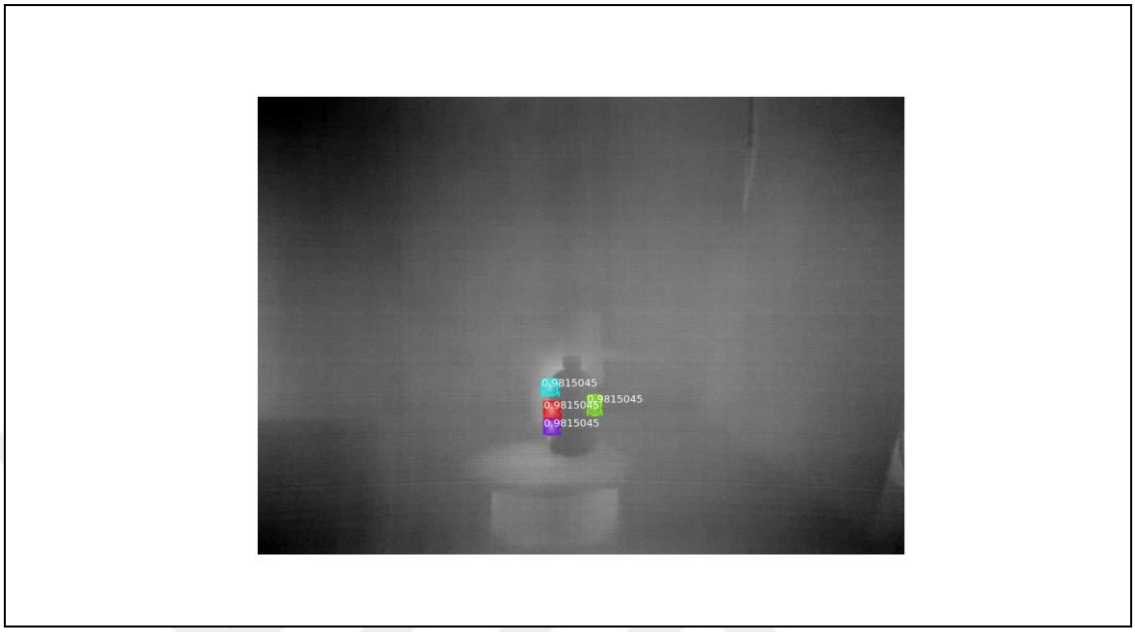


Figure 4.8: The result of instance segmentation for the object “wineglass” by Mask R-CNN.

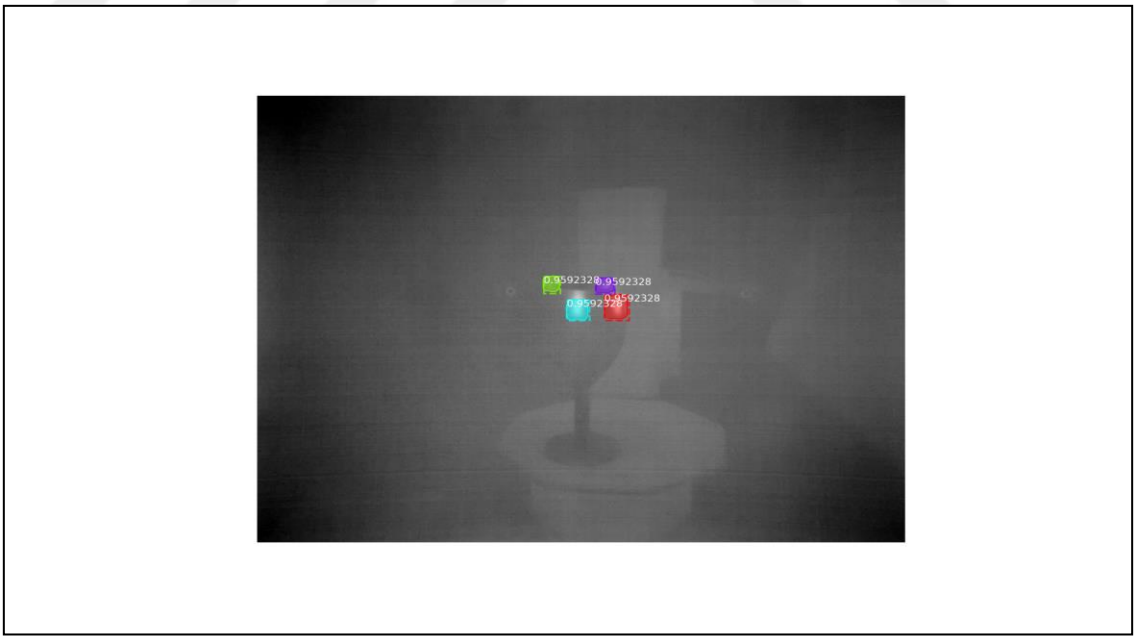


Figure 4.9: The result of instance segmentation for the object “cube” by Mask R-CNN.

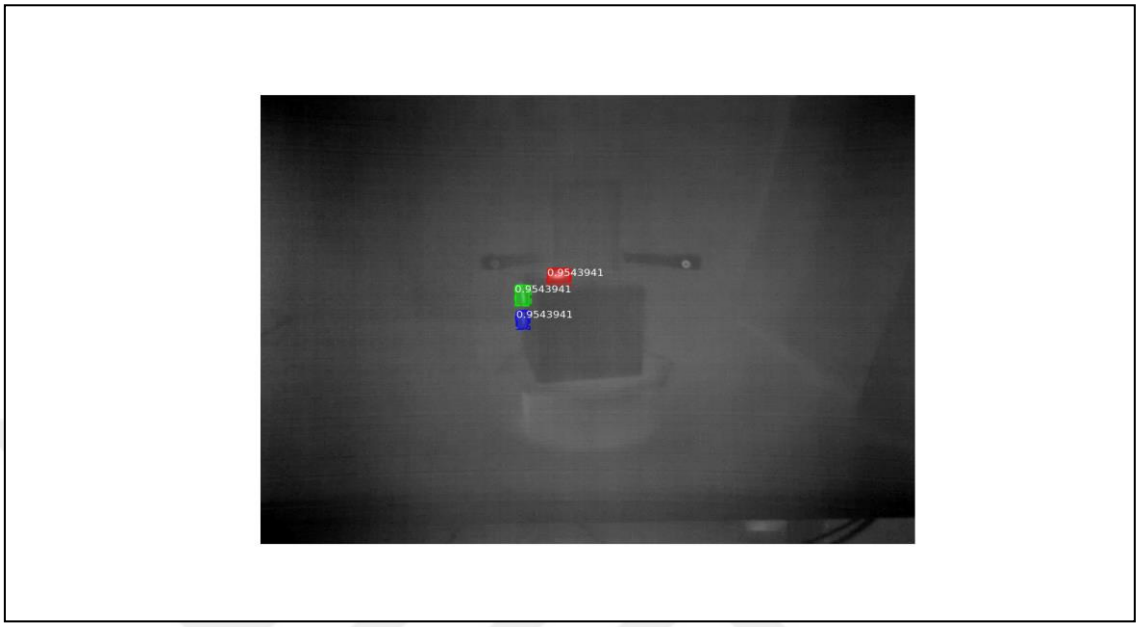
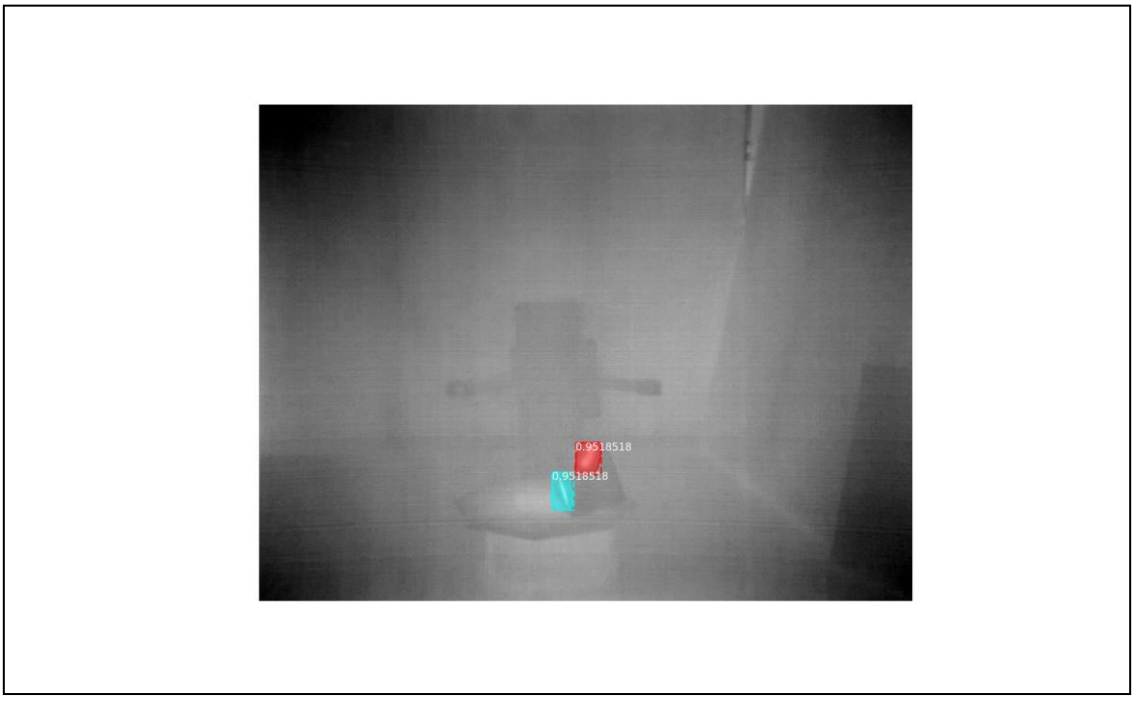


Figure 4.10: The result of instance segmentation for the object “pyramid” by Mask R-CNN.



The second pre-trained model that we trained on our dataset was U-net for semantic segmentation prediction of the “Grasping_Area” class. We used ResNet34 as a backbone of the model to perform the decoding and up sampling part of the training. For the training metrics, we used The Jaccard distance to measure the intersection over the union (IoU) as our loss function and the dice coefficient function to measure the precision of the prediction during the training. We have achieved 88% accuracy with 19.77 loss value. For the weight initialization we used “*ImageNet*”.

The following figure shows the training and validation loss curves for pre-trained U-net.

Figure 4. 11: Training and validation accuracy curves of U-net model.

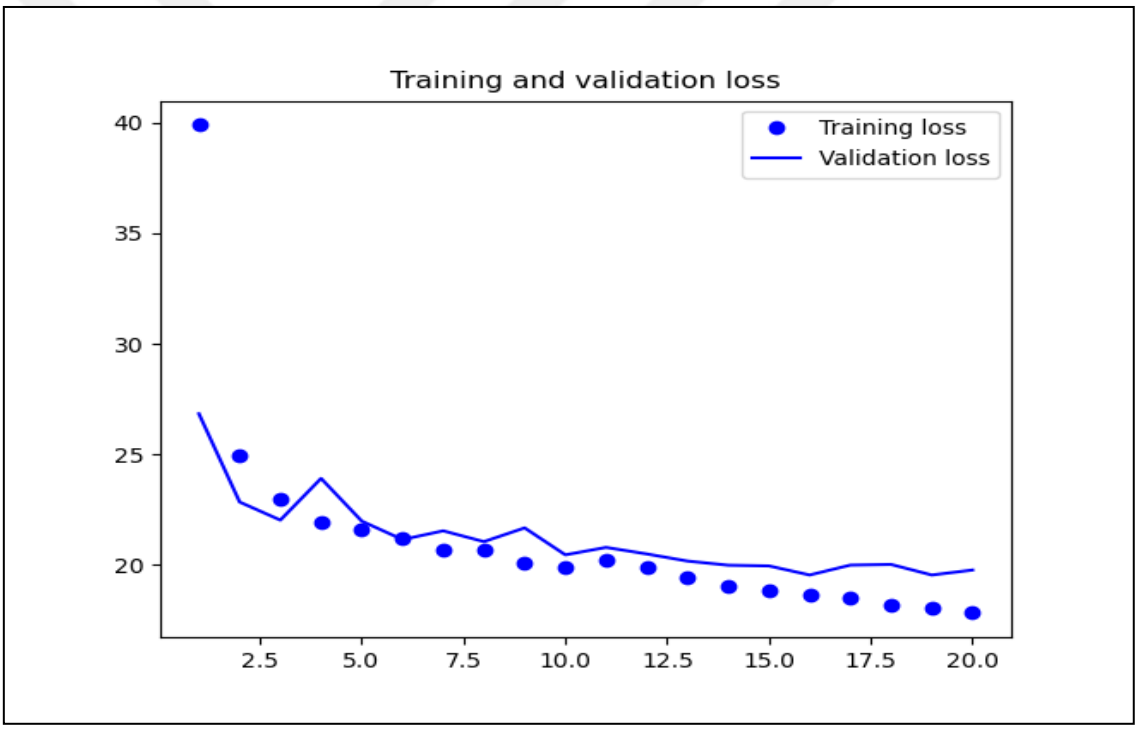
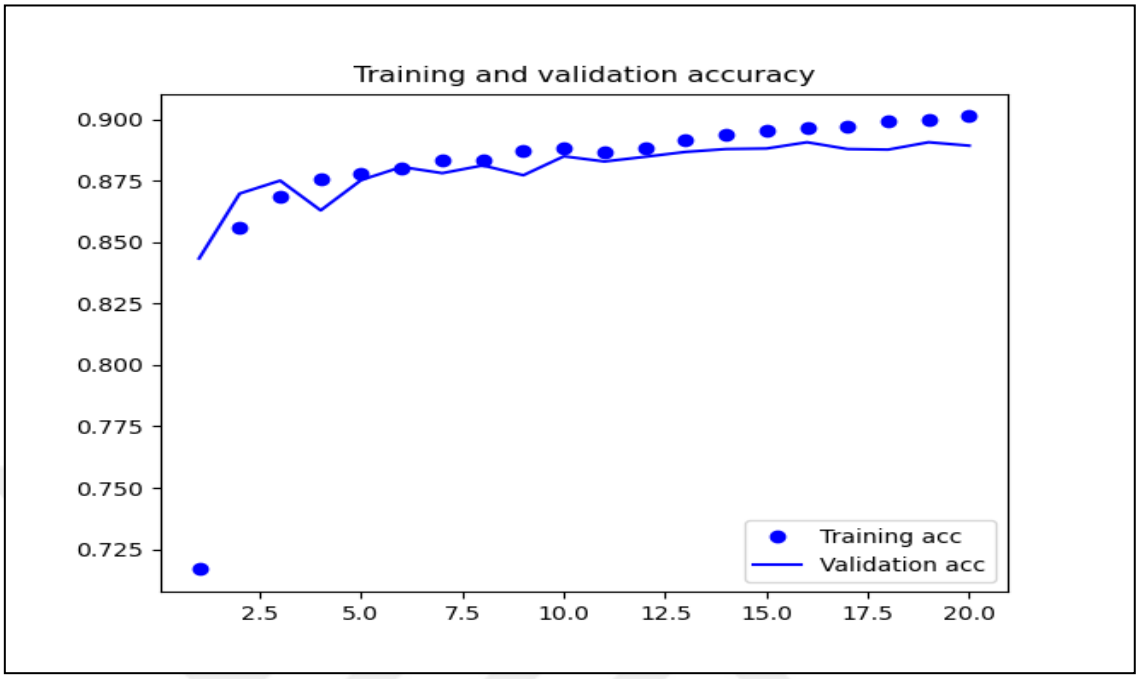


Figure 4.12: Training and validation loss curves of U-net model.



The following figures show the semantic segmentation prediction of the pre-trained U-net model for the "Grasping_Area" class.

Figure 4.13: The result of semantic segmentation for the object “cup” by U-net.

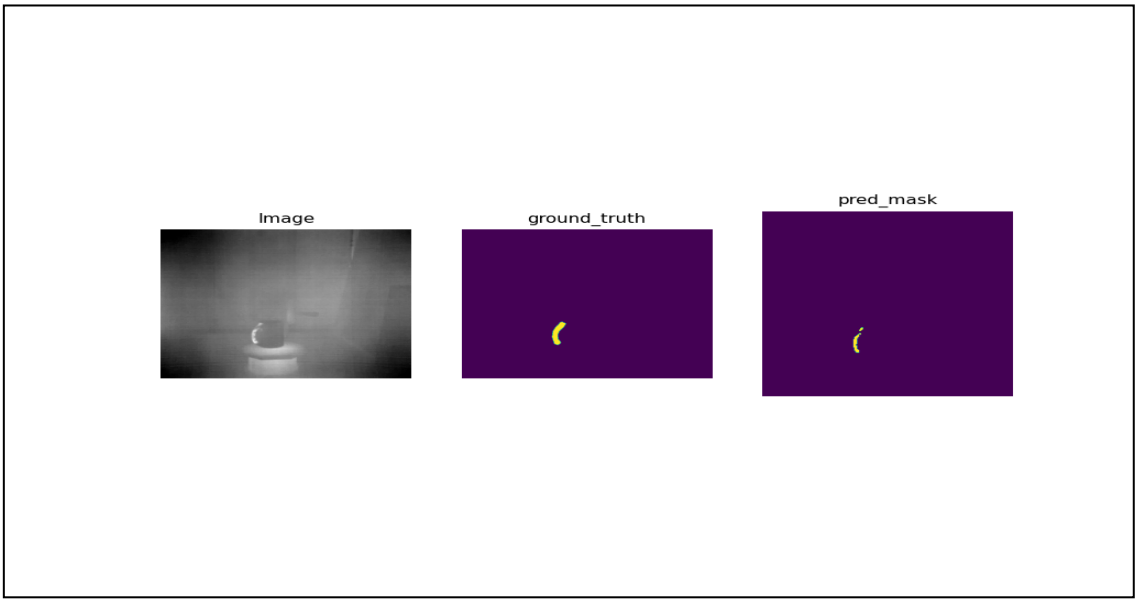


Figure 4.14: The result of semantic segmentation for the object “wineglass” by U-net.

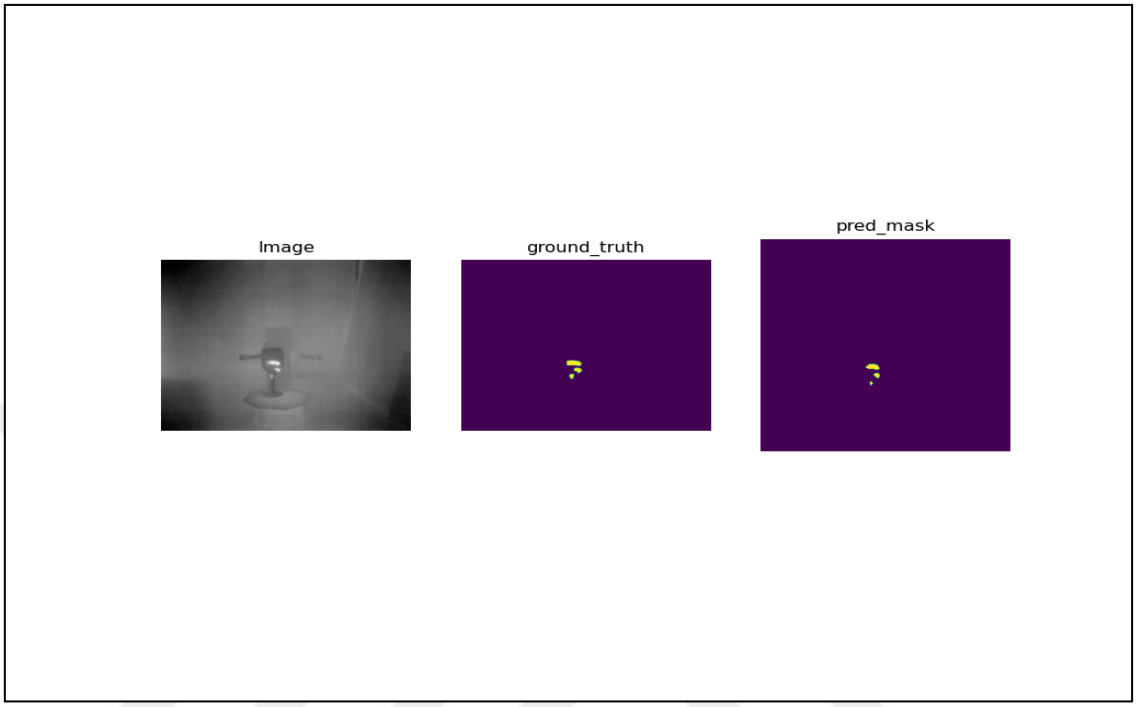


Figure 4.14: The result of semantic segmentation for the object “bottle” by U-net.

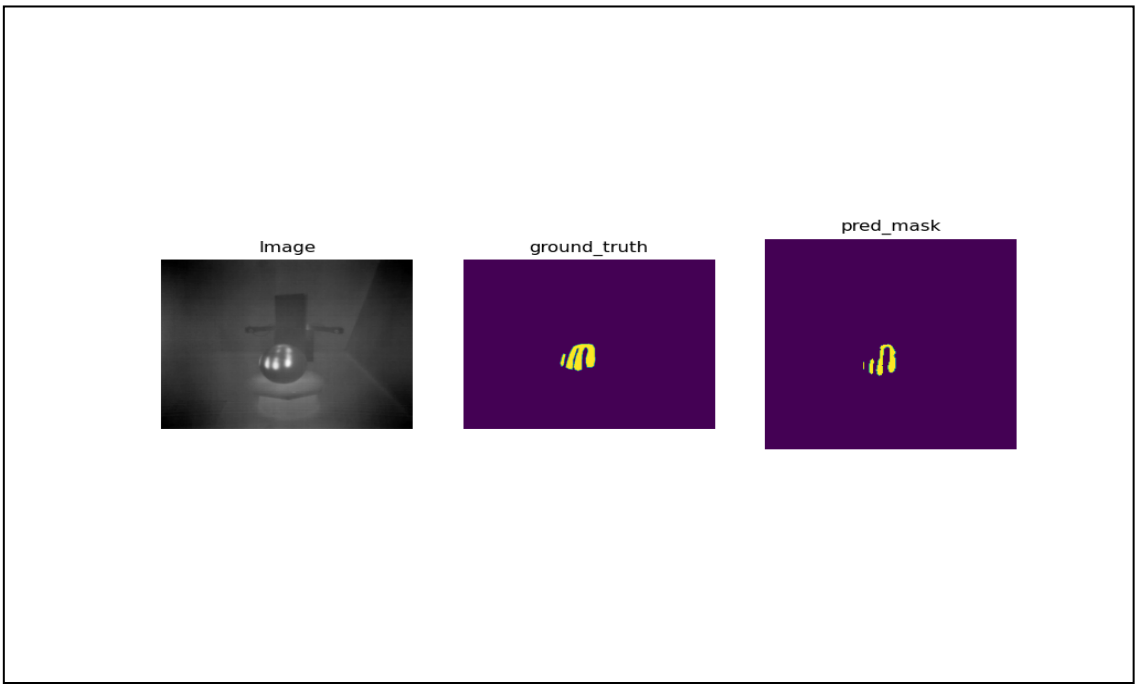


Figure 4.16: The result of semantic segmentation for the object “cup” by U-net.

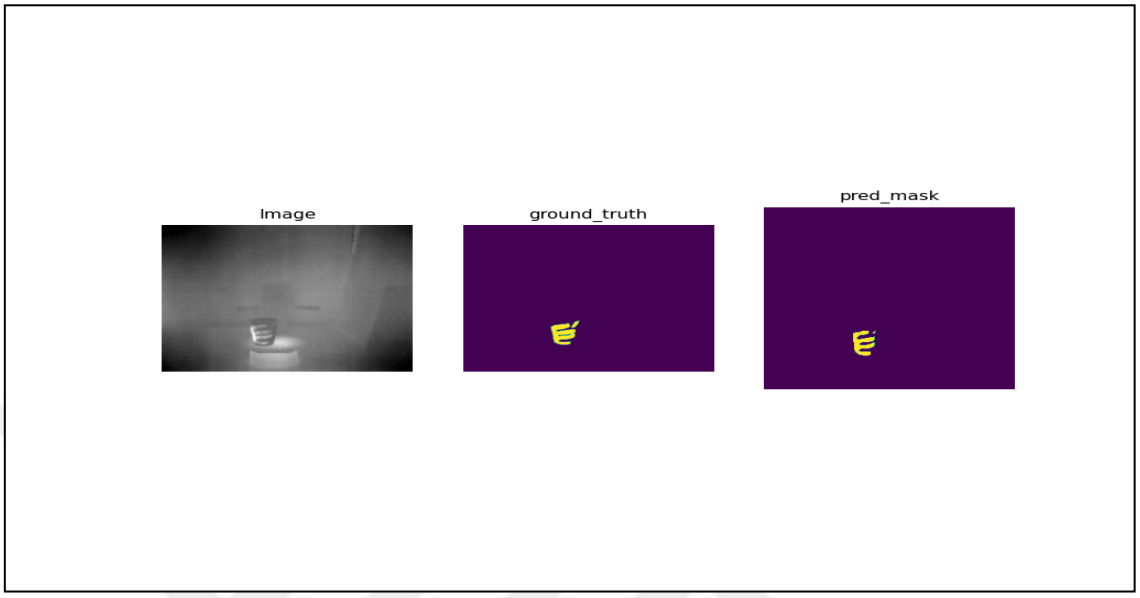


Figure 4.15: The result of semantic segmentation for the object “bowel” by U-net.

The third pre-trained model that we trained on our dataset was Feature Pyramid Network (FPN) for semantic segmentation prediction of the “Grasping_Area” class. The difference here from the U-net model is that we used "Resnet50" as our backbone with 10 batch size and 5 epochs. The Jaccard distance loss the dice coefficient functions were used as our metrics during the training. We have achieved 76% accuracy with 21.95 loss value.

The following figure shows the training and validation loss curves for pre-trained U-net

Figure 4.17: Training and validation loss curves of FPN model.

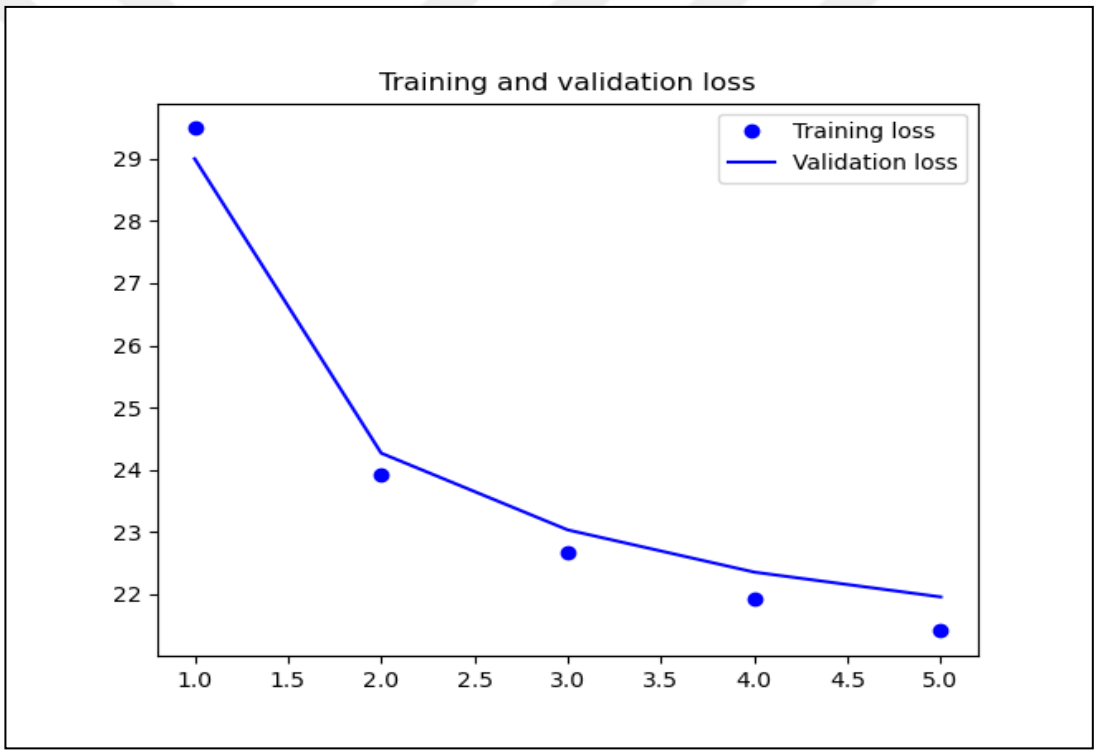
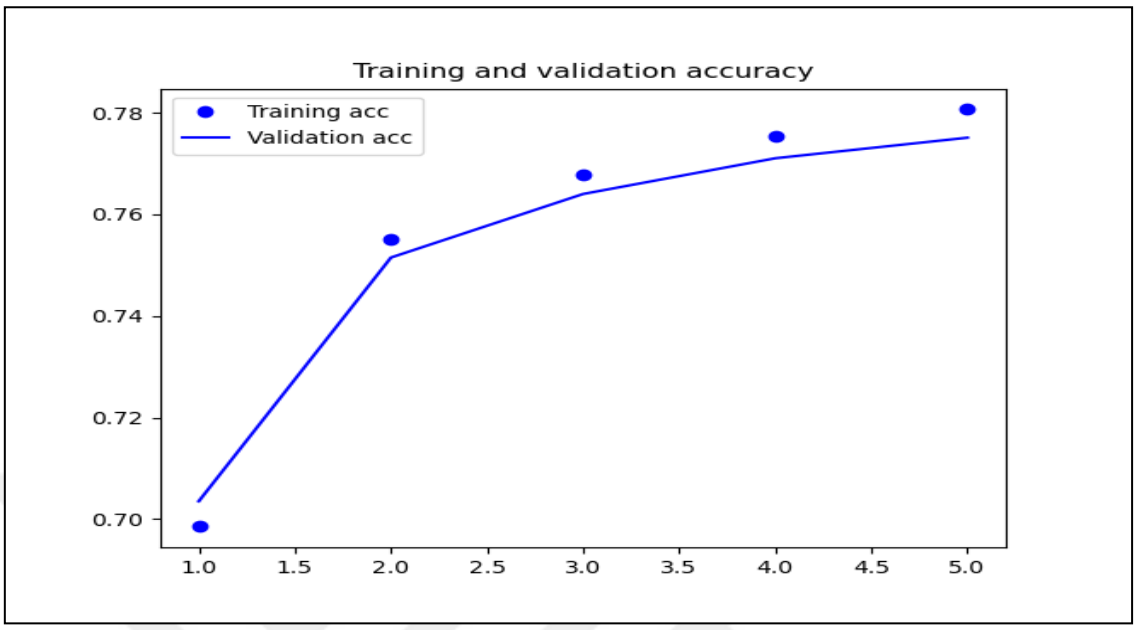


Figure 4.18: Training and validation accuracy curves of FPN model.



The following figures show the semantic segmentation prediction of the pre-trained FPN model for the "Grasping_Area" class.

Figure 4.20: The result of semantic segmentation for the object “bottle” by FPN.

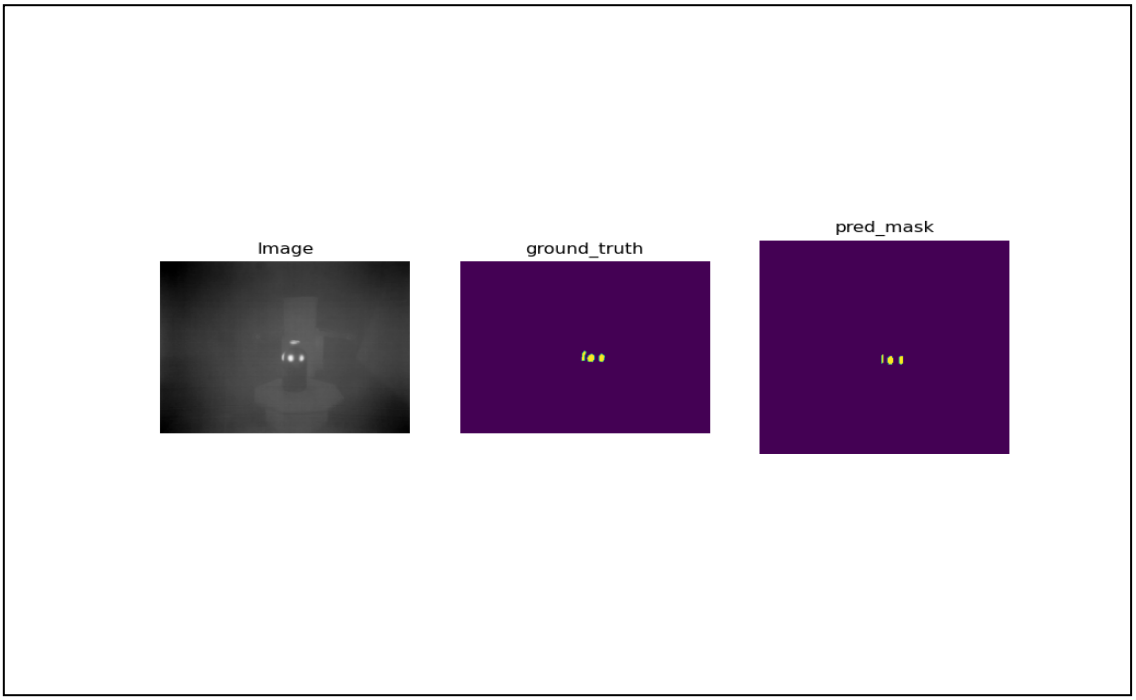


Figure 4.20: The result of semantic segmentation for the object “bowel” by FPN.

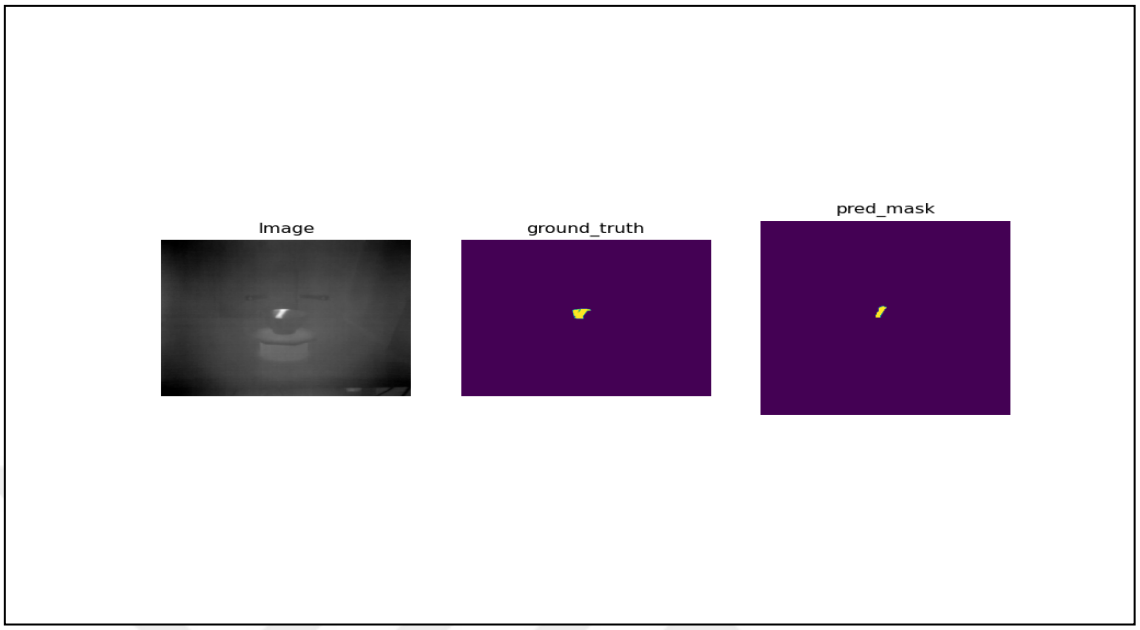


Figure 4.19: The result of semantic segmentation for the object “cube” by FPN.

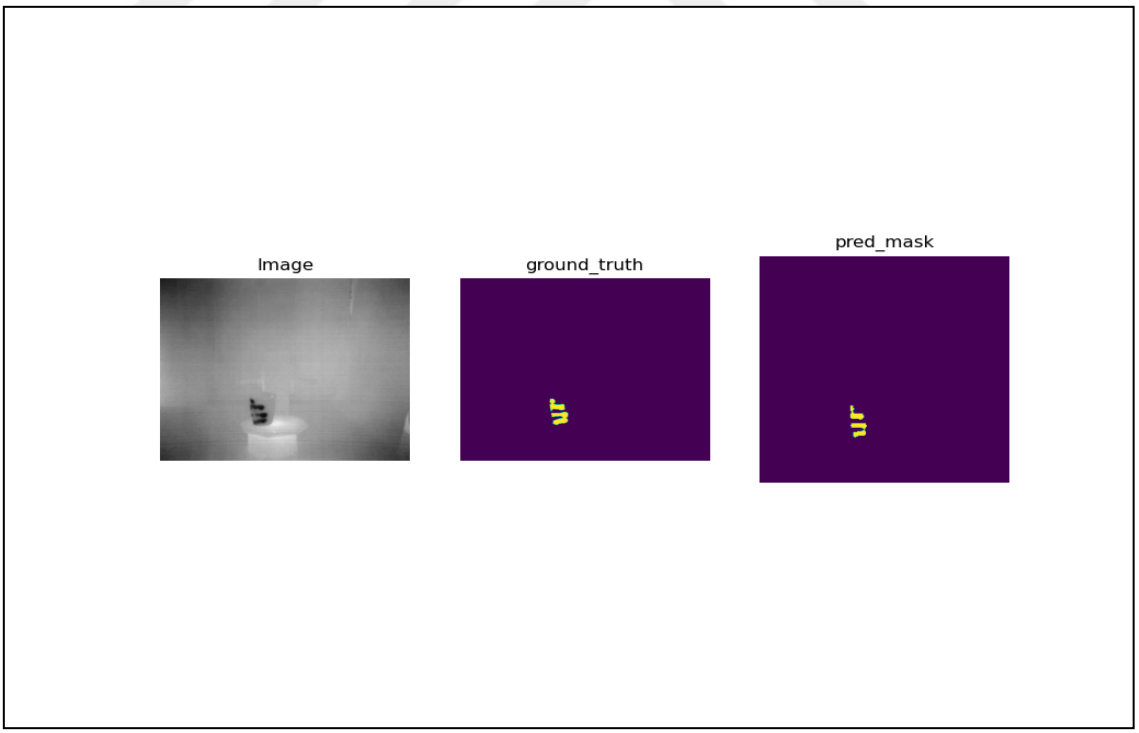
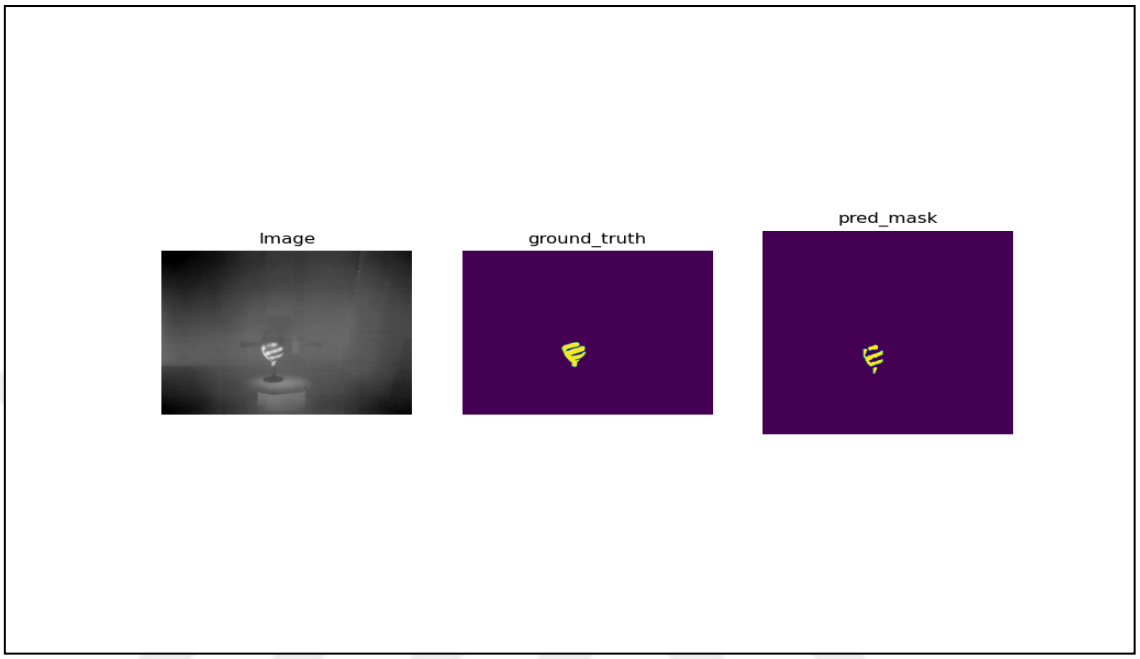


Figure 4.21: The result of semantic segmentation for the object “winglass” by FPN.



The following table shows the accuracy results obtained from the three pre-trained models for the “Grasping_Area” class segmentation.

Table 4.1: The accuracy results of the three pre-trained models.

Model	Backbone	Weights	Prediction time	Loss	Accuracy
Mask RCNN	Resnet101	mask_rcnn_coco	1.31s	1.11	0.95
U-net	Resnet34	ImageNet	2.12s	19.77	0.88
FPN	Resnet50	ImageNet	2.73s	21.95	0.76

The following table shows the accuracy results for semantic segmentation of the pre-trained U-net and FPN models with their hyperparameters values.

Table 4.2: The accuracy results of the semantic segmentation.

Model	Backbone	Optimizer	Learning rate	Batch_ size	Epochs	Loss	Accuracy
U-net	Resnet34	SGD	0.01	32	20	19.77	0.88
FPN	Resnet50	Adam	0.001	10	5	21.95	0.76

The following table shows different mAP values obtained from the pre-trained Mask R-CNN model by tuning particular hyperparameters.

Table 4.3: mAP results from our experiments on pre-trained Mask R-CNN model.

Exp.	Valid. _Steps	RPN_ Anchor_ Scales	Learning _Rate	Weight_ Decay	Best Epochs	Valid_ Loss	mAP
1	50	(4,8,16,32,64)	0.001	0.0001	5	1.850	0.00
2	50	(32,64,128,256, 512)	0.001	0.0001	5	0.999	0.500
3	80	(32,64,128,256, 512)	0.001	0.01	3	1.14	0.723
4	50	(32,64,128,256, 512)	0.001	0.005	3	1.22	0.728
5	80	(32,64,128,256, 512)	0.001	0.001	5	1.033	0.781
6	80	(32,64,128,256, 512)	0.001	0.005	5	1.109	0.783
7	80	(32,64,128,256, 512)	0.001	0.0001	5	1.110	0.95

5. DISCUSSION

This thesis aims to localize and segment the human hands' contact patterns on the object (fingers and palm locations) as references for stable grasps. The proposed method is based on data driven. We used a detailed data set called ContacDB, which contains 50 household objects grasps with two particular intents by 50 participants to predict the bounding boxes of the labeled grasping area and segment it by using a pre-trained model for Mask R-CNN. In this proposed method, we were interested in the fingers and the palm locations; based on that, our data set contained eight objects (mug, bottle, cup, wineglass, sphere, cube, and pyramid) selected from the ContactDB data set. The labeling process was made manually for 6037 images in our data set by using LabellImag. We achieved significant results for the bounding boxes prediction part and the segmentation part with ~0.95 value for mAP with minimum overfitting in the training phase. From the deep learning implementation process, we observed that the hardware requirements for implementing deep learning for computer vision tasks, specifically CNN architectures, are demanding to have a lot of computing power as training is significantly computational expensive, especially for long period epochs or iterations with data set of size more than two thousand. For the operating system, the best choice is either Windows or Ubuntu; These operating systems can amplify their development environment with more processors and General Purpose-Graphic Processing Unit (GP-GPU). The part of the hardware responsible for speeding up the training process of training deep learning models is the Graphics Processing Unit (GPU). The suitable choice to work with is the GPUs from the NVIDIA company. It considers very popular for deep learning training and deployment as it has robust software and community support. The RAM should be carefully selected to be suitable for the GPU's power (more than 10 GB). The CUDA (Computer Unified Device Architecture) library version should be cooperative with the GPU (NVIDIA CUDA) to using the GPU's parallel nature. The CUDNN (CUDA Deep Neural Network) library that has been used to provide primitives for deep learning methods is from NVIDIA to be highly optimized for the other hardware components and runs faster.

Our data set contained infrared images (grayscale images); the benefits we have found through the training are that it was more than enough to implement object detection for the grasping area as our targets were brighter, and the background has no color. Using an infrared data set was less computationally heavy to train and test since it reduced the input dimension, removed redundant information and with minimum number of parameters to train in the network. In some of our results, the model was unable to recognize objects close to each other. This problem is often related to using grayscale images with non-color diagnostic objects. This problem can be avoided when using color images as the influence of color in object recognition as it showed a significant color effect higher than non-color images. The color information has positive effects on object recognition as it increased the ability to recognize objects that are presented without surface details or to recognize the type of objects. The best results on the test data when training on the infrared data set was in experiment five with high mAP, mAP=0.95. The model showed extreme overfitting when training for significant iterations of epochs, especially for more than five epochs. By using data augmentation techniques, we were able to reduce the overfitting to a minimum; This ratio of overfitting was caused by using images without surface details for the training. For the hyperparameters tuning, the tuning of anchor boxes parameter showed a significant increase in the accuracy (from 0.00 to 0.500 mAP) when testing the model. However, the tuning of the weigh_decay parameter showed a considerable impact on the L2 regularization, which reduced the overfitting of the model and improved its performance to ~0.95. Using Mask R-CNN has a significant advantage in our implementation. Using our annotations files, including bounding box coordinates (xmin, ymin, xmax, ymax) for the object in the image, we were able two implement two-stage Mask R-CNN. The first stage is the bounding boxes prediction by Mask R-CNN to localize or detect the desired object by loading the bounding boxes for a given object and return them as masks; then Mask R-CNN will infer bounding boxes from the masks, which will be in the same size. The second stage network uses the best results we obtained from the first stage to produce the final segmentation with Mask R-CNN.

Since U-net was introduced as a model for segmenting biomedical datasets (grayscale images), its training and validation loss curves show more stability than Mask R-CNN during training which indicating that the model is more suitable for training a dataset containing infrared images. In training, we used "*ImageNet*" as the initializing weights; this step shows a positive effect on the stability of the model than "*mask_rcnn_coco" since* the first one contains large daily objects with complex shapes as our dataset has. Resnet34 was used as our backbone to solve the problem of getting the model's performance to be more saturated and decreasing rapidly. By using the identity function in Resnet34, any loss of information can be avoided by preserving the input of the first layer. Since U-net is used in single-pixel classification tasks with a single feature (gray level) and with one-dimensional feature space by applying a loss function for each pixel in the input image, we achieved a good accuracy (88%) with such limited size of a dataset containing grayscale images. Despite the stability of the U-net model performance during the training, which is better than Mask R-CNN, it achieved 88% accuracy with 20 epochs less than Mask R-CNN, where it reached 95% in only 5 epochs. This is due to the characteristics of the Mask R-CNN and U-net models; the first model was used to predict the locations of the bounding boxes and then segment it using instance segmentation for each recognized location with foreground pixels(background) which solved the problem directly, the second model is used to segment the union of all masks followed with complex post-processing to split the segmented mask into one mask object with no foreground pixels. FPN training and validation curves exhibit a loss and accuracy fullness within only 5 epochs but provide segmentation output performance with positive prediction. Despite using Resnet50 as a backbone and "ImageNet" as initializing weights, FPN could not reach more than 76% accuracy. Also, it displays the typical signs of overfitting within minor training epochs as the other models.

6. CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

In conclusion, three pre-trained models were adopted in this thesis to localize and segment the grasping area of a human hand using infrared images. The first pre-trained model we used was Mask R-CNN to implement two stages of object detection; the first stage was using Mask R-CNN with ResNet101 as a backbone to predict the bounding boxes of the grasping area on the object with a high value of mAP. The second stage was using the results we obtained from the first stage to produce the final instance segmentation of the grasping area with Mask R-CNN. In this model, we have achieved 0.95 accuracy with a 1.11 loss value. The second pre-trained model was U-net with ResNet34 as a backbone to implement semantic segmentation of the grasping area on the objects. In this model, we have achieved 0.88 accuracy with a 19.77 loss value. The last model was FPN with ResNet50 as a backbone to also implement semantic segmentation on the grasping area on the objects. In this model, we have achieved 0.76 accuracy with a 21.95 loss value. Despite the positive predictions we have achieved with these pre-trained models, each pre-trained model shows signs of overfitting within minor epochs during the training on our dataset. This problem is often related to using a limited-sized dataset containing grayscale images with non-color diagnostic objects. The results obtained from this work indicate stable grasp locations extracted from the human hands' contact patterns on the object. Finally, we can say one realizes when searching for the state-of-the-art object detecting network that this area is under significant development with new ideas, almost daily papers, and improvements being released. With all progress, it is quite a challenge to keep up with it. Object detecting and image analysis using machine learning future will be an interesting topic to follow.

6.2 FUTURE WORK

In future work, A high recommendation for data collection and generation in large-sized for this specific contact pattern (fingers and palm) is needed to overcome or decrease the

overfitting and increase the stability during the training process. Further exploration of other object detection models would be interesting to implement and compare with the models we used (e.g., YOLO) in this thesis. Moreover, in an industrial setting, the extracted locations can be used as a reference for stable grasps to be used by soft robotic manipulators aiming to cover these localized locations.

REFERENCES

Books

Chollet, F., 2017. *Deep Learning with Python.*

Kuenzer, C., 2013. *Thermal Infrared Remote Sensing,* London: Springer.

Michelucci, U., 2018. *Applied Deep Learning.* Switzerland: Apress.

Michelucci, U., 2019. *Advanced Applied Deep Learning,* Switzerland: Apress.

Shanmugamani, R., 2018. *Deep Learning for Computer Vision*. UK: Packt.

Thomas, S., 2019. *Pytorch Deep Learning Hnads-On,* UK: Packt.

Periodicals

Altalib, A., 2019, *Depth Map Extraction Using Structured Light*. [online].

Bicchi, A. et. al, 2000, *Robotic grasping and contact: A review*. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.7081&rep=rep1&type=pdf [accessed 22 November 2020].

Brahmbhatt, S. et al., 2019, *ContactDB: Analyzing and Predicting Grasp Contact via Thermal Imaging*. [online] Available at : https://openaccess.thecvf.com/content_CVPR_2019/html/Brahmbhatt_ContactDB_Analyzing_and_Predicting_Grasp_Contact_via_Thermal_Imaging_CVPR_2019_paper.html [accessed on 1 January 2020].

COCO, 2020. [online] Available at: https://cocodataset.org/#home [accessed on 5 April 2020].

Corona, E. et al., 2018, *Active garment recognition and target grasping point detection using deep learning*. [online] Available at: https://www.semanticscholar.org/paper/Active-garment-recognition-and-target-grasping-deep-Corona Aleny%C3%A0/3ebc0c8a328a17ccfc55cf5dcc41b56bb9d3f480 [accessed on 10 December 2020].

Data School, 2021, *Simple guide to confusion matrix terminology*. [online] Available at: http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology.

Dollar, P. et. al, 2009, *Pedestrian detection: a benchmark, 2009 IEEE Conference on Computer Vision and Pattern Recognition*. [online].

Fang, Y. et. al, 2003, *Comparison between infrared-image-based and visible-image-based approaches for pedestrian detection, IEEE IV2003 Intelligent Vehicles Symposium. Proceedings*. [online].

Gao, J. et. al, 2019, *Infrared small target detection using a temporal variance and spatial patch contrast filter*. [online] Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8663277 [accessed on 29 October 202].

Girshick, R. et al, 2016, *Faster R-CNN*. [online] Available at: https://arxiv.org/abs/1506.01497 [accessed on 21 April on 2021].

Girshick, R. et al., 2018, *Mask R-CNN*. [online] Available at: https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html [accessed on 5 February 2021].

Goldfeder, C. et. al, 2009, *Data-driven grasping with partial sensor data. In Intelligent Robots and Systems*. [online] Available at: http://www.cs.columbia.edu/~allen/PAPERS/datadrivengraspiros2009.pdf [Accessed on 10 July 202].

Goldfeder, C. et. al, 2011, *Data-driven grasping. Autonomous Robots*. [online]

He, K. et al, 2015, *Deep Residual Learning for Image Recognition*. [online] Available at:https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html [accessed on 5 November 2020].

Huang, G. et al.,2018, *Densely Connected Convolution Networks*. [online] Available at: https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html [accessed on 2 February 2020].

Jaderberg, M. et. al, 2016, *Spatial Transform Networks*. [online] Available at: https://arxiv.org/abs/1506.02025 [accessed on 3 March 2020].

Jin, X. et. al, 2017, *A survey of infrared and visual image fusion methods*. [online] Available at: https://ui.adsabs.harvard.edu/abs/2017InPhT..85..478J/abstract [accessed on 11 August 2020].

Kang, H. et al., 2020., *Real-Time Fruit Recognition and Grasping Estimation for Autonomous Apple Harvesting*. [online] Available at:

https://www.mdpi.com/1424-8220/20/19/5670/pdf [accessed on 12 January 2021].

Krizhevsky, A. et. al, 2012, *ImageNet Classification with deep learning convolution neural network.* [online] Available at: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf [accessed 6 May 2020].

Kumra1and, S. et. al, 2017, *Robotic Grasp Detection using Deep Convolution Neural Network.* [online] Available at: https://www.researchgate.net/publication/310953119_Robotic_Grasp_Detection_using_Deep_Convolutional_Neural_Networks [accessed on 10 December 2020].

Mahler, J. et. al, 2019, *Learning ambidextrous robot grasping policies*. [online] Available at: https://robotics.sciencemag.org/content/4/26/eaau4984?utm_campaign=the_download.unpaid.engagement&utm_source=hs_email&utm_medium=email&_hsenc=p2ANqtz9XWWVE7Xcwk55OpV3KpluuqKVKES6HATVGbDSx2epWupSYZFcJbyyQYO41p9rwVynq1M3w9WRiyV9HZVSzYVt-j3are8KboBS_3fvywgQc8pUQ5so [accessed on 15 January 2021].

Mahler, J. et. al, 2017, *Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics*. [online] Available at: https://arxiv.org/abs/1703.09312 [accessed on 10 July 2020].

Mahler, J. et. al, 2017, *Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning.* [online] Available at: https://arxiv.org/abs/1709.06670 [accessed on 13 April 2020].

Nazhat, A., 2020, *Stereo Vision Utilizing Deep Learning*. [online].

Prattichizzo, D. et al., 2012, *On the manipulability ellipsoids of underactuated robotic hands with compliance*. [online] Available at: https://www.sciencedirect.com/science/article/abs/pii/S0921889011001461 [accessed on 14 August 2020].

Ronneberger, O. et. al., 2015, *U-net: Convolution Networks for Biomedical Image Segmentation.* [online] Available at: https://arxiv.org/pdf/1505.04597.pdf [accessed on 21 July 2021].

Szegedy, C. et. al., 2014. *Going Deeper with Convolutions. IEEE* [online] Available at: https://arxiv.org/pdf/1409.4842.pdf [accessed 6 May 2020].

Varley, J. et al.,2015, *Generating Multi-Fingered Robotic Grasps via Deep Learning*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/7354004 [accessed on 12 May].

Wang, B. et al, 2017, *Fast infrared maritime target detection: binarization via histogram curve transformation*. [online] Available at: https://www.sciencedirect.com/science/article/abs/pii/S1350449516304546 [accessed on 15 July 202].

Watson, J. et. al., 2017, *Real-World, Real-Time Robotic Grasping with Convolution Neural Network.* [online] Available at: https://link.springer.com/chapter/10.1007/978-3-319-64107-2_50 [accessed on 29 August 2020].

Yi Lin, T. et. al., 2017, *Feature Pyramid Networks for Object Detection.* [online] Available at: https://arxiv.org/pdf/1612.03144.pdf [accessed on 21 July 2021].