Endoscopic capsule spatial awareness by using image feature extraction and machine learning

Submitted in fulfillment of the conditions for the degree of the
Diploma of Electrical and Computer Engineering.

Sofia Athanasiou

Thesis Committee
Prof. Michael Zervakis (Supervisor)
Prof. George Stavrakakis
Dr. Eleutheria Sergaki (Co-supervisor)

Electrical and Computer Engineering
Technical University of Crete
Chania,Greece

September,2022

Abstract

Wireless Endoscopy Capsule becomes more and more famous since it is the only way for the physicians to have a non-invasive view of the small intestine and identify bleeding, polyps, Crohn's disease and other abnormalities of the small intestine. Wireless Capsule Endoscopy is happening with the patient swallowing an electronic capsule, the capsule includes a camera, that way is taking photographs of the Gastrointestinal Track as it moves from one organ to another. Those pictures are transferred to a recorder and from there a software is producing a video stream using those frames. The journey of the capsule in gastrointestinal track might last 8 or more hours based on the movement of the track, during that time it will collect more than 50,000 to 100,000 pictures producing a long video. Examining that amount of frames, it is a very time-consuming procedure and a tiring task for gastroenterologists. There are many researches that try to reduce that time using extra processing of the videos. Nowadays, with the evolution of Machine Learning and Artificial Intelligence for image classification, they are a great supporting tool for physicians.

This thesis studies the spatial information extraction problem for Wireless Endoscopy Capsule using Machine Learning techniques. Being able to locate the endoscopy capsule can provide doctors with the opportunity to examine only the part of the gastrointestinal track that is of their most interest. Moreover, producers of endoscopy capsules can use such techniques in order to save battery, by reducing the pictures tracked in areas that are not of the most importance.

In order to resolve that problem, we developed three different Convolution Neural Networks models, which achieve the automatic classification of the pictures they receive to the corresponding organ of the digestive system (esophagus, stomach, small intestine and large intestine). The data we used to train and verify our models are from the same collection as in the Thesis of Mr. A. Polydorou, at the Electrical and Computer Engineering Department of Technical University of Crete in 2018. For the feature extraction part of the model, Convolutional two-dimensional (Conv2D) and Maxpooling two-dimensional (Max2D) were used with ReLU activation function. For the classification were used Flatten and Dense layers with activation function softmax function. In one model there was also use of Dropout in order to randomly disconnect nodes and their edges.

For the validation of the performance of our models, we used metrics such as Accuracy(Acc), Sensitivity(Sens), Specificity(Spe), Error Rate(Err) and Precision(Pre). Since it is a multi-class classification those values are calculated for each organ (or class) individually. The best performance among our models holistically comes from Model 2 where is providing for esophagus: Accuracy of 95.16%, Error Rate of 4.83%, Precision of 84.84%, Specificity of 94.73% and Sensitivity of 96.55%. For stomach provides, Accuracy of 91.93%, Error Rate of 8.06%, Precision of 90.9%, Specificity of 96.55% and Sensitivity of 81.08%. For small intestine, is performing with Accuracy of 95.96%, Error Rate of 4.03%, Precision of 92.85%, Specificity of 97.89% and Sensitivity of 89.65%. Finally for large intestine, Accuracy of 99.19%, Error Rate of 0.8%, Precision of 96.66%, Specificity of 98.94% and Sensitivity of 100%.

Keywords:

Wireless Capsule Endoscopy(WCE), gastrointestinal (GI), Machine Learning(ML), Artificial Intelligence(AI), Convolutional Neural Network(CNN), Deep Learning, supervised learning, False Positive Ratio (FPR), False Negative Ratio (FNR), Dropout, Layer, ReLU function, Softmax function, digestive organ classification, Sensitivity, Specificity, Error Rate, Accuracy(ACC), Precision.

Abstract

Η ενδοσκοπική κάψουλα γίνεται όλο και πιο διάσημη καθώς αποτελεί τον μόνο μη-επεμβατικό τρόπο ώστε να υπάρχει μια εικόνα του λεπτού εντέρου, ώστε να αναγνωριστούν αιμορραγίες, πολύποδες, ασθένεια του Crohn και άλλες δυσμορφίες του βλενογόννου του λεπτού εντέρου. Για την εξέταση ο ασθενής καταπίνει την ηλεκτρονική κάψουλα, η οποία περιέχει και κάμερα, έτσι μπορεί να λαμβάνει φωτογραφίες του γαστρεντερολογικού συστήματος καθώς κινείται από το ένα όργανο στο άλλο. Αυτές οι φωτογραφίες μεταφέρονται σε έναν υπολογιστή και με την βοήθεια λογισμικού παράγεται ένα βίντεο το οποίο περιέχει αυτές τις εικόνες. Το ταξίδι της ενδοσκοπικής κάψουλας μέσα στο γαστρεντερολογικό σύστημα του ασθενούς μπορεί να διαρκέσει από 8 εώς και περισσότερες ώρες ανάλογα με την κίνηση του. Μέσα σε αυτή την ώρα η κάψουλα λαμβάνει 50.000 με 100.000 φωτογραφίες παράγοντας ένα αρκετά μεγάλο σε διάρκεια βίντεο. Έτσι η διεξοδική εξέταση αυτού του βίντεο αποτελεί μια εξαιρετικά χρονοβόρα διαδικασία για τους γιατρούς. Υπάρχουν πολλές έρευνες οι οποίες προσπαθούν μέσω προεπεξεργασίας του βίντεο να μειώσουν αυτό τον χρόνο. Σήμερα, με την εξέλιξη της Μηχανικής Μάθησης και της Τεχνητής Νοημοσύνης, ειδικά στον τομέα της ταξινόμησης φωτογραφίων, αποτελούν σπουδαία εργαλεία για τους ειδικούς υγείας.

Αυτή η διπλωματική, μελετάει την εξαγωγή χωρικών πληροφοριών της Ενδοσκοπικής κάψουλας με την χρήση τεχνικών Μηχανικής Μάθησης. Το να είμαστε σε θέση να εντοπίζουμε την ενδοσκοπική κάψουλα κατά την πορεία της στο γαστρεντερολογικό σύστημα δίνει την δυνατότητα στους γιατρούς να εναποθέσουν ενέργεια στην εξέταση μέρους του βίντεο που αφορά μονάχα το σημείο ενδιαφέροντος. Επιπλέον, οι κατασκευαστές της ενδοσκοπικής κάψουλας θα μπορούσαν να χρησιμοποιήσουν τα προτεινόμενα μοντέλα ώστε να κάνουν εξοικονόμιση μπαταρίας όπου δεν χρειαζόμαστε πολλές φωτογραφίες και να λαμβάνουν περισσότερες λήψεις, όταν αυτό είναι απαραίτητο.

Για την επίτευξη αυτού, δημιουργήθηκαν 3 διαφορετικά μοντέλα Νευρωνικών Δικτύων, τα οποία επιτυγχάνουν την ταξινόμηση των φωτογραφιών στο αντίστοιχο όργανο που ανήκουν. Τα δεδομένα που χρησιμοποιήσαμε έρχονται από την διπλωματική του κ. Α. Πολυδώρου για την σχολή ΗΜΜΥ, του Π.Κ. το 2018. Για την εξαγωγή των χαρακτηριστικών χρησιμοποιήθηκαν Convolutional επίπεδα και Maxpooling επίπεδα και ReLU activation function. Για την πρόβλεψη χρησιμοποιήθηκαν Flatten και Dense επίπεδα με την χρήση Softmax activa-

tion function. Σε ένα από τα μοντέλα χρησιμοποιήθηκε και Dropout το οποίο αποσυνδέει τυχαία του κόμβους και όλες τις ακμές τους από το νευρωνικό δίκτυο.

Για την αξιολόγηση των μοντέλων χρησιμοποιήθηκαν μετρήσεις όπως Accuracy(Acc), Sensitivity(Sens), Specificity(Spe), Error Rate(Err) και Precision(Pre). Καθώς είναι ένα πρόβλημα με πολλαπλές τάξεις, οι τιμές αυτές υπολογίζονται για κάθε όργανο ξεχωριστά. Η καλύτερη ολιστικά απόδοση έρχεται από το Μοντέλο 2 το οποίο προσφέρει όσον αφορά τις τιμές αυτές και για κάθε ξεχωριστό όργανο του πεπτικού συστήμα. Για τον οισοφάγο: Accuracy 95.16%, Error Rate 4.83%, Precision 84.84%, Specificity 94.73% και Sensitivity 96.55%. Για το στομάχι: Accuracy 91.93%, Error Rate 8.06%, Precision 90.9%, Specificity 96.55% και Sensitivity 81.08%. Για το λεπτό έντερο: Accuracy 95.96%, Error Rate 4.03%, Precision 92.85%, Specificity 97.89% και Sensitivity 89.65%.. Τέλος, για το παχύ έντερο: Accuracy 99.19%, Error Rate 0.8%, Precision 96.66%, Specificity 98.94% και Sensitivity 100%.

Acknowledgements

I would like to thank professor Zervakis and Dr. Sergaki for their amazing help and trust towards me in order to take over that topic. Their knowledge and their contribution and their methodically approach were major importance in order to begin and continuous my studies in implementation problems of data processing using AI methodologies. I would like to thank professor Stavrakakis for his contribution in my academic education, both in knowledge and in approaching engineering problems in their real life aspects. Moreover, I would like to thank Dr. Andreas Polydorou Professor of Surgery and Endoscopic Surgery, who provided me with the data and who was always there to discuss and answer my question around technical details regarding the dataset. Also, I would like to thank Alex Polydoroy, who provided me with the dataset and he supported me with questions and technical information. I hope the cooperation with the team the last years to create the substructure for future cooperations.

I would like also to thank my family, Georgia, Thomas and Thanasis and my brothers Konstantinos and Sotiris for providing me with the tools and support towards that degree, each one with his own way. A big thank you to all of my friends especially Fotis, Elpida, Marianna, Aggelos and Katerina for being there for me, providing me with great times and memories. Furthermore, I would like to thank Alexandros for his contribution in getting done with my subjects fast and supporting me. I would also like to thank people that came in my life the last year and were a great inspiration for me in order to grow and also fulfill this thesis, Lydia and Christina!

Last but not least, I would like to thank Charlotte Cardin who made writing my thesis way more fun with her music!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we are setting out the aims and objectives of our project, explaining the overall intention of the project and specific steps that will be taken to achieve that intention.

1.1 Motivation

Endoscopy capsule looks like a pill that patients can swallow in order to examine the majority of their digestive system. It is starting from esophagus and is ending up at the colon. Many times, physicians need to go through videos that might even last 16 hours so that they make sure that their examination was in every part of the digestive system. Often it is beneficial to know where the endoscopy capsule is at the moment as well as where each organ starts to be obvious by the endoscopy capsule. That will save a great amount of time from the doctors that are examining the video of endoscopy capsule. In the current research, the aim is to be able to classify the digestive organ that each frame belongs using machine learning techniques. Resulting into having the important information of where the capsule is currently located. In average an experienced doctor needs approximately 40 minutes in order to examine a video from an area of interest, that amount of time is much bigger for a doctor without experience. Using spatial information extraction, doctors can focus only in the area of interest from the very beginning, without the need of going through the video so to detect where that areas stand. Taking under

consideration, the amount of videos that a doctor might be in the need to examine each day, without smarter ways of eliminating the information that he needs to manually go through, the possibilities of a human error or a misdiagnosis are getting higher, so using such techniques a diagnosis is becoming faster and more accurate.

1.2 Aims and Objectives

The aims and objectives of the project are the following:

- Create and test different machine learning models using as a base Convolutional Neural Networks in order to extract spatial information of the endoscopy capsule by using image classification techniques.
- One other main objective was to be able to have quite good results without the need of going in any further image pre-processing of the frames before feeding them to the model for training. Without extra pre-processing of the images, there is a resources gain.
- After creating the models, extract information from their comparison.
- Broaden our knowledge around Machine Learning and the tools that are currently been used in the field.

1.3 Description of the work

This research is based on the frames that were extracted from Endoscopy Capsule videos of patients that were provided to us by Dr. Polydorou. Used the Pillcam Software in order to view and process the Endoscopy Capsule videos, it provides also the ability to extract smaller videos from the areas of interest. So videos were divided into smaller once, for each area of interest(esophagus, stomach, small intestine and colon).

A small python script was used in order to split the videos into frames and perform a sampling of the images. Then those images were divided into folders and label so to

become input in the training models that we developed.

The final goal of the research was to be able to locate the endoscopy capsule by using machine learning techniques, and that translates to an image classification problem.To implement that, we used python programming language and some tools that come with it like Keras API and Tensorflow in order to create our models. All the three models were created, compiled and ran using Google Colab platform in order to have better performance of GPU and CPU.

When the models were ready, we analyzed their performance, taking under consideration the loss and accuracy values of both training and validation datasets. For the models that those results were sufficient, we stepped in and calculated also Sensitivity, Error Rate, Precision Specificity and Accuracy when it comes to predictions of unknown images.

1.4 Structure of the thesis

Following comes a brief description of the document, explaining in short what you should expect reading in each chapter of it.

Chapter 1: A brief introduction around the objectives of this thesis and the motivations behind it.
Chapter 2: What is endoscopy capsule and the evolution of spatial information extraction using machine learning.
Chapter 3: Theory around Machine Learning, Convolutional Neural Networks and some tools that were used in order to implement the current research.
Chapter 4: Design of the thesis, how the data were prepared and manipulated. Machine Learning parameters that were tuned. Design of the models.
Chapter 5: Architecture and implementation of the models. Models' elements and decisions based on their outputs.
Chapter 6: Results of each model and comparison of those results, analysis of values

such as sensitivity, Error Rate.

Chapter 7: Future work that can be benefit out of the current research.

Appendix A: Sample images of the dataset, representing all the different organs of the digestive system.

Appendix B: Graphs of accuracy and loss while running the Machine Learning algorithms.

Chapter 2

Background and Related Work

In this chapter, there follows an explanation of the endoscopy capsule and a presentation of researches that are around the same topic as this one. Goal is to provide information around the endoscopy capsule and what is the current state of researches around the spatial information extraction using image classification and machine learning.

2.1 Endoscopy Capsule

All images used for this research were taken from the endoscopy capsule and specifically from Pillcam SB3. It is intended for visualization of the small bowel mucosa, may be used for diagnosis of Crohn's disease, the detection of bleeding and lesions that might be causing iron deficiency anemia. Can be used as a diagnosis tool for adults and children above 2 years old [1]. The patient should fast for at least 12 hours before the examination. The patient ingests the capsule in the supine position with 10 ml of water. The images are received to the recorder via three thoracic sensors.[[2]]

Pillcam SB3 comes with a frame rate per second of 2 to 6 frames and a 12 hours battery life. The size of the camera is 12x26 mm and weights 3 gr and is build out of biocompatible plastic. It has a field of view as of 156 degrees, and it is using 4 white light emitting diodes for illumination[3].

Figure 2.1: Structure of wireless endoscopy capsule

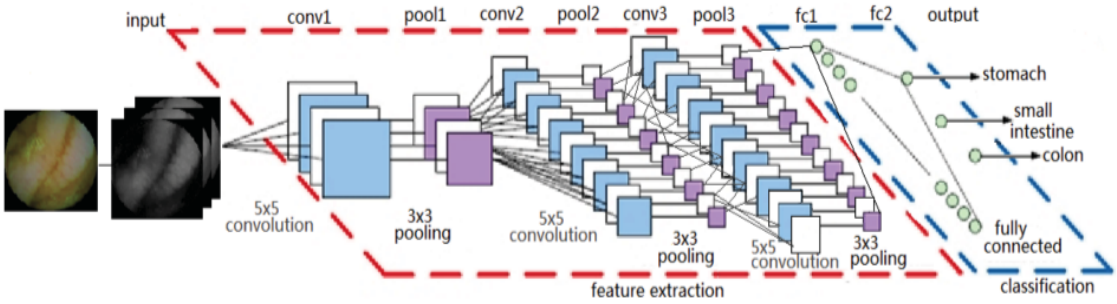
In Figure 2.1 you can see the structure of a wireless endoscopy capsule[4]. Breaking down its components based on the numbers that are in the figure:

1. Optical dome
2. Lens holder
3. Lens
4. Illuminating Lens
5. CMOS
6. Battery
7. ASIC Transmitter
8. Antenna

2.2 Similar researches

Starting with the research of Yuexian Zou et al. [5], one of the first researches around Deep Convolutional Neural Networks and organ classification for Endoscopy Capsule. Their dataset consists of 25 WCE test cases, at an amount of 1 million images. There was a selection of 60 thousand images for training and 15 thousand images for testing. Since they classified three organs: stomach, small intestine and colon, that means 20 thousand

Figure 2.2: Paper 1 Model



images for testing from each organ. The resolution of the images is 480x480.

There was an implementation of different experiments in order to examine the performance while using different network parameters. The system design comes in Figure 2.2.

In the first experiment, there is a comparison of the Deep Neural Network of Figure 2.2 performance and the state-of-art classification of the moment that refers to a SIFT-SVM classifier. The outcome is that SIFT-SVM classifies is providing less accuracy than a Deep Convolutional Neural Network plus is introducing higher computation cost since SIFT-SVM and SIFT-SC-SVM feature extraction is the most time-consuming.

Next, in the second experiment, there are changes in the parameters that the model is using in order to examine how the system corresponds to different configurations in terms of classification accuracy. From these experiments they came to the following conclusions:

1. Maxpooling is more robust for Wireless Endoscopy Capsule images than average pooling

2. Reducing number of filters will bring a reduction of training time but will reduce also the accuracy

3. Larger filter size may fail to catch the details of the image, again though it will reduce the training time

4. The impact of non-overlapping or overlapping for pooling way is not apparent.

In conclusion and after examining the results from table in Figure 2.3, using a deep Neural Network so to succeed to organ classification for Wireless Endoscopy Capsule, the classification accuracy is among 95%.

Another research, on digestive organs' classification, comes from Lee et al. [6]. They proposed a novel technique to segment Wireless Capsule Endoscopy video into the digestive organs using color change pattern analysis. Each digestive organ different patterns of intestinal contractions that are quantified as the features. They convert the frames for Red Green Blue (RGB) domain to the Hue Saturation Intensity (HSI) domain, using the mean of entire pixel of a frame. Then transform them in the frequency domain and using the High Frequency Content (HFC) function they segment the video into events. Those detected boundaries are either transition to another organ either unusual events in the current organ(e.g. bleeding). Lastly, there is a high level event representation using a tree structure, that is called event hierarchy of Wireless Endoscopy Capsule. Those events represent the different digestive organs. That technique is providing as a maximum Recall(H_r) of 76% and a maximum Precision(H_p) of 51%.

In 2013, Zhou et al.[7] proposed a new way of Wireless Capsule Endoscopy video segmentation, with the intention to break the video into two basic parts, stomach/small intestine and small intestine/large intestine. Using Color Uniform Local Binary Pattern (CULBP) for the variation of illumination and Ada-Support Vector Machine or Ada-SVM, a variation of SVM that boosts the performance of general SVM algorithm. There was a two-level segmentation method that is utilized for segmentation, which is reducing the computation time and overcomes the difficulty of training data acquisition. Later they use CULBP to advance discrimination of general uniform local binary pattern feature extraction, providing a more robust variation of illumination and better input for classification. Finally, Ada-SVM classifier is non-sensitive to changes of SVM classifier parameters, providing more robust predictions. To conclude, their method provides an average Precision(H_p) of 91.37% and an average Recall(H_r) of 88.5% when it comes to stomach/small intestine

Figure 2.3: Paper 1 Results

	accuracy	feature extraction	training time	testing time	total time
SIFT-SVM	71.70%	126min	51.82s	2.89s	127min
SIFT-SC-SVM	90.31%	205min	450.44s	12.62s	253min
DCNN-WCE-CS	95.52%	0	371min	2.45s	371min

recognition and respectively an average Precision(H_p) of 90.35% and average Recall(H_r) of 97.28% for small intestine/large intestine.

2.3 Present work

It this research, the goal if to extract spatial information of the endoscopy capsule using machine learning techniques, precisely using convolutional neural networks. As we can see in researches similar to this usually image preprocessing is used, we try to achieve equally good results without having to go through any image preprocessing since this increases the computational complexity and the resources' demands. Moreover, another thing we want to examine is the training behavior while using a quite small dataset for supervised learning process.

Chapter 3

Machine Learning Theory

In this chapter, there will be a discussion around Machine Learning and Convolutional Neural Networks. Nowadays, there are more and more people that are using machine learning, for a great variety of reasons. Stating with predicting values, for example stock values or how the real estate market will be in a couple of years. Continuing with matters of life such as predicting the possibility of cancer for a patient or making a classification between a bleeding image and a non-bleeding image. As it is becoming obvious, being able to use machine learning in order to extract outcomes and predictions will become a part of our date to date tasks regardless of the field that one is working on.

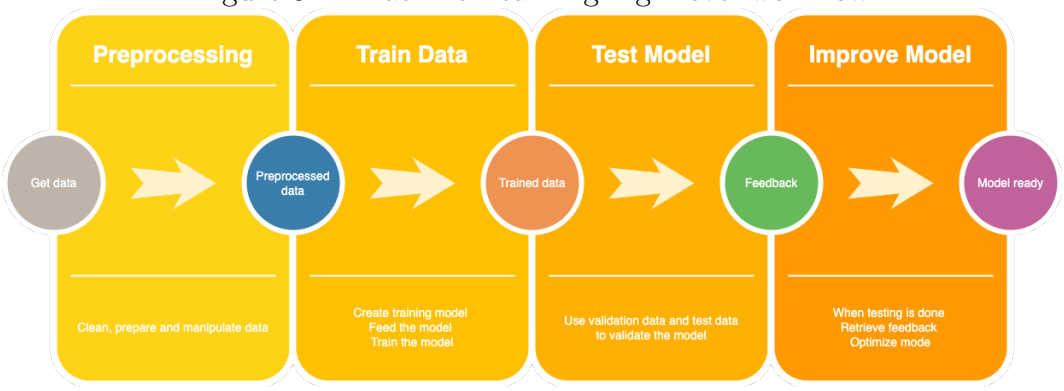
3.1 What is Machine Learning

A Machine Learning system will always have the same high level workflow. What will be different among the different methods and categories is actually what is taking place inside the different steps of the workflow.

The high level workflow that is taking place is the following:

1. Gather data
2. Process data
3. Create training model

Figure 3.1: Machine Learning high level workflow



4. Test training model

5. Optimize model

The high level workflow can be found as a graph in Figure 3.1.

Machine learning can be divided in three big categories:

1. Supervised Learning

2. Unsupervised Learning

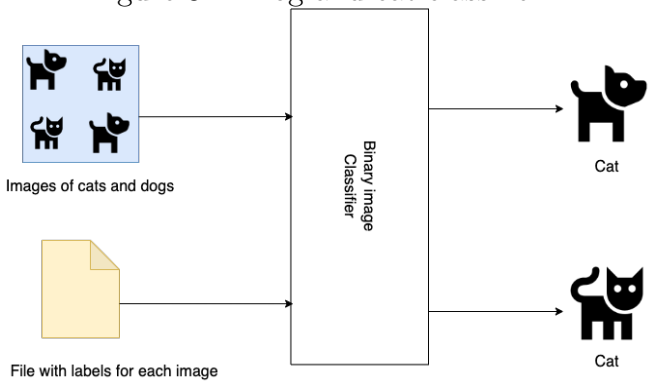
3. Reinforcement Learning

Let's see what each one is representing and when is better to use each.

3.1.1 Supervised Learning

The main difference of Supervised Learning among the rest of machine learning methods is in the training process. In Supervised learning as input data rather than the data that will be used, it is included also a file that contains the labels of the data that are received. For example if you are building a supervised learning model to classify images of cats and dogs, the input of the model will be the images of cats and dogs and a file that every image will be labeled with its corresponding category. That way you know what label will use before giving it to our algorithm [8].

Figure 3.2: Dog and cat classifier



The two main subcategories of supervised learning are classification and regression. **Classification** is used when the goal is to determine in which group a data instance belongs to. For example, let's say that you want to determine if an email belongs to the spam category or not, this is a binary classification problem. When the problem contains more than two possible categories, then we call it a multi-class classification. Recognizing letters, for instance, belongs to a multi-class classification problem. In Figure 3.2 there is an example of a binary image classifier.

Regression, is a method that allows us to predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x). Given the size of a house predict its price for the upcoming years (Figure 3.3) or predict the stock price, these two, could be regression problems.

3.1.2 Unsupervised Learning

When it comes to **Unsupervised Learning**, the data that are inserted in the training model, have no labels associated with them. A high level architecture of a cats and dogs classifier using unsupervised learning can be seen in Figure 3.4. The goal is to organize the data into that way that they could describe their structure. This could mean grouping them into clusters or finding different ways of looking at complex data so that they appear simpler (like quantization). Usually, since there is no prior knowledge regarding the data,

Figure 3.3: Prediction of house price based on their size for city of Branalle

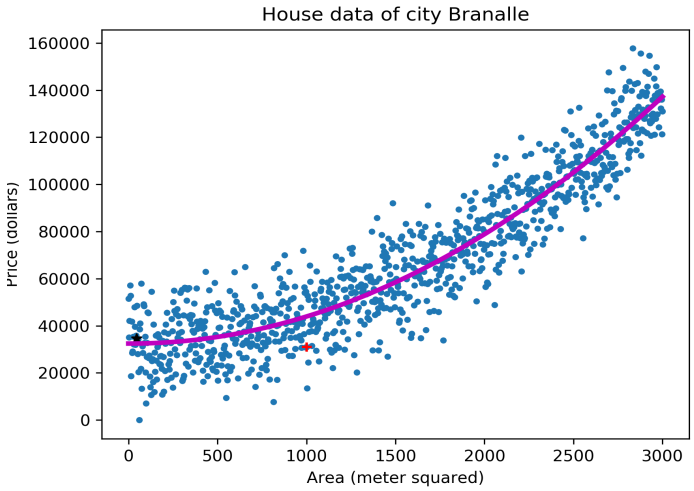
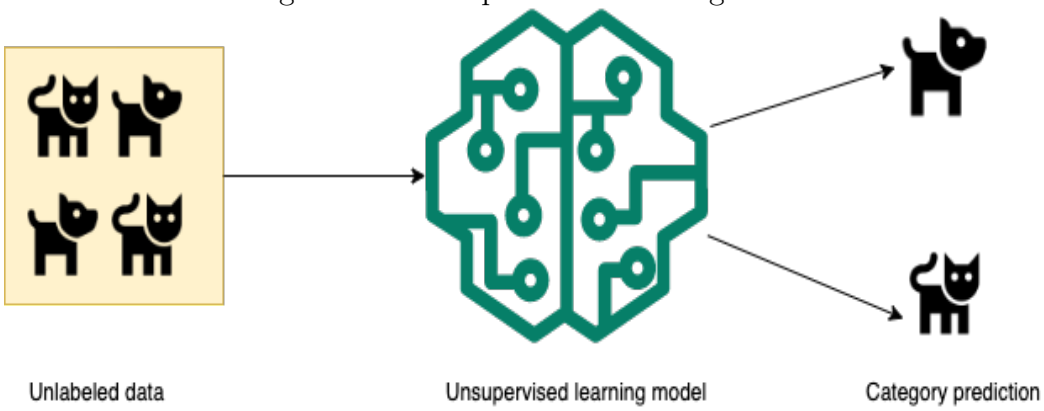


Figure 3.4: Unsupervised Learning model

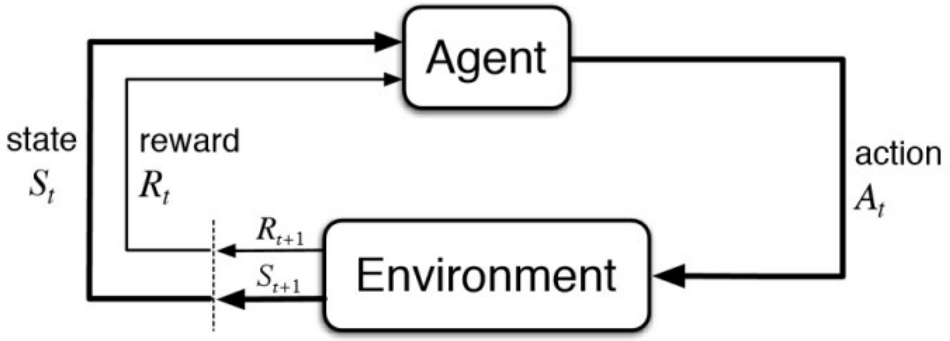


there is the need of big amount of data in order the training processes to be efficient [9].

3.1.3 Reinforcement Learning

The last category of Machine Learning is the **Reinforcement Learning**. Here there is an agent that improves the performance of the model based on the state of the environment during repeated interactions with an environment[10]. It is easier while trying to understand that process, thinking of it as a reward system with an agent that makes a series of decisions about how to interact with the environment and then evaluates those decisions he made, according to the state of the environments after the interactions.

Figure 3.5: Reinforcement Learning model



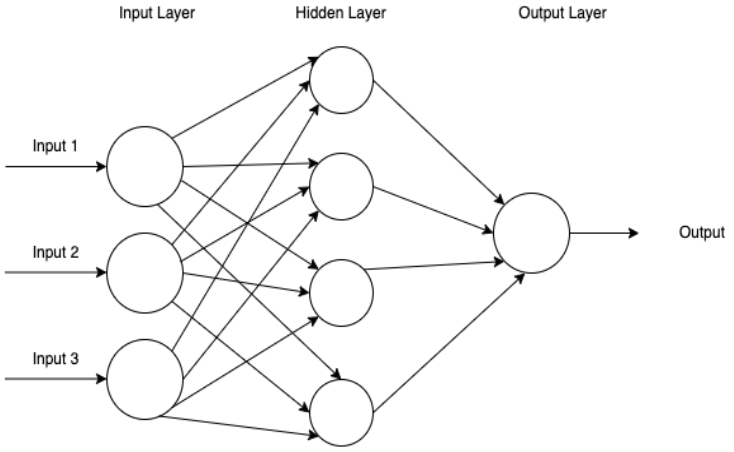
3.2 Convolutional Neural Networks

Convolutional Neural Networks were inspired by the biological processes that the connectivity pattern between the neurons resembles the organization of the animal visual cortex [11]. Usually they are fully connected networks, each neuron in one layer is connected with all neuron in the next layer. That property makes those networks sensitive when it comes to overfitting the model, there are techniques that are used so to overcome that, more information on that are coming later. Convolutional Neural Networks have many applications when it comes to image and voice processing:

- Image and video recognition
- Image classification
- Image Segmentation
- Medical Image Analysis
- Natural Language Processing

They took the name Convolution Neural Networks since they use the mathematical operation of convolution in at least one of the connected layers. That convolution acts as a filter of a general matrix multiplication designed to process the pixel data that are used for the image recognition [12].

Figure 3.6: Convolutional Neural Network High Level Design



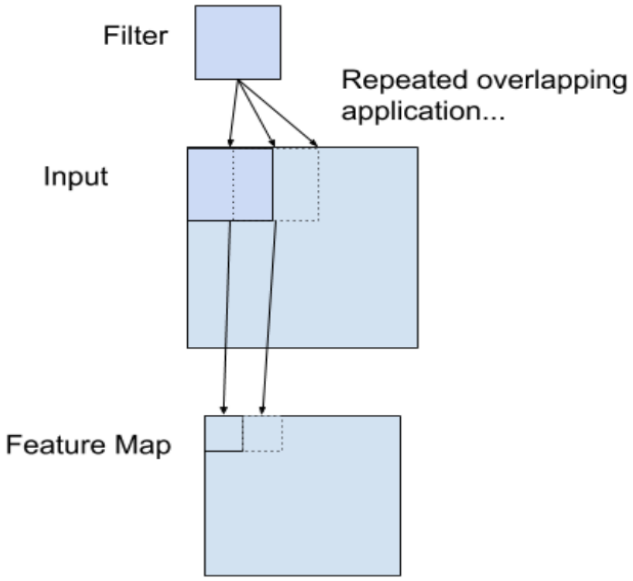
When it comes to the architecture of the models, they follow the same high level architecture. There is the input, later on what are called hidden layers and finally the output. Such a high level description can be found on Figure 3.5. When we refer to the depth of the model, then we speak about the number of hidden layers. When we refer to the width of the model, then that number refers to the number of units in each hidden layer. Input layer will take the raw input data and transfer them to the hidden layers. Hidden layers take input from one layer and transfer the output to another layer. Finally, the output layers will make the predictions.

In more depth, now is the time to see what exists inside the hidden layer or layers. Hidden layer, consists of multiple different layers that each one is fully connected with the next layer. Main components of those layers are, Convolutional layers, Pooling layers, Flatten layers and Dense layers. Depending on what you want to succeed and the techniques that one is using there can be used even more layers plus layers can be connected differently for each case.

3.2.1 Convolutional Layer

Starting with **Convolutional Layer**, they are the main building block of the Convolutional Neural Network. When it comes to image detection and recognition, convolution

Figure 3.7: Convolution over a two-dimensional image



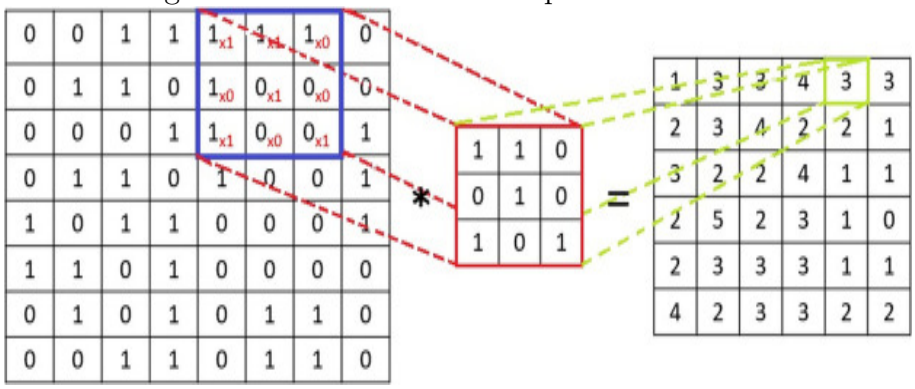
is the simple application of a filter over the image. The filter is smaller than the input data, and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. What is achieved through that is the creation of a feature map that summarizes the detected features of the input(Figure 3.6).

A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value (Figure 3.7), the operation is often referred to as the scalar product. Using a filter smaller than the input is intentional, as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different locations on the input. Specifically, the filter is applied systematically to each overlapping filter-sized patch of the input data, left to right, top to bottom. There is a sliding filter that will perform a convolution over the whole table of pixels of the image of interest[11].

3.2.2 Activation Functions

Another important element of the hidden layers and the Convolutional Neural Networks are the **Activation Functions**, also called transfer functions. Their role is to define how

Figure 3.8: Convolution example with matrices



the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. Usually the activation functions are non-linear functions and this is something desired since we can not stack together layers that are following all linear relations. Typically all hidden layers use the same Activation Function, the output layer will use a different activation function from the hidden layer and the type will depend on the detection type we want to perform. For example, is it a binary classification or a multi-class classification?[13]

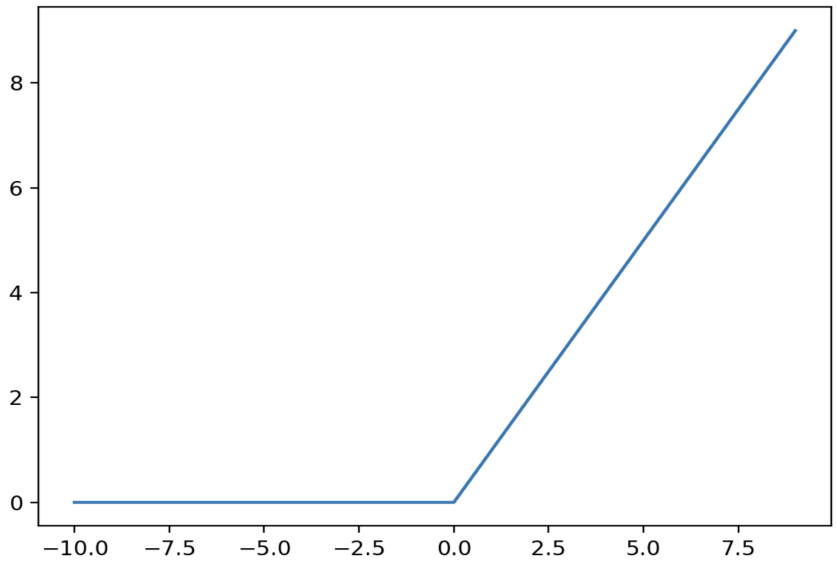
There are many types of activation functions used in convolutional neural networks, though some of them are becoming more often used for hidden and output layers. More commonly used activation functions for hidden layers [13]:

- ReLU Hidden Layer Activation Function
- Sigmoid Hidden Layer Activation Function
- Tanh Hidden Layer Activation Function

The most commonly used among the activation functions is **ReLU** or Rectified Linear Unit function. It became more famous since it is both easy and effective in overcoming the limitations of other activation functions. What will happen with ReLU is that if the input is positive, it will provide as output that one, otherwise the output will be zero (3.1).

$$ReLU(x) = \max\{0.0, x\} \tag{3.1}$$

Figure 3.9: ReLU activation function



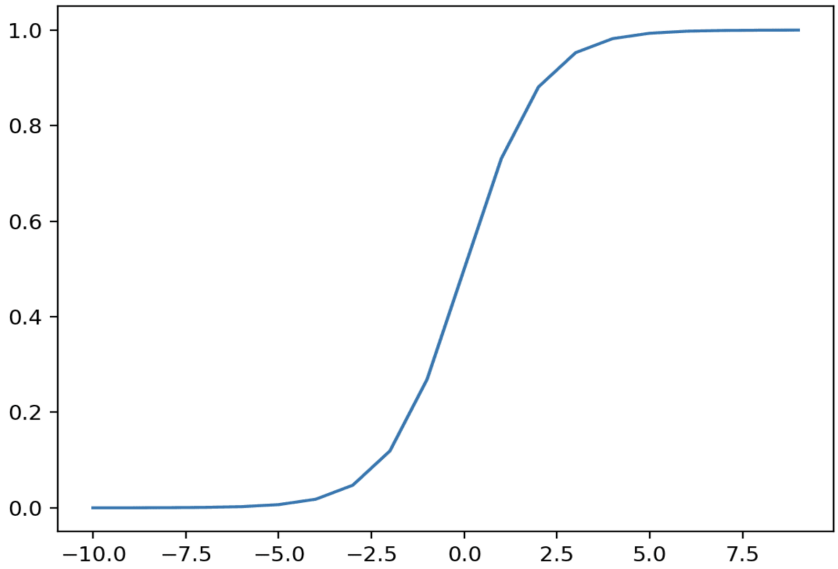
ReLU has become the default activation function for Convolutional Neural Networks, since it is working pretty good with deep neural networks [14]. A visual representation of the function can be found in Figure 3.8.

The **Sigmoid** activation function, also called the logistic function since it is the same function used for the logistic regression classification function. This function takes any input and transforms it in the range of [0,1]. As the input gets bigger the output value will be closer to 1 and respectively as the input gets smaller or negative then the output will be closer to 0 (3.2).

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

A graphical representation of Sigmoid function can be found in Figure 3.9. Although sigmoid function used to be the default activation function in the previous decade, now is presenting some drawbacks [15]. Sigmoid activation function can not be used in networks with many layers due to the vanishing gradient problem. If the input values are very big the output tends to be stacked in the value of 1 and similarly if the values are very small then the output tends to be in 0, that behavior is reducing the models' sensitivity since models are getting more sensitive in the change of the middle value(0.5). Another

Figure 3.10: Sigmoid activation function



disadvantage is that the model learning rate is low while using the sigmoid activation function.

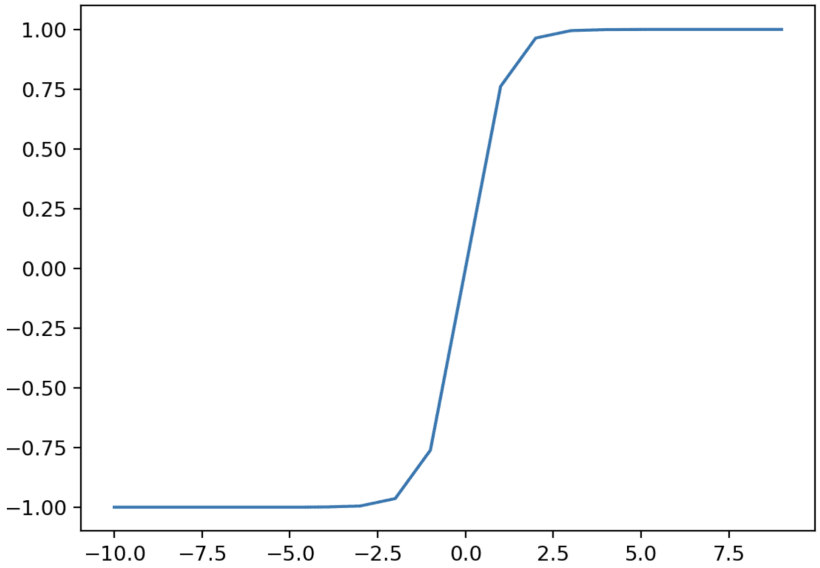
Lastly, another hidden layer activation function that is not that commonly used nowadays is **Tanh** or The hyperbolic tangent activation. That function is quite similar to sigmoid hidden layer activation function. This time, the function takes as input any real value and transforms it in a range of [-1,1]. As before, the higher the input then the output will be close to 1 and the lower the input then the output gets closer to -1(3.3).

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.3}$$

A graphical representation of Tanh hidden activation function can be found:
Tanh was getting more popular in the later 1990s and at the beginning of 2000s since it was providing better results when it comes to speed of training and the performance of the predictions than sigmoid that before was the state of the art for the activation functions [15]. Though it is carrying the same issues that were mentioned earlier, as the sigmoid function.

For those reasons, the most commonly used hidden layer activation function is ReLU.

Figure 3.11: Tanh activation function



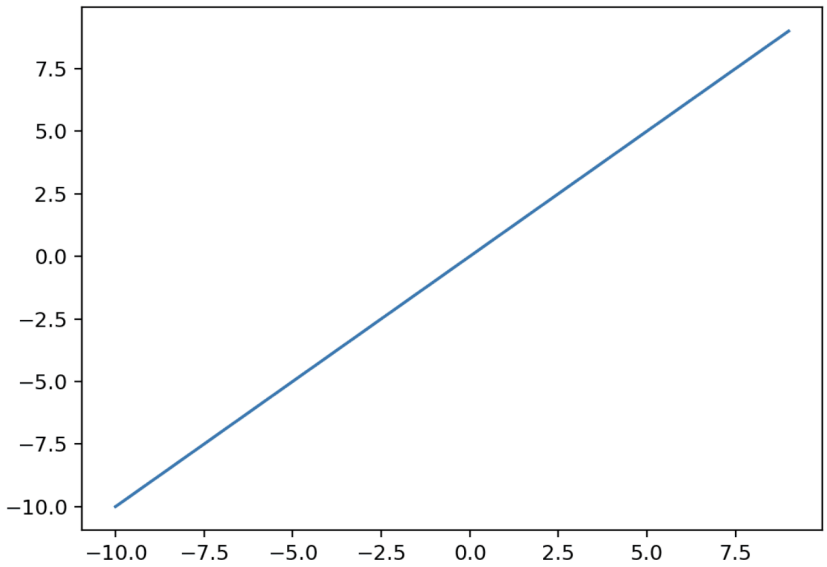
So far, we analysed the activation functions that are used inside the hidden layers. As was mentioned before, activation functions are also used in the **output layer**. Here there are three main activation functions and the decision on which one to use is coming based on the type of prediction you wish to make. Otherwise, based on the prediction problem you want to solve, then you select the most proper output layer activation function. The three most common ones[13]:

- Linear
- Sigmoid
- Softmax

Sigmoid activation function was analyzed before, therefore we will not go more through it. Worth mentioning that sigmoid is commonly used in binary classification problems. Think that you want to classify images that are taken using the endoscopy capsule in bleeding and non-bleeding images, then sigmoid can be used as an output layer activation function.

For the **Linear** output layer activation function, or called identity or no-activation. This is because the linear activation function does not change the weighted sum of the input

Figure 3.12: Linear output layer activation function



in any way, and instead returns the value directly[13]. Using that function there will be a diagonal line (Figure 3.11) draw and based on where the output stats (over or under the line) a prediction and final decision will be made. Also, another way to make a prediction using the linear function is based on the distance of the result from the line. Those two make linear activation function perfect when it comes to solving a regression problem.

Following, the **Softmax** output activation function receives as input a vector of real values and provides as output a vector of the same length as the input vector, with its values summing up to 1(Equation 3.4)[16].

$$Softmax(x) = \frac{e^x}{\sum e^x} \tag{3.4}$$

We can think of it as summing up the probabilities of each class. For example, if the experiment is about classifying fruits in the categories of apples, bananas and mango, then the output of the softmax activation function will be something like :

$$Softmax(apple) = 0.5 \tag{3.5}$$

$$Softmax(banana) = 0.3 \tag{3.6}$$

$$Softmax(mango) = 0.2 \tag{3.7}$$

Having the above example, if the problem we are trying to resolve is the one of classifying fruits among those three categories, then the output layer will come to the conclusion that the input image refers to an apple.

Therefore, Softmax is commonly used as an output layer activation function is problems such as image classification [17].

3.2.3 Pooling Layer

While trying to create the features' map using Convolutional Neural Networks so to conclude to a prediction, another important layer is the **Pooling** layer. That layer is used in order to reassure that only the most important features of the input will be used while training the model and concluding to a prediction [18]. Additionally, when it comes to image detection and recognition, pooling is important while detecting the edges and the lines. There are two main types of Pooling functions:

- Max Pooling
- Average Pooling

Pooling works by selecting a mask size, this mask size will be applied over the input matrix and depending on the pooling type either it will select the maximum value (Max Pooling) either the average of the values (Average Pooling). This is leading to a down sample and moreover having only the most important elements in the feature map.

For Max Pooling, The mask will slide to the matrix positions as many spaces as the mask size is and select the maximum value for that window. For example, in Figure 3.12, it is

Figure 3.13: Max Pooling for 2x2 mask

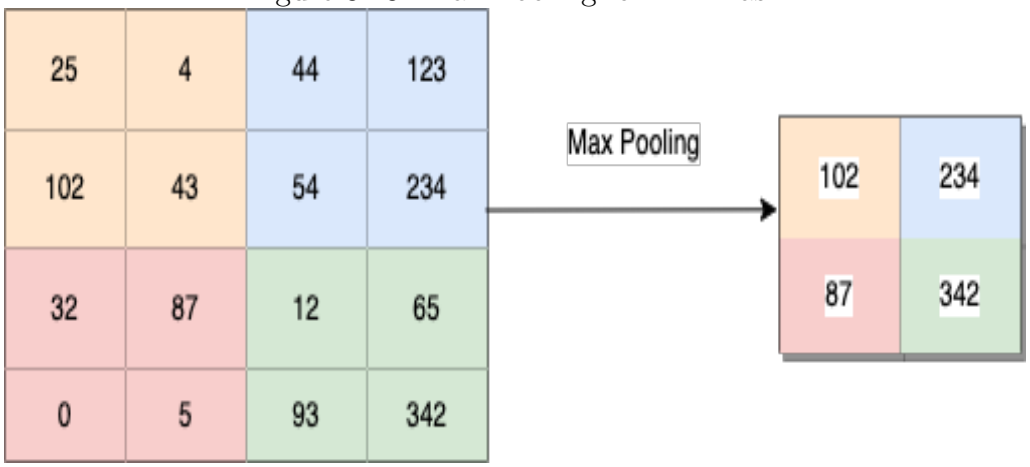
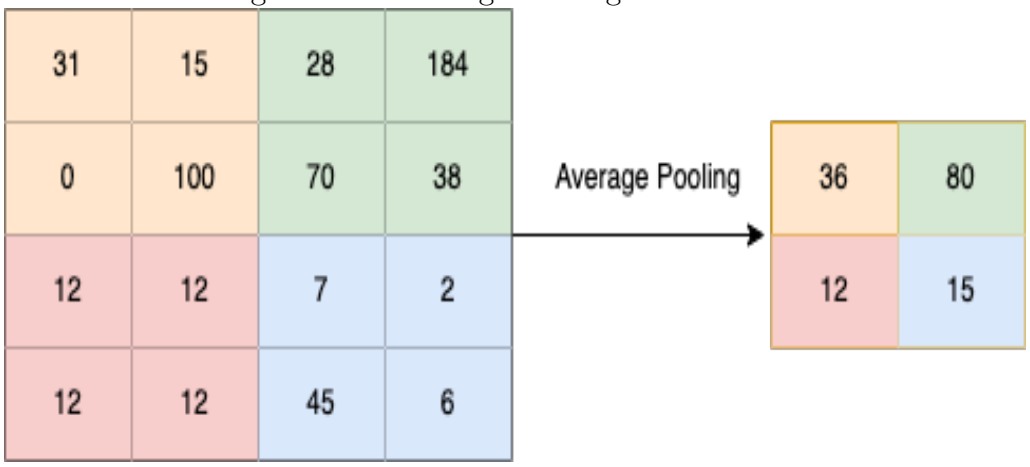


Figure 3.14: Average Pooling for 2x2 mask



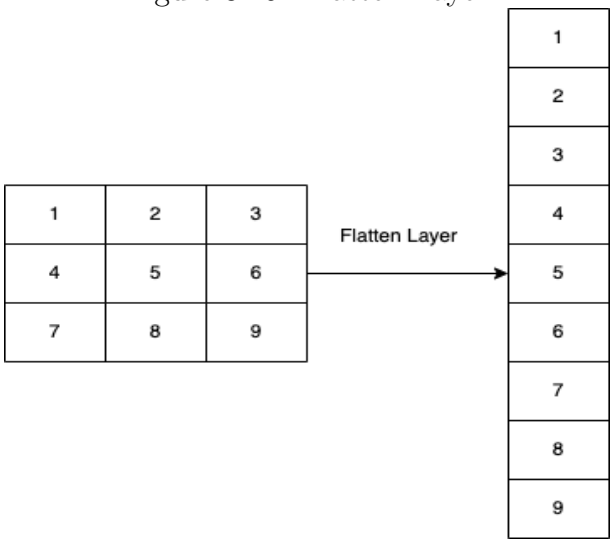
used in a mask of size 2x2. We can see that the original input is an array 4x4 and after using the 2x2 max pooling the output comes up as a 2x2 array with the maximum matrix values.

For the Average Pooling again the window will be moving across the matrix as many positions as the size of the mask with the difference that this time, the output will be the average value of the elements that the mask covers (Figure 3.13).

3.2.4 Flatten Layer

After convolutional and pooling layers are done, their output is a multidimensional matrix, its dimensions depend on the original input and their masks' sizes. In order to be able

Figure 3.15: Flatten Layer



to proceed with the prediction, there is a need to create a column out of those results (Figure 3.14). This is happening in the **flatten** layer. In simple words, in flatten layer, the data are converted in a single dimensional array. This is happening, so the data can be properly prepared to become input for the next layer [19]

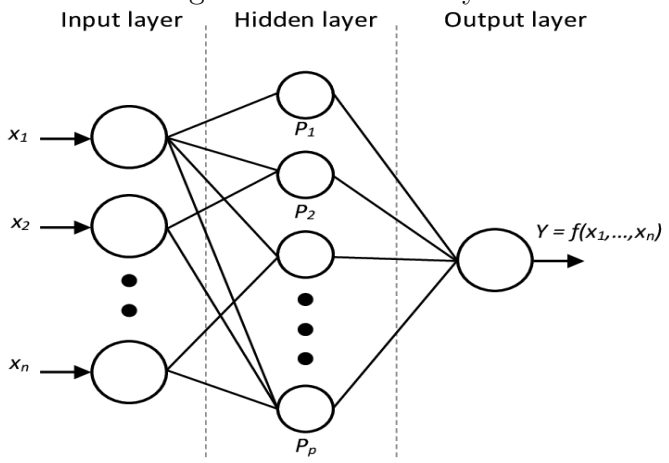
3.2.5 Dense Layer

Dense layer seems to be the most commonly used in artificial neural networks. It is a deep layer and a fully connected layer, meaning each neuron of previous layer is connected with every neuron in that layer. In the background, dense layer is performing a matrix-vector multiplication[20].

$$Output = activation(dot(input, kernel) + bias) \tag{3.8}$$

As we can see from the equation 3.8 the main ingredients of dense layer are the activation function, the kernel weights matrix, the input matrix and the bias vector. Dense requires the input to be a single-dimensional array i.e. 1-D array, that is the reason we are using Flatten layer that was mentioned before, one layer before the Dense Layer while building a Convolutional Neural Network model. Dense layer is used to classify an image based on output from the convolutional layers. The number of Dense layers to be used to verify with main differentiator to be the number of classes that we have to predict upon.

Figure 3.16: Dense Layer



3.2.6 Overfitting & Underfitting

One of the main problems that someone can face while experimenting with Machine Learning and Convolutional Neural Networks is overfitting and underfitting. **Overfitting** is when a model learns the data too well, that leads the model to perform very well with that dataset but failing to perform with an unknown image. **Underfitting** on the other hand happens when a model fails to sufficiently learn the problem and performs poorly on a dataset as well as while trying to make a prediction for an element outside the dataset [21]. Both behaviors are unwanted when it comes to our model and should be taken seriously under consideration and be resolved.

Overfitting of a model can be recognized when viewing the model's behavior during a single run, observing the performance curve of the model for each iteration that is running. Splitting the data into two datasets, one that will represent the training dataset and another that will be the validation dataset. When the results for the training dataset are significantly better than the validation dataset, then there the possibilities that your model is overfitting are high (Figure 3.17) [22].

You can recognize underfitting using the same technique, the difference now will be that now the model will have a great difference between the training dataset and the validation dataset with training dataset performing even iller than the validation dataset(Figure

Figure 3.17: Overfitting model

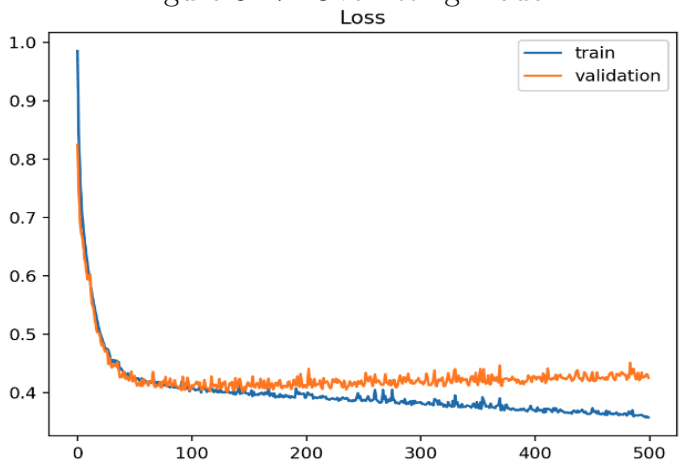
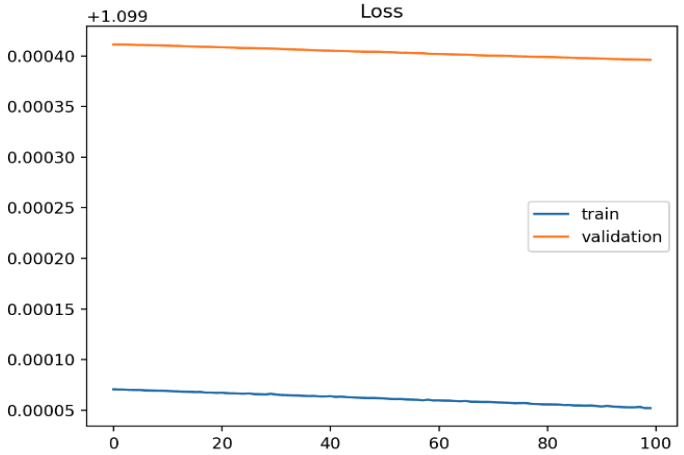


Figure 3.18: Underfitting model



3.18)[22].

While, you can observe in Figure 3.19 how a model with a good learning curves looks like. That is the goal for your model learning curve [22].

3.2.7 Dropout

One of the ways to resolve overfitting problems is using Dropout Layer in your model. What dropout will actually do is randomly dropping out (or ignoring) nodes during the training process(Figure 3.20). It is a computationally cheap and exceptionally effective way to reduce overfitting in deep neural networks [23]. By ignoring some nodes, dropout makes the training process noisy, forcing nodes within a layer to probabilistically take on

Figure 3.19: Good fit learning curve

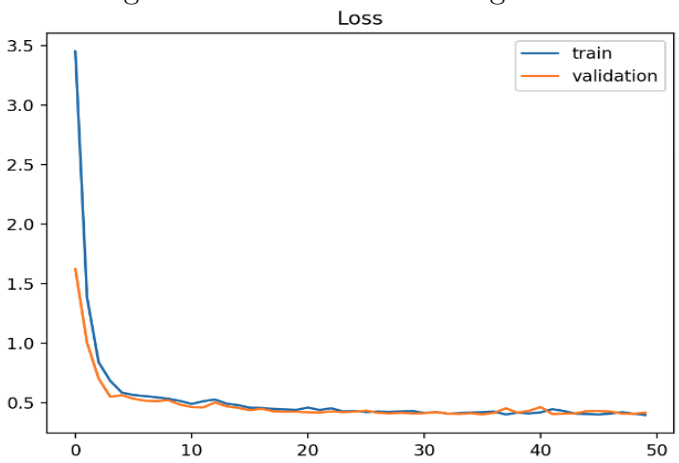
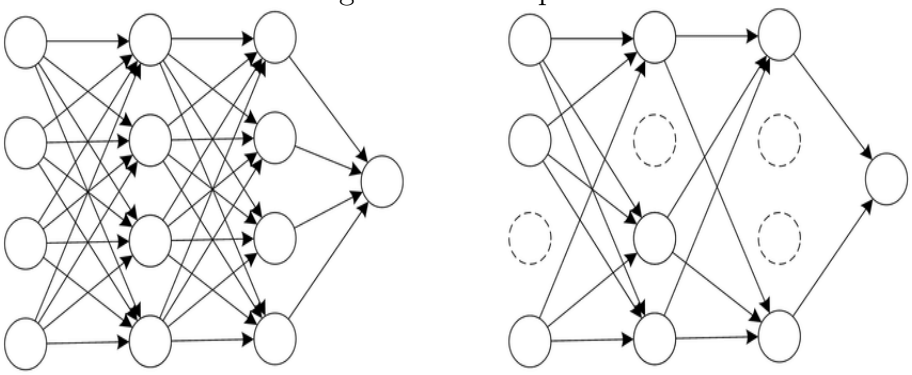


Figure 3.20: Dropout



more or less responsibility for the inputs [23]. Dropout can be implemented on any or all hidden layers in the Convolutional Neural Network as well as in the input layer. Though it cannot be used on the output layer.

As mentioned also before, it can be used with all types of neural networks. Dropout rate, is the hyperparameter that will define the percentage of nodes that will be dropped out. If dropout rate is 1.0 mean that there is no dropout, and 0.0 means that all nodes will be ignored. Usually a value between 0.5 and 0.8 is sufficient dropout rate for a hidden layer, input layers are using a larger dropout rate such as 0.8[24].

3.3 Tools and libraries

In this section, we explain tools and python libraries that were used in the implementation of this thesis.

3.3.1 Python

The programming language that we used is Python version 3.6. Python is a high level programming language that is widely used in the Machine Learning world, and it emphasizes in code readability with the use of significant indentation [25]. It is coming with many ready to use libraries and APIs for this purpose, such as Tensorlow and Keras API. Some other libraries that were used in order to be able to work with arrays and manipulate the images were NumPy and cv2(OpenCV is the newest version). Pandas library was also used, helping us while working with Excel sheets when calculating the results of the models and comparing the predictions with the real values of the images.

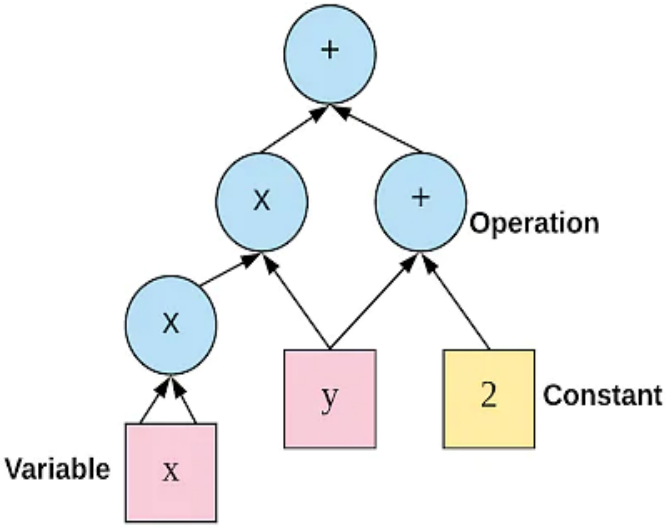
NumPy library is adding support for large, multidimensional arrays and matrices, along with high-level mathematical functions to operate on these areas [26] [27].

OpenCV is a library that supports image processing and manipulation. It was used in order to read the images from the files as well as in order to extract the frames from the videos[28].

3.3.2 Tensorflow

Tensorflow is an end-to-end open source platform for machine learning. It can be used for training and inference of deep neural networks. It was developed by the Google Brain team for initially for internal Google use in research and production. In cooperation with Keras API makes Machine Learning easier and faster in use and prototyping. It helps in model creation, in data loading and in the model training [29].

Figure 3.21: Tensorflow graph operations



Tensorflow accepts data in the form of multidimensional arrays or even higher dimensions called tensors, this is where it took the name from. Tensorflow supports both CPUs and GPUs Computing devices, training process requires high computational capabilities. When the activities are performed in a CPU level then it takes much longer time than when they are executed in a GPU, so one of the main advantages of Tensorflow is that it can work also with GPUs.

It allows you to create data flow graphs that describe how data moves through a graph. The graph's nodes represent mathematical operations. A connection or an edge between nodes is a multidimensional data array(Figure 3.22) [30]. So in simple words, Tensorflow is providing all the mathematical operations that need to be performed in the arrays or tensors that are loaded for the model creation, training and execution.

3.3.3 Keras API

Keras is running as a northbound interface of Tensorflow, taking easily and fast information by the user and providing them to Tensorflow library as input. It was developed with a focus on enabling fast experimentation. It is quite simple while using it, providing you more time and freedom into thinking the actual model architecture rather than the code that needs to come with it later on. It can support the implementation of small and deep

Figure 3.22: Model definition using Keras API

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

model = Sequential([
  layers.Conv2D(16,(3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, (3,3) , padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes, activation='softmax')
  ])
```

neural networks [31].

From Keras API, we used the APIs: Layers and Models and their functions. Layers API consists of all the necessary actions for layer creation, definitions of the core layers: Convolutional layers, Pooling layers, Dense Layers, Regularization Layers, etc [32] and their hyperparameters as well as the definitions of the activation functions. Models API [33] is used to create the models, there are three ways to create a Keras model:

- Sequential Model
- Function API
- Model subclassing

We used the Sequential Model type of creation, it is a straightforward way to create a model since you create a simple list of layers, example of a model creation using Keras Models and Layers APIs come in Figure 21. After model definition, you need to compile it, then build it and finally fit it in order to the training process to start. All those actions are easily provided by keras Models API.

3.3.4 Google Colaboratory

Google Colaboratory or Google Colab for short is a Google Research product that provides everyone with the ability to write and execute python code using Jupyter notebooks

and resources from CPU and GPU point of view that lay in the cloud. That way, you are able to use resource better than your local system and execute faster your machine learning training process. It is an open source tool, so everyone can use it [34].

You can save your files in the cloud, select the size of the resource you are going to need and there is no need of configuring them. Since your code will be hosted on cloud more precisely, all your Jupyter notebooks will be saved in Google Drive, it is very easy to share it with other developers.

Chapter 4

Design

4.1 Dataset

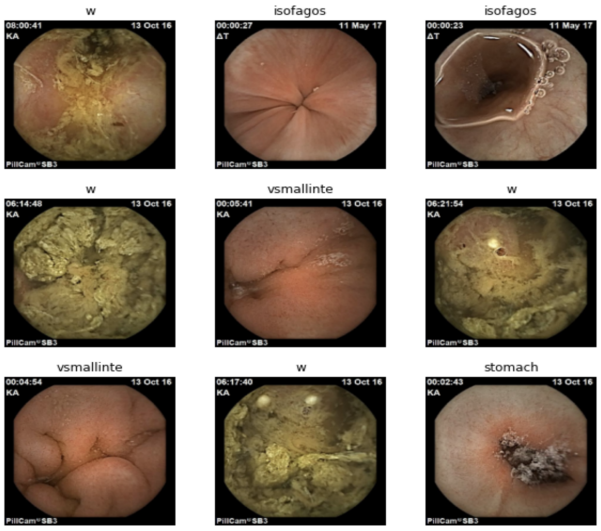
Data gathered for this Thesis are coming from the personal collection of Dr. Polydorou. All data refer to the period May 2016 - May 2017. Using videos coming from different patients, using the same dataset from videos that were used in thesis of Mr. A. Polydorou in 2018[35] extracted frames that represent each different organ of the digestive system (esophagus, stomach, small intestine, great intestine). In order to do that, the tool of Pillcam Reader was the video player and the information extractor.

After getting the smaller videos with the places of interest with Pillcam Reader, those videos were used in order to extract the frames to be used later on for both the training and testing data. That happened through a function created by the writer. With the frames per minute of the camera, there were 5 frames for each instance, so there was the need to select one image of those in order to downsample and not use duplicated data.

4.2 Data Preprocessing

All images are of the size 576x576, data are referring both to male and female patients.Based on the results, there was no need to perform any kind of preprocessing though

Figure 4.1: Sample images of different organs



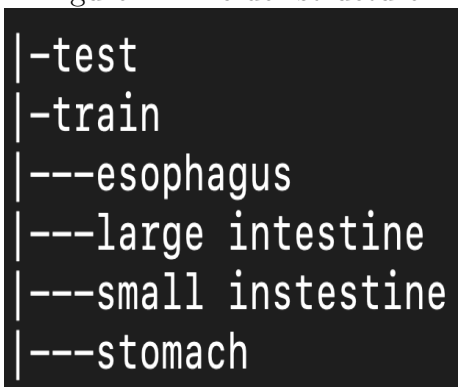
might be interesting for future work and investigation to implement the same Machine Learning Models in images from different spectrums, such as HSV and grayscale images even further in images as a signal in the frequency spectrum. For a sample of the images, refer to Figure 4.1.

After extracting the images in order to use them for machine learning purposes, images were divided in different folders. Foremost, they were distinguished in training and test data. Training data will be the once that will be fed into the model in the training phase, and testing are the ones to be used in order to verify that the model is producing comprehensive data. Later on, the train data were divided in 4 folders. Each folder represents one of the 4 points of interest, so for example, the stomach folder will contain only images that are coming from that area. Also for the training process there was created a CSV file in which there is a labeling of the images based on the category they belong and their name. Folder structure can also be found in Figure 4.2. All images are in three color channels (RBG - Red Blue Green). Pixel level are the typical ones: [0,255] although during the experiment the data are normalized.

Some famous data preprocessing techniques on Machine Learning are the following:

- **Image scaling**: When you handle square images, there are techniques to scale

Figure 4.2: Folder structure



each image based on your needs. There are prebuild functions in machine learning libraries that can support in down-scaling and up-scaling of the image.

- **Data Augmentation**: This technique is great when it comes to having small range of samples use during the training process. For example, in the current experiment case, the dataset size is very low.

- **Dimension Reduction**: In the current experiment, all images were in 3 color channels (RBG), there is also the option to implement the same experiment using 2 color channels (grayscale images). Someone may decide to use less color dimensions in order to reduce the computational cost.

- **Normalization of data**: So to ensure that all pixels follow a uniform data distribution. This will make the convergence faster in the training step of the machine learning process. This is achieved by subtracting the mean of each pixel and then diving that result by the standard deviation. That will lead to a Gaussian curve centered to zero. In case of images the possible values are only the positive ones, so it will move the data from [0,255] to [0,1]. This is also used in the current experiment.

- **Splitting** : Also in the case of our data, they were divided in the structure that is defined in Figure 4.2.

4.3 Machine Learning Parameters

In this section, there is a definition of basic machine learning parameters and how those affect the models and their results. Some of them and were defined in Chapter 3 (e.g. activation functions, dropout), so we will not go through them again. In this section, there will be a discussion on how different filter number and size effect convolutional layers, what different optimizers, loss functions and metrics bring when compiling the model and how epochs affect how the model fits.

When it comes to **filter size**, as discussed in Chapter 3, convolutional layer is supporting with feature extraction of the images. Using bigger filters it becomes computationally cheaper, but it covers fewer areas, meaning that it is reducing the accuracy and the possibility of observing a feature. Vice versa, smaller filters will have more work to do, and it will take more time, but it will provide better chances when it comes to catching all details[36]. Regarding the **number of filters**, convolutional neural networks learn multiple features in parallel for a given input and that is affected by the number of filters. For example, a convolutional neural network might be learning from 32 to 512 filters in parallel for a given input. This provides the model with the corresponding number of different ways of extracting features or many different ways of seeing an input[12]. As you can see in the following chapter we used different sizes and number of filters so to conclude to our best method.

Epochs is a hyperparameter which defines the number of times that the algorithm will work through the entire training dataset. You can think of it as a loop over the dataset, usually the number of epochs is big, allowing the algorithm to iterate over the dataset multiple times until it is the optimal. Creating a curve of loss and accuracy over the epochs of the training procedure is common and that curve is called learning curve[37].

Finally, other parameters worth of mentioning are the once that are inputs in the compile function of Keras: optimizer, loss function, metrics. Keras compile function comes with

an argument where you can specify the metric to be used while making a decision if the model works good or not. There are different types of metrics based on the prediction problem you are trying to resolve[38]. Since our case is a classification case, follows more information on them. The available Keras Classification Metrics:

- Binary Accuracy
- Categorical Accuracy
- Sparse Categorical Accuracy
- Top k Categorical Accuracy
- Sparse Top k Categorical Accuracy

Since we are resolving a multi-class classification problem in our study, we use Sparse Categorical Accuracy.

Regarding losses, there are even more that you can use and are divided in three bigger categories[39]:

- Probabilistic losses
- Regression losses
- Hinge losses for "maximum-margin" classification

In all of our models' compilation, we are using the Sparse Categorical Crossentropy which computes the cross entropy loss between the labels and the predictions. It is beneficial to use it in cases as ours that involves more than two labels.

Finally, the other hyperparameters of compile function worth of mentioning are the Optimizers. Optimizer is one of the two arguments required to compile a Keras model[40]. In our models, we used the optimizer with name: Adam. Adam is a stochastic gradient descent method[41].

Chapter 5

Implementation

For the current Thesis there were created and tested 3 different models of Convolutional Neural Networks:

- **Model 1**: Model with 3 layers of 2D convolution.
- **Model 2**: Model with 4 layers of 2D convolution.
- **Model 3**: Model with 4 layers of 2D convolution but different mask size.

All models will be described in detail in this chapter.

As was mentioned before, the main objective of this thesis is to be able to allocate the endoscopy capsule through image classification using machine learning techniques. In order to succeed that and create the models, similar researches and convolutional networks that are achieving the same were studied. The model tuning, parameters used for that as well as some technical terms will be analyzed in the first section of this chapter.

5.1 Before creating the models

Before model creation, there were some tasks to take care of. Firstly, as was mentioned above, since the original source was a video stream, it was needed to transform that stream into frames and downsample them so that we use 1 image at a time. For

that a function was created that could have as an input a video and the number of frames the user wants to extract and as an output creates a folder containing the desired number of images. In order to do that, main function for frame creation was $\text{video}_{c}aptureandinordertocreateandwritetheimageswascv2libraryandfunction : cv2.imwrite.$

Data are separated in two different big folders, one that contains the training data and the other that contains the testing data. The purpose of the training data folder is to host 4 subfolders, each one of those contains images of the digestive organ that represents (refer to Figure 4.1). There is a CSV file that contains two columns, the first column refers to the image ID and the second to the image class. Image classes were created using one hot encoding with the following principle:

digestive organ	code
esophagus	0
stomach	10
small intestine	11
large intestine	10

After creating the dataset as above, then the actual machine learning can start. Since for the experiment Google Colab was used, the folders and images were uploaded to Google Drive for easier access. After the images and folders were read, using the keras library and the preprocessing of images from directory function we can[42]:

- **Create validation data**: data are created using the 20 percent of the training data, they are useful in order to act as unknown data during the training process.
- **Reduce images size**: image size is reduced in order to be of 256x256 size.
- **Batch size**: Define the batch size to be 16.
- **Autotune data**: This will help our model to be faster, since it will decouple the time when data is produced from the time when data is consumed. It will use a background thread and an internal buffer to prefetch elements from the input

dataset ahead of the time they are requested. Autotune function will prompt the tf.data to tune the value dynamically at runtime.

- **Data normalization**: All pixel values were transformed in the interval of [0,1].

Now the dataset contains images of size 256x256 in 3 color channels (RGB) with batch size 16 and normalized in the interval of [0,1].

5.2 Model creation steps

The input of the models are pixel values in the shape of [WIDTH, HEIGHT, CHANNELS], in our cases specifically the size is [256x256x3].

Model creation is following the same structure:

1. **Define** the model: Here you define all the different layers that the network will have, defining also values such as filter size, number of filters, activation function, layer type.

2. **Compile** the model: After defining the model, you need to compile it in order to be configured with losses and metrics used.

3. **Fit** model: This way you start the training process.

4. Model **prediction**: Using that method you can start having predictions, anyhow this is the final outcome from model creation!

After following those principles, came the creation of all three model of our research. As you can see, we used the Sequential model for creating the model.

Model 1 (Figure 5.1), consists of Convolutional Layers that their size is same every time (3x3) and there is a different number of filters. Also, they use Maxpooling after each Convolutional layer. When the feature extraction is done, one Flatten layer and two Dense

Figure 5.1: Definition of Model 1

```
num_classes = 4

model = Sequential([
  layers.Conv2D(16,(3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, (3,3) , padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes, activation='softmax')
  ])
```

Figure 5.2: Definition of Model 2

```
num_classes = 4

model = Sequential([
        layers.Conv2D(32,(3,3), activation='relu'),
        layers.MaxPool2D((2,2)),
        layers.Conv2D(64,(3,3), activation='relu'),
        layers.MaxPool2D((2,2)),
        layers.Conv2D(128,(3,3), activation='relu'),
        layers.MaxPool2D((2,2)),
        layers.Conv2D(256,(3,3), activation='relu'),
        layers.MaxPool2D((2,2)),
        layers.Flatten(),
        layers.Dense(256,activation='relu'),
        layers.Dense(num_classes,activation='softmax'),
])
```

layers follow.

Model 2 (Figure 5.2), consists of Convolutional Layers that their size is same every time (3x3) and there is a different number of filters, this time starting with a larger and continuing with a bigger filter number (32,64,128 and 256). Again, there are Maxpooling layers after each Convolutional layer. As before, one Flatten layer and two Dense layers follow.

Model 3 (Figure 5.3), consists of Convolutional Layers with same number (32) and size (5x5). Again there are Maxpooling layers after each Convolutional layer. As before, one Flatten layer and two Dense layers follow. In that model there is also a Dropout layer,

Figure 5.3: Definition of Model 3

```
num_classes = 4

model = Sequential([
          layers.Conv2D(32,(5,5),activation='relu', input_shape= (256,256,3)),
          layers.MaxPool2D((2,2)),
          layers.Conv2D(32,(5,5),activation='relu', input_shape= (128,128,3)),
          layers.MaxPool2D((2,2)),
          layers.Conv2D(32,(5,5),activation='relu', input_shape= (64,64,3)),
          layers.MaxPool2D((2,2)),
          layers.Flatten(),
          layers.Dense(256,activation='relu'),
          layers.Dropout(0.2),
          layers.Dense(num_classes,activation='softmax')
])
```

Figure 5.4: Compile model

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

which is "dropping" nodes in a 20% sequence

After we define the models, we compile them using the Sparse Categorical Cross Entropy as loss function and accuracy as a metric (Figure 5.4). We use Adam function as an optimizer, since it is one of the best optimization algorithms for deep machine learning.

Lastly, we fit the model in order to initiate the training process. Using 30 epochs where each epoch is repeated 16 times.

Figure 5.5: Fit model

```
epochs=30
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs= epochs
)
```

Table 5.1: Model 1 Summary

Layer Name	Layer Type	Number of filters	Output Shape	Number of Parameters
Conv2D	Conv2D	16	(256,256,16)	448
Max_Pooling2D	MaxPooling2D		(128,128,16)	0
Conv2D_1	Conv2D	32	(128,128,32)	4640
Max_Pooling2D	MaxPooling2D		(64,64,32)	0
Conv2D_2	Conv2D	64	(64,64,32)	18496
Max_Pooling2D	MaxPooling2D		(32,32,64)	0
Flatten	Flatten		65536	0
Dense	Dense		128	8388736
Dense_1	Dense		4	516

A detailed description of the models comes in the next section.

5.3 Our models description

Model 1 consists of connected two-dimensional convolution layers and two-dimensional maxpooling layers as feature extractors, a flatten layer in order to have a vector of features after the last pooling layer that will be an input for the upcoming two dense layers that follow in order to make the final decision of the classification. In total, using that model, there were 8.412.836 trainable parameters. In the table 1, there are details regarding the different layers, as well as the sizes of the filters and the trainable parameters. Also in Figure 2 there is a graphical representation of Model 1.(Figure 5.1)

Model 2 is a deeper neural network using again connected two-dimensional convolution layers and two-dimensional maxpooling layers as feature extractors, a flatten layer in order to create a vector of features after the last pooling layer that will be an input for the upcoming two dense layers. In total, that model leads to 13.235.756 trainable parameters. In table2, there are details regarding output size of the different layers, as well as the sizes of the filters and the trainable parameters. In Figure 5.2 there is a graphical representation of the Model 2.

The third model is using convolutional layers with filters of the size (5x5) instead the

Figure 5.6: Structure of Model 1

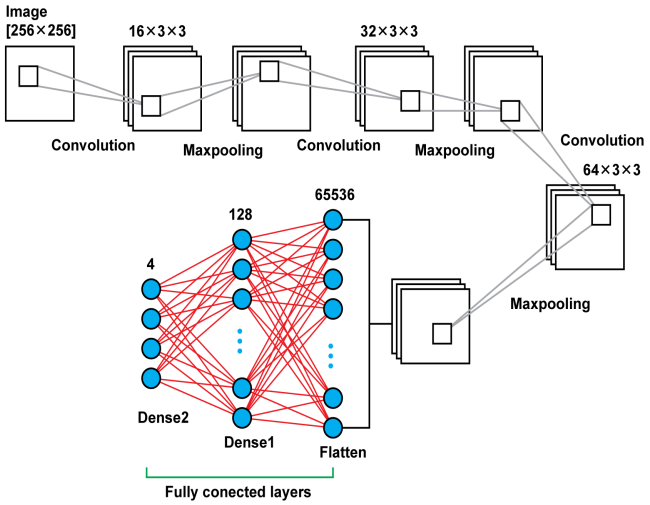


Table 5.2: Model 2 Summary

Layer Name	Layer Type	Number of filters	Output Shape	Number of Parameters
Conv2D	Conv2D	32	(256,256,32)	896
Max_Pooling2D	MaxPooling2D		(127,127,32)	0
Conv2D_1	Conv2D	64	(125,125,64)	18496
Max_Pooling2D	MaxPooling2D		(62,62,64)	0
Conv2D_2	Conv2D	128	(60,60,128)	73856
Max_Pooling2D	MaxPooling2D		(30,30,128)	0
Conv2D_3	Conv2D	256	(28,28,256)	295168
Max_Pooling2D	MaxPooling2D		(14,14,256)	0
Flatten	Flatten		50176	0
Dense	Dense		256	12845312
Dense_1	Dense		4	1028

Figure 5.7: Structure of Model 2

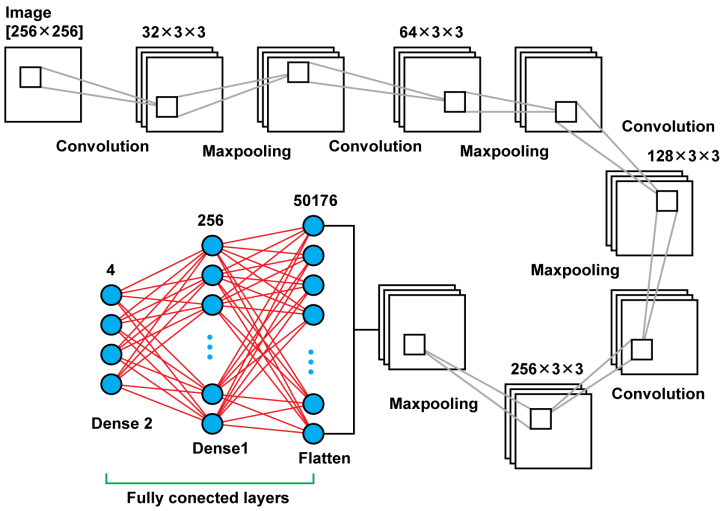
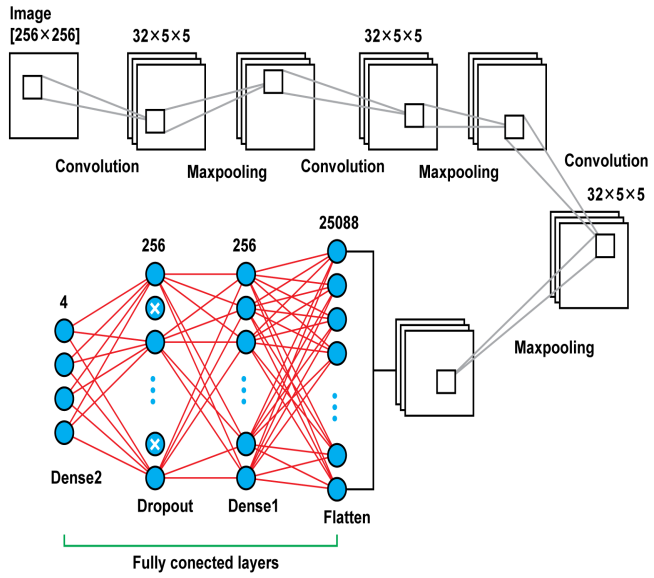


Table 5.3: Model 3 Summary

Layer Name	Layer Type	Number of filters	Output Shape	Number of Parameter
Conv2D	Conv2D	32	(252,252,32)	2432
Max_Pooling2D	MaxPooling2D		(126,126,32)	0
Conv2D_1	Conv2D	32	(122,122,32)	25632
Max_Pooling2D	MaxPooling2D		(61,61,32)	0
Conv2D_2	Conv2D	32	(57,57,32)	25632
Max_Pooling2D	MaxPooling2D		(28,28,32)	0
Flatten	Flatten		25088	0
Dense	Dense		256	6422784
Dropout	Dropout		256	0
Dense_1	Dense		4	1028

Figure 5.8: Structure of Model 3



filters of size (3x3) that were using the previous models. Also in the previous models there was an increase of the number of filters that were used as the network was getting deeper. For this model, there were used in all steps filters of the same size and number, in all convolutional layers there were used 32 filters of the size (5x5). Another difference of this model is that there was used also a Dropout layer between the last two dense layers. Rather than those two key differences the rest remain the same. In total that model results to 6.477.508 trainable parameters. In Figure 5.3 there is a graphical representation of model 3.

Chapter 6

Evaluation

In this chapter we examine the results of the models and specifically their performance.

6.1 Models comparison

Taking under consideration the small set of data that we are using for training as well as the fact that the data did not go under any pre-processing before getting to the training process, our results stand out. Following, there are references on the curves that refer to the loss and accuracy of training and validation data as per epoch of the training process. Those curves are provided as per Model.

All models used 30 epochs in order to conclude to a stable value of validation and training accuracy, this amount of epochs worked good enough to allow our training model to conclude to a good training converge. There was used also, instead of individual images, a patch of images of size 16. For the Model 1, training accuracy and validation accuracy start being stable after epoch 15, for Model 2 we see the same behavior after epoch 20 and for the Model 3 those values start being stable after epoch 25. The graphs for training loss, accuracy and validation loss and accuracy can be found for Model 1 at Figure 6.1, for Model 2 at Figure 6.2 and for Model 3 at Figure 6.3.

We can observe that even though there were used a small amount of images in order to

Figure 6.1: Loss and accuracy graph Model 1

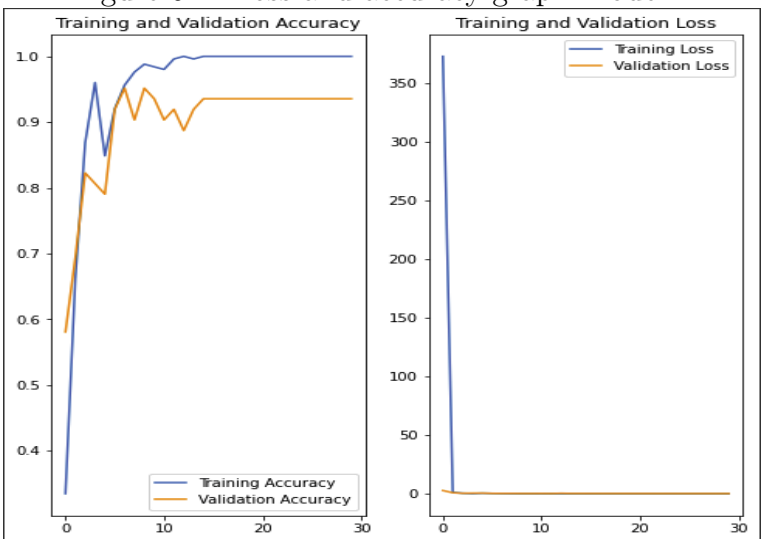


Figure 6.2: Loss and accuracy graph Model 2

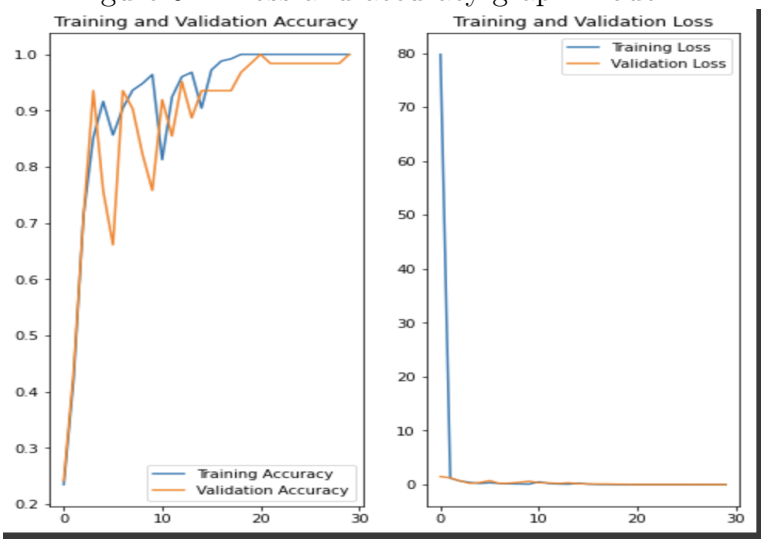
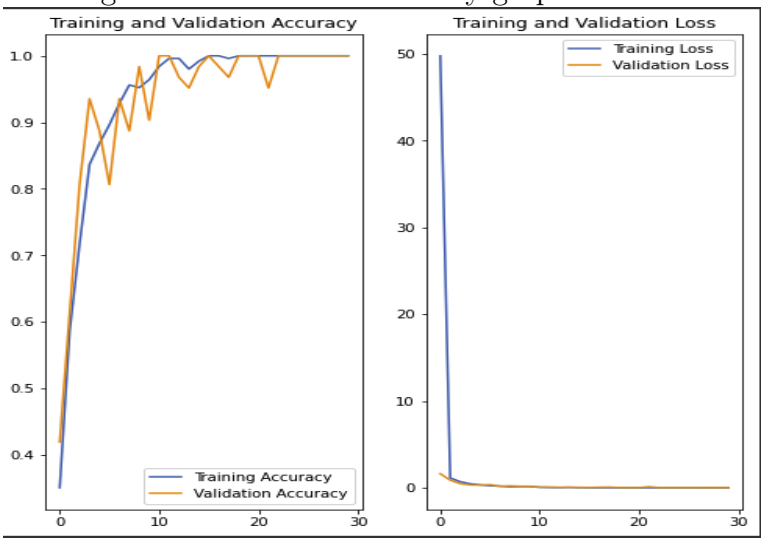


Figure 6.3: Loss and accuracy graph Model 3



train the models, after a relatively small amount of epochs, the results we are taking are pretty sufficient. Model 1 concludes with a training accuracy 1, and with a validation accuracy 96.77%. Model 2 provides a training accuracy 1 and a validation accuracy 96.77%. Moreover, Model 3 provides a training accuracy 1 and a validation accuracy 1. All those values refer to the results of 30th epoch. Among the three models the one that provides a more stable view as many times as the experiment was reproduced is the Model 2, we observe that increasing the number of filters as the model becomes deeper shows that is providing a more stable result.

Another part of analyzing the models is calculating the confusion matrix. Since our experiment refers to a multi-class classification in order to calculate the values of True Positive(TP), True Negative (TN), False Positive(FP), False Negative(FN) so that we conclude in the calculation of the rest of the metrics. All the metrics will be calculated as per class for each model for example in order to calculate the accuracy of predicting the frames that refer to esophagus for Model 1, the formula will be:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Accuracy is providing the overall accuracy of the model for that specific class. Meaning the fraction of the total class objects that were correctly classified by the classifiers. All the values will refer to the outcomes when it comes to esophagus frames while using Model 1 to predict the class of the frame.

Furthermore, with same way for class calculation we provide the value of Error Rate:

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \tag{6.2}$$

Error rate is calculated by the sum of all error predictions divided by the total number or predictions and indicates the quality of the classifiers towards the wrong predictions. Another metric is Precision, which is telling you what fraction of predictions as a positive class were actually positive. To calculate is, the formula is:

Table 6.1: Metrics of Model 1

Model 1	Accuracy	Error Rate	Precision	Specificity	Sensitivity
Esophagus	91.93%	8.06%	77.14%	91.57%	93.1%
Stomach	87.09%	12.9%	86.2%	95.4%	67.56%
Small Intestine	94.35%	5.64%	86.66%	95.78%	89.65%
Colon	99.19%	0.8%	96.66%	98.94%	100%

$$Presicion = \frac{TP}{TP + FP} \tag{6.3}$$

Specificity is the value that tells you what fraction of all negative samples are correctly predicted as negative by the classifier. To calculate it, use the formula:

$$Specificity = \frac{TN}{TN + FP} \tag{6.4}$$

Lastly, the value of Sensitivity or Recall, tells you what fraction of all positive samples were correctly predicted as positive by the classifier. To calculate that value, use:

$$Sensitivity = \frac{TP}{TP + TN} \tag{6.5}$$

After using the formulas above for each class and for each model, the results of the metrics when it comes to each model are presented in the tables below. For Model 1 follows in Table 6.1, we can observe that the model is working well for all organs, with slightly bigger error rate when it comes to stomach. After the examination of the error that we saw in the predictions, many times this error is coming because the presence of pituitary or of saliva is misunderstood as the pituitary of colon which is causing a confusion in the model.

Next, for model 2, the results of the analysis can be found in Table 6.2. As it coming from the table too, this model is the most accurate among the three when it comes to predictions, Error Rate is improved for stomach frames prediction and also provides the best metrics for esophagus frames prediction.

Lastly, Model 3 metrics appear in Table 6.3. As we can observe from the following table,

Table 6.2: Metrics of Model 2

Model 1	Accuracy	Error Rate	Precision	Specificity	Sensitivity
Esophagus	95.16%	4.83%	84.84%	94.73%	96.55%
Stomach	91.93%	8.06%	90.9%	96.55%	81.08%
Small Intestine	95.96%	4.03%	92.85%	97.89%	89.65%
Colon	99.19%	0.8%	96.66%	98.94%	100%

Table 6.3: Metrics of Model 3

Model 1	Accuracy	Error Rate	Precision	Specificity	Sensitivity
Esophagus	98.38%	1.61%	100%	100%	93.1%
Stomach	87.9%	12.09%	89.28%	96.55%	67.56%
Small Intestine	94.35%	5.64%	100%	100%	75.86%
Colon	85.48%	14.51%	61.7%	81.05%	100%

model 3 provides the biggest Error Rate in stomach frames prediction among the rest of the models. Although the predictions for esophagus and small intestine frames are having better metrics using that model.

Chapter 7

Summary and Reflections

Including a discussion of results in a wider context (considering other work).

7.1 Discussion of results

From the results, we can see that there is the possibility of reaching out to good results in the problem of localization of Wireless Endoscopy Capsule using machine learning and specifically Neural Convolutional Networks. Also, that can be succeeded without any image preprocessing, which is adding an extra computational cost, and with a quite small training dataset. That way the time of training is smaller and the resource needs of CPU and GPU are reduced.

7.2 Reflections and Future Work

Using our research as a localization and classification algorithm, physicians can recognize the digestive organ that the capsule is at the point. This can be used as a compass for navigating in the digestive system, as well as an energy saver tool. That can be achieved by loading the algorithm in the capsule and having an organ as the main point of interest, the endoscopy capsule can shoot fewer images if the capsule is not yet located in the desired area. Later on, by saving that energy can be configured to shoot more images

without the fear or running out of energy before the process is completed. Concluding, as a future work, we suggest the use of the algorithm for that purpose.

Another proposition for future work is the use of transfer learning, something that might reduce the learning process time even more.

Lastly, it would be interesting to use data augmentation in order to examine if using a larger input out of the same frames will increase the performance of the models.

Bibliography

[1] Medtronic. Pillcam SB3 by Medtronic.

[2] Amit Tanwar Virender Chauhan and Sandeep Nijhawan. Capsule endoscopy: What we know and what is new in the horizon. *J Digest Endosc*, 10 2019.

[3] GivenImaging. Pillcam Capsule Endoscopy - User manual.

[4] Poh Chee Khun, Zhang Zhuo, Liang Zi Yang, Liu Liyuan, and Liu Jiang. Feature selection and classification for wireless capsule endoscopic frames. *2009 International Conference on Biomedical and Pharmaceutical Engineering*, pages 1–6, 2009.

[5] Yuexian Zou, Lei Li, Yi Wang, Jiasheng Yu, Yi Li, and W. Deng. Classifying digestive organs in wireless capsule endoscopy images based on deep convolutional neural network. pages 1274–1278, 07 2015.

[6] Jeongkyu Lee, Jung-Hwan Oh, Subodh Shah, Xiaohui Yuan, and Shou-jiang Tang. Automatic classification of digestive organs in wireless capsule endoscopy videos. pages 1041–1045, 01 2007.

[7] Ran Zhou, Baopu Li, Hongmei Zhu, and Max Q.-H Meng. A novel method for capsule endoscopy video automatic segmentation. pages 3096–3101, 2013.

[8] Qiong Liu and Ying Wu. Supervised learning. 01 2012.

[9] Salim Dridi. Unsupervised learning - a systematic literature review. 12 2021.

[10] Ahmad Hammoudeh. A concise introduction to reinforcement learning. 02 2018.

[11] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

[12] Rahul Chauhan, Kamal Ghanshala, and R. Joshi. Convolutional neural network (cnn) for image detection and recognition. pages 278–282, 12 2018.

[13] Jason Browlee. How to Choose an Activation Function for Deep Learning.

[14] Jason Browlee. Rectified Linear Activation Function For Deep Learning Neural Networks.

[15] Data Science Central. Deep Learning Advantages Of ReLU over sigmoid function in deep learning.

[16] Wikipedia. Softmax Activation Function.

[17] DeepAI. Softmax.

[18] Muhamad Yani, S Irawan, and Casi Setianingsih. Application of transfer learning using convolutional neural network method for early detection of terry's nail. *Journal of Physics: Conference Series*, 1201:012052, 05 2019.

[19] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. 12 2014.

[20] AnalyticsInDiamag. Dense Layer.

[21] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022, 02 2019.

[22] Jason Browlee. Learning Curves For Diagnosing Machine Learning Model Performance.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[24] Jason Browlee. Dropout For Regularizing Deep Neural Networks.

[25] Wikipedia. Python.

[26] Wikipedia. NumPy.

[27] Organisation NumPy. NumPy Organisation.

[28] Org OpenCV. OpenCV Organisation.

[29] Wikipedia. Tensorflow.

[30] Simplilearn. Tensorflow Usage.

[31] API Keras. Keras API.

[32] API Keras. Keras Layers API.

[33] API Keras. Keras Models API.

[34] Google. Google Colaboratory.

[35] Alexios Polydorou, Eleftheria Sergaki, Andreas Polydorou, Christos Barbagiannis, Ioannis Vardiambasis, George Giakos, and Michail Zervakis. Improving cad hemorrhage detection in capsule endoscopy. *Journal of Biomedical Science and Engineering*, 14(03):103–118, 2021.

[36] Owais Khanday and Samad Dadvandipour. Convolution neural networks and impact of filter sizes on image classification. *Multidiszciplináris Tudományok*, 10:55–60, 04 2020.

[37] Jason Browlee. Difference Between a Batch and an Epoch in a Neural Network.

[38] Jason Browlee. How to Use Metrics for Deep Learning with Keras in Python.

[39] API Keras. Keras losses.

[40] API Keras. Keras optimizers.

[41] API Keras. Keras Adam optimizer.

[42] Christos Barbagiannis, Alexios Polydorou, Michail Zervakis, Andreas Polydorou, and Eleftheria Sergaki. Detection of angioectasias and haemorrhages incorporated into a multi-class classification tool for the gi tract anomalies by using binary cnns. *Journal of Biomedical Science and Engineering*, 14(12):402–414, 2021.

Appendix A

Dataset samples

Following you can see how the frames of different digestive organs that were used for the research look like.

Figure A.1: Esophagus images

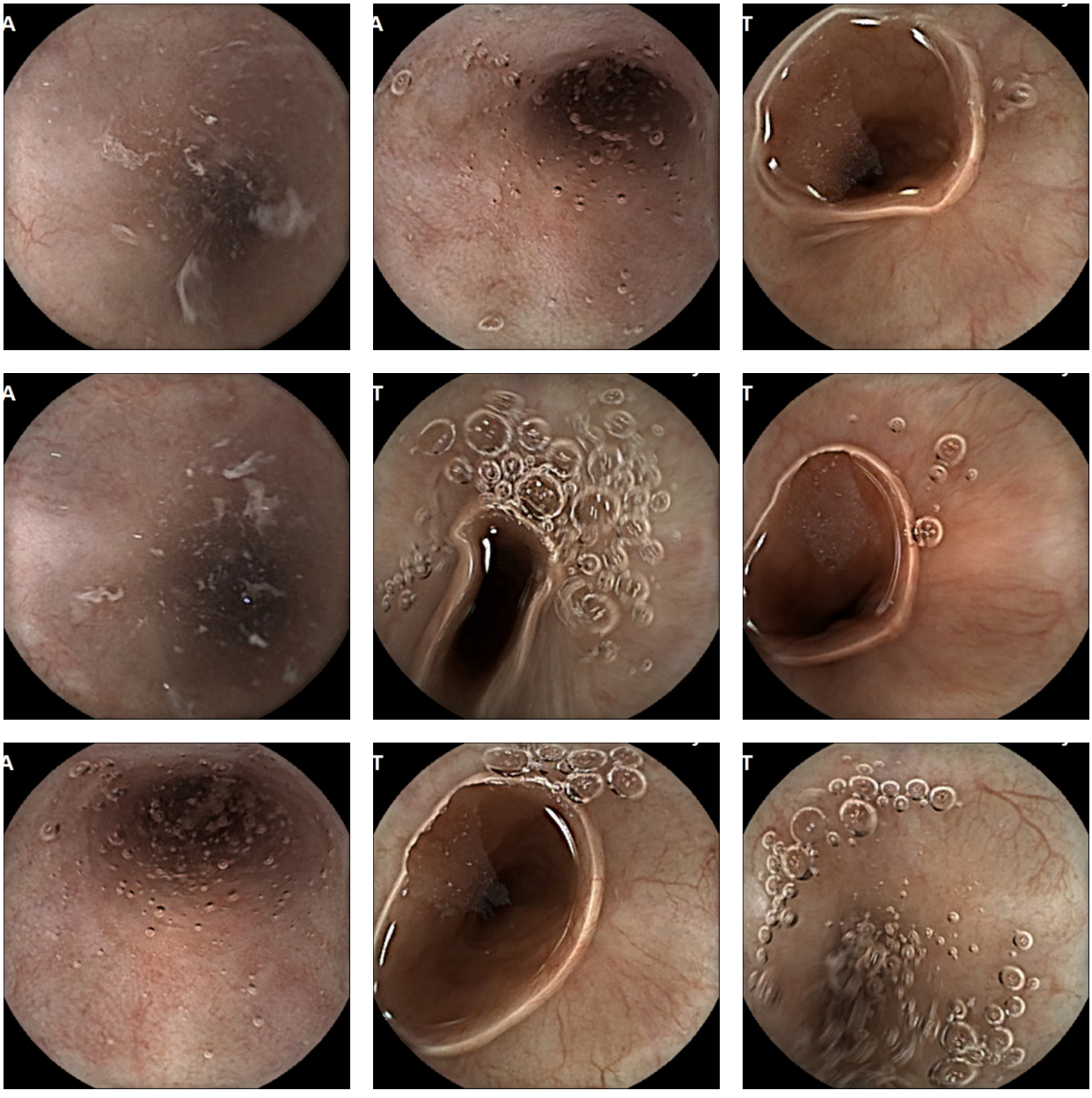


Figure A.2: Stomach images

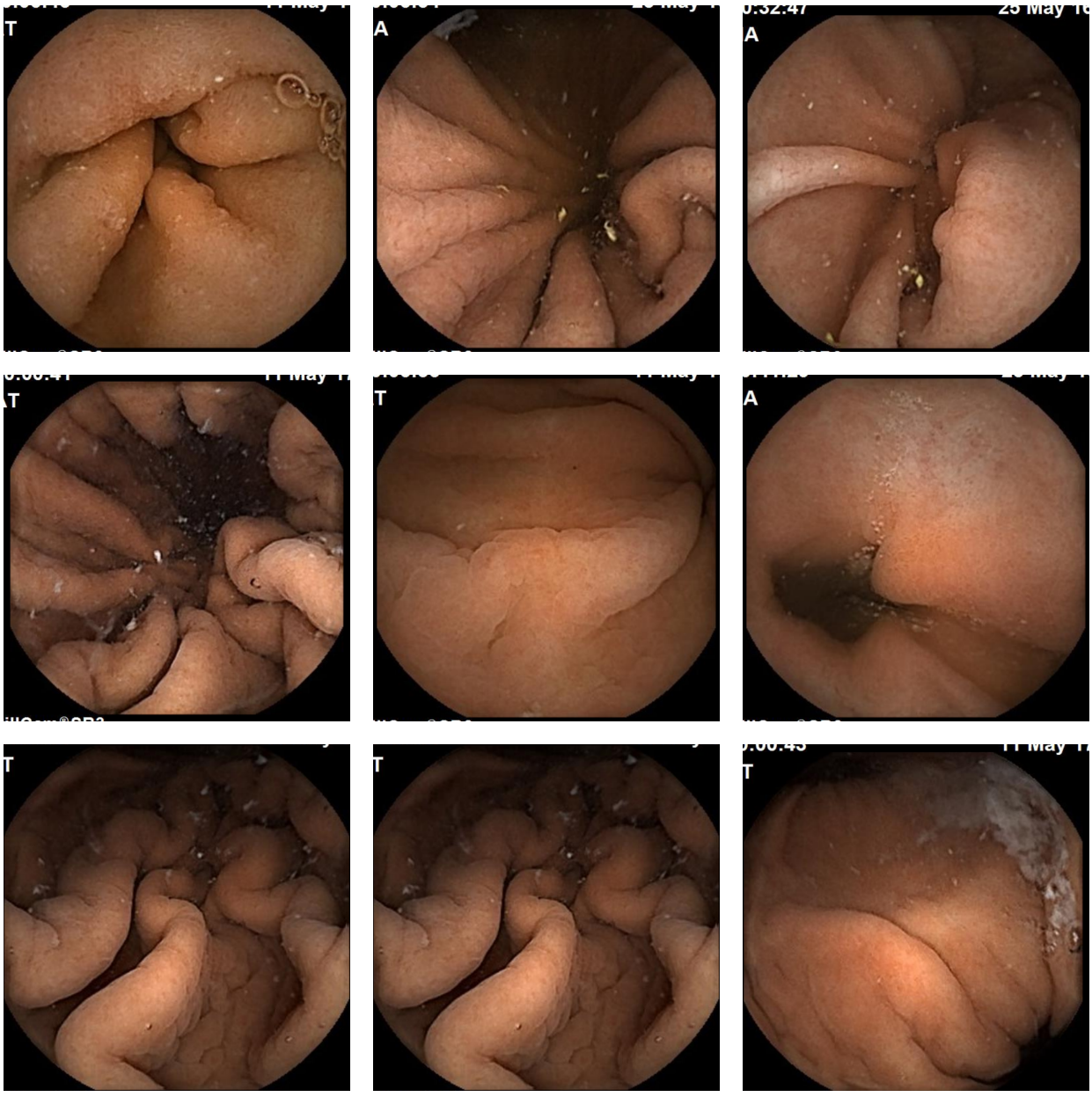


Figure A.3: Small intestine images

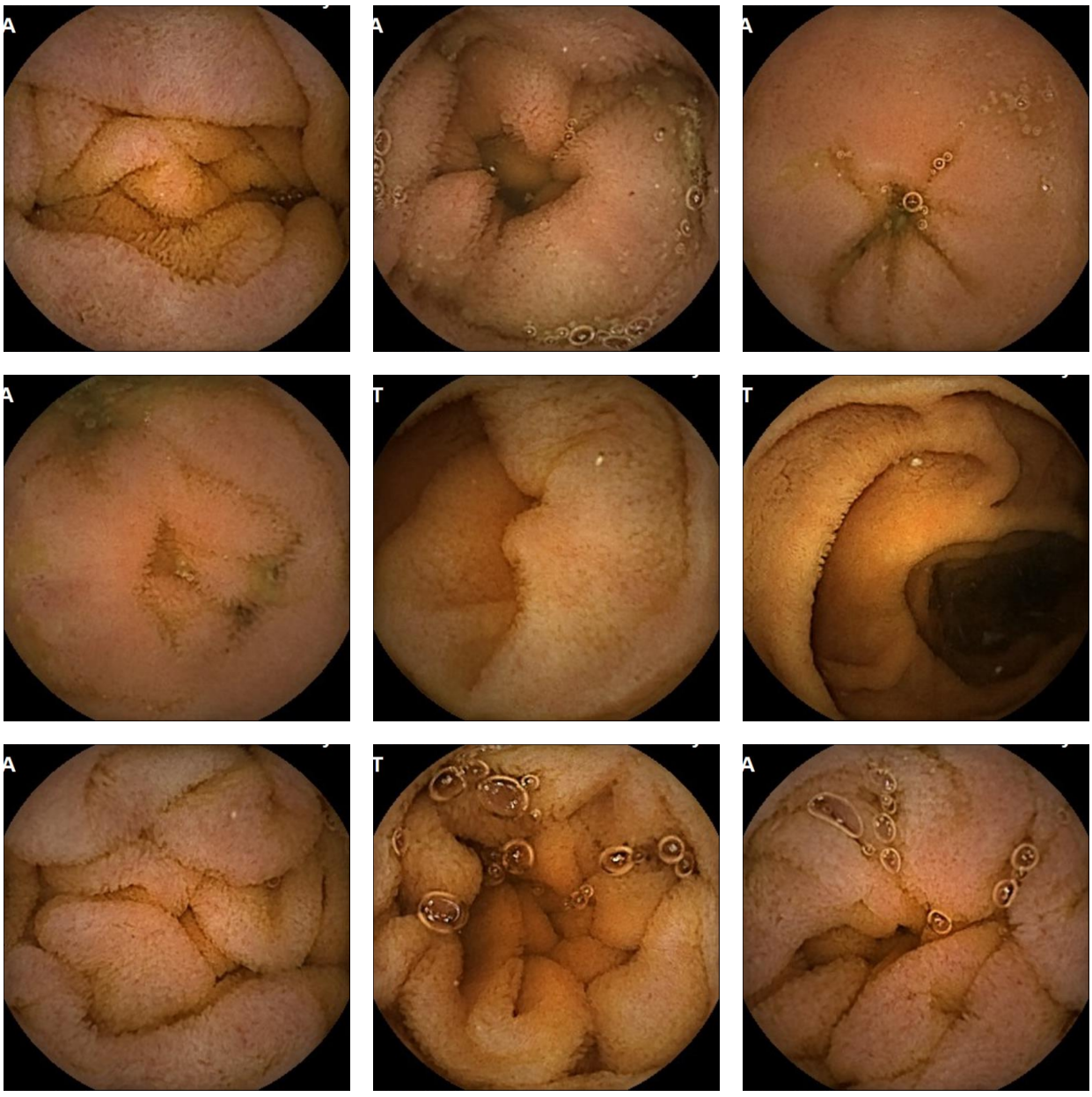
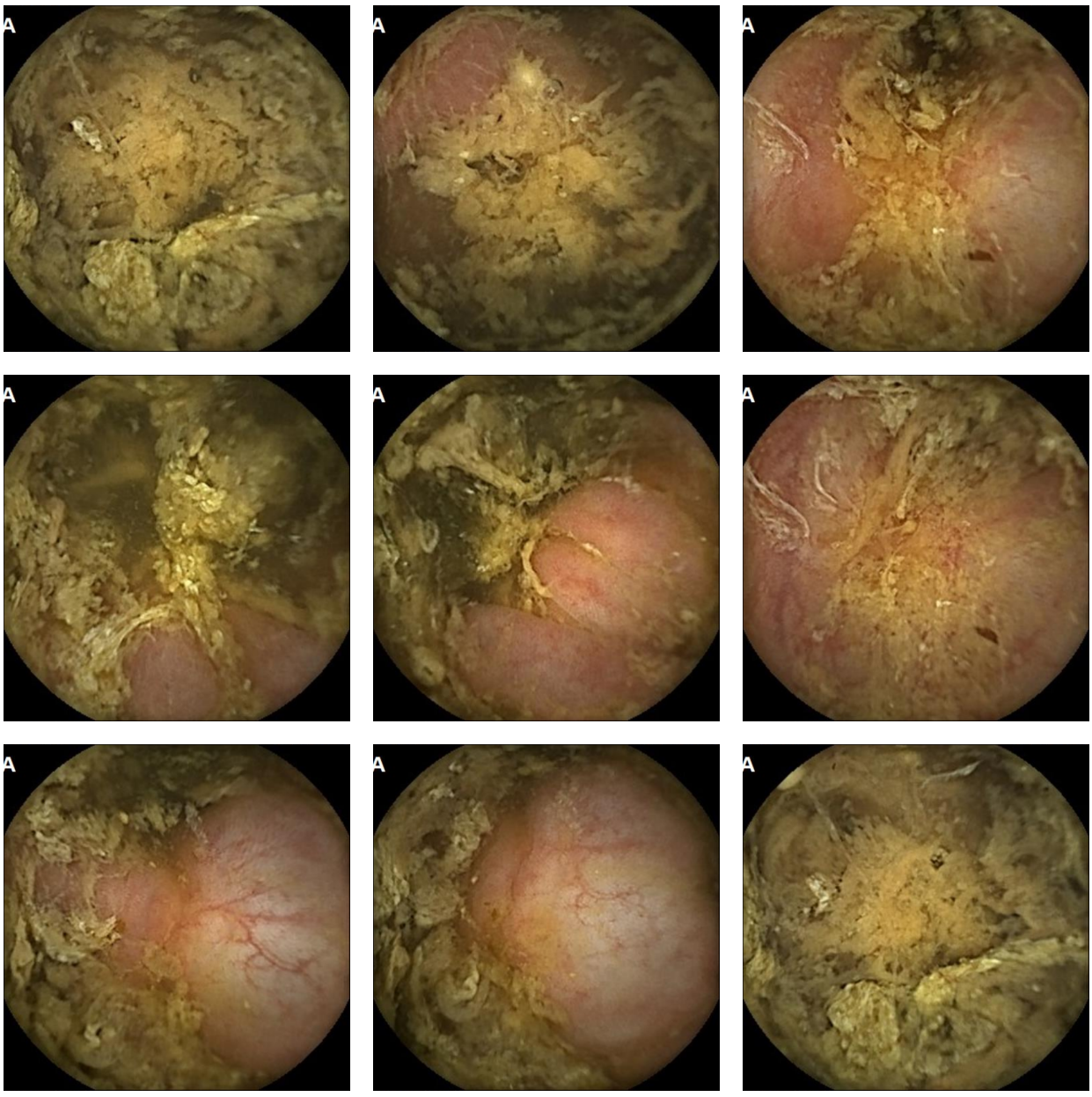


Figure A.4: Colon images



Appendix B

Accuracy and Loss graphs

Figure B.1: Model 1 Accuracy and Loss for training and validation data

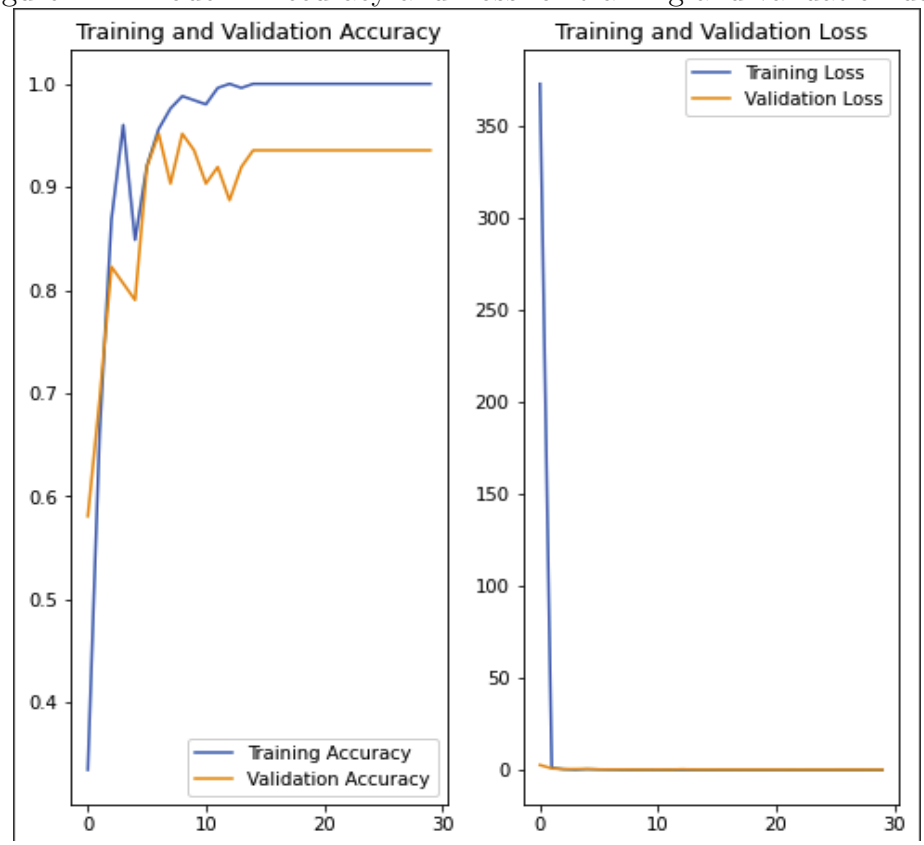


Figure B.2: Model 2 Accuracy and Loss for training and validation data

Figure B.3: Model 3 Accuracy and Loss for training and validation data