# Detection of COVID-19 disease from X-ray images using capsule-based network

by

Donya Ashtiani Haghighi

M.Sc., Tabriz University, 2012
B.Sc., University of Guilan, 2008

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

# Supervisory Committee

Detection COVID-19 disease from X-ray images using convolutional capsule networks

by

Donya Ashtiani Haghighi

M.Sc., Tabriz University, 2012
B.Sc., University of Guilan, 2008

**Supervisory Committee**

---

Dr. Amirali Baniasadi, Supervisor
(Department of Electrical and Computer Engineering)

---

Dr. Hong-Chuan Yang, Committee Member
(Department of Electrical and Computer Engineering)

# Abstract

The coronavirus (COVID-19) disease has spread abruptly all over the world since the end of 2019. The rapid and accurate diagnosis of COVID-19 is crucial for a better prognosis of this disease and breaking the chain of transition and flattening the epidemic curve. There are different types of COVID-19 diagnosis tests which sometimes have relatively low sensitivity. Computed tomography (CT) scans and X-ray images are other methods for the detection of this disease. However, one of the challenges of using these human-centered diagnosis methods is the overlap with other lung infections. Motivated by this challenge, different Deep Neural Network (DNN)-based diagnosis solutions have been developed, mainly based on Convolutional Neural Networks (CNNs), to accelerate the identification of COVID-19 cases. However, CNN's lose spatial information between image instances and require large datasets. In this project, an alternative framework based on Capsule Networks and Convolutional Neural networks is used which is able to handle small datasets. In addition, by investigating different parameters, the lowest loss of 0.0092, best accuracy of 0.9885, f1 score of 0.9883, the precision of 0.9859, recall of 0.9908, and Area Under the Curve (AUC) of 0.9948 is achieved when Plateau learning rate scheduler and margin loss function is used in capsule network. On the other hand, different dropout rates are used to decrease overfitting, and the dropout rate of 0.1 shows better results. In the last part, by removing one capsule layer and having far less number of trainable parameters 146,752 in comparison to the main architecture, it still shows promising results.

# Table of Contents

# List of Tables

# List of Figures

# Acknowledgments

I would like to thank my supervisor, Dr. Amirali Baniasadi, for his constant guidance, support and motivation for my project and throughout my program. I would also like to thank Pouya Shiri, for his continuous support and assistance all through my project.

# Dedication

Dedicated to my parents for their support, motivation and guidance.

# Chapter 1

# Introduction

## 1.1 Problem Statement

Since the end of 2019, the novel COVID-19 disease has been spread very fast all around the world and one of the crucial actions to avoid the spread of infection in the body is early detection of diseased cases, also it can accelerate the tracking of the people who were in contact with the sick people. By tracing these people, further spread of the pandemic can be avoided. COVID-19 infection can be revealed as lung infection, and for detection of any type of lung infection, computed tomography (CT) and chest X-ray (CXR) images are mainly used [1]. To help doctors, researchers are developing different methods for early detection of coronavirus infections and the most commonly used ones are artificial intelligence and deep learning methods. Deep learning methods have been widely used in many research applications, such as computer vision, object tracking [2], gesture recognition [3], face recognition [4], and steganography [5]–[7], and have shown significant performance compared to traditional methods. Convolutional neural networks (CNN) are typically used for image classification and are useful for mitigating overfitting. However, there are some challenges in CNNs to recognize the pose, texture, and deformations of a whole or part of an image [8]. Using a pooling layer is the reason for losing some features in the images and also translation-invariant behavior which leads to the inability in handling rotation and scale-invariance without explicit data augmentation. They are not equivariant and equivalence. Therefore, to have an effective performance and compensate for these issues, CNNs require a large amount of data based on the complexity of the task. In addition, the training time of CNNs is longer since they are more depth [9]. Also, due to adversarial attacks such as pixel perturbations, CNNs [10] have wrong classifications [11]. To overcome the aforementioned limitations of CNNs,

Capsule Network can be used. In addition, for training a model with the best possible features in this project, different changes in parameters and configurations of the Capsule network architecture are investigated. Models are evaluated with different metrics and compared with each other.

## 1.2 Related Work

In [12], authors presented COVID-Net which uses a new CNN architecture for classifying an open-source COVID CXR images into one of three classes. The lightweight residual projection-expansion projection extension (PEPX) based architecture is designed with two stages of projections, expansions, a depth-wise representation, and an extension. In addition, transfer learning weights generated by using the ImageNet dataset are used for training the CNN architecture on the COVIDx dataset.

In [13], the proposed model consists of a backbone, a classification part, and an anomaly detection part. In the first step, for extraction of high-level features from the input which are X-ray images, the pretrained backbone architecture on ImageNet is used and then fed to the classification and anomaly detection heads and in the last step, for every 'l' predictions, a cumulative score is used.

In [14], COVID-CAPS is a pretrained capsule network-based model with transfer learning using a publicly available dataset for detection of COVID cases from CXR and CT scan images which performs significantly with the small-size datasets. In [15], a new CNN model, DeTraC, is proposed which includes backbone architecture for feature extraction from images, decomposition part for training by the SGD optimizer, and class composition for classification. Another novel architecture COVIDLite is developed by using the depth-wise separable convolutional neural network (DSCNN) for the classification of CXR images [16]. In addition, using preprocessing step, (CLAHE) visibility and the white balance are improved which results in the color fidelity enhancement of the images. Another depth-wise separable convolutional neural network, the Fast COVID-19 detector (FCOD) is based on the inception architecture to decrease the computational complexity and computation time [17]. The other new CNN consists of one 16-filter convolutional layer, batch normalization, ReLU activation, and two fully connected layers with softmax classification and then compared with the pretrained Alexnet on the ImageNet dataset in [18]. In [19], a set of CNN models based on established architectures is proposed to detect normal, viral

pneumonia, and bacterial pneumonia classes, and an estimator for the infection rate is provided from the predictions. The other developed method in [20] is about using extracted features from two models (Xception and ResNet50V2) and then a convolutional layer and a classification layer to detect infected cases [20]. In [21], similar to the previous research, features are extracted from MobileNet, and then a global pooling layer, a fully connected layer, and a classifier are used for image classification. These models are trained with different methods such as fine-tuning, transfer learning, and training from scratch. In [22], CoroNet is proposed which is based on Xception. In addition, it is used a dropout and two fully connected layers to classify X-ray images into four classes: normal, bacterial pneumonia, viral pneumonia, and COVID-19 positive. In [23], the DarkCovid net along with transfer learning on the ImageNet dataset is used for object detection and includes fewer layers than Darknet-19 [24], average pooling, and softmax. In [25], exemplar-based pyramid feature generation, Relief, and iterative principal component analysis (PCA) analysis are used for feature extraction, and then a deep neural network (DNN) and an artificial neural network (ANN) are implemented for classification. In [26], a novel CNN architecture called CovXNet is designed including depth-wise convolutional layers trained from scratch with transfer learning and fine-tuning and then compared with various methods. In [27], unique CNN architectures without transfer learning are developed for binary classification and multiclass classification of chest X-ray images.

In [28], first deep features of images are extracted by using a pretrained model, and then, a linear support vector machine (SVM) and One-Vs-All SVM classifier are applied for classification. In [28], different pretrained architectures on the ImageNet dataset such as AlexNet, DenseNet201, GoogleNet, InceptionV3, ResNet18, ResNet50, ResNet101, VGG16, VGG19, XceptionNet, and InceptionResNetV2 [29] are used to extract the deep features and then to classify X-ray images, a different approach is applied. In [30], VGG-16, GoogleNet, and ResNet-50 are three networks for feature extraction and fusion. Then, the t-test method is used to decrease the redundancy of the features by ranking the features based on frequency. In the last layer, a binary SVM classifier is applied for classification. In [31], before feature extraction, a Gaussian filter is used for preprocessing raw data and then a depth-wise separable convolution neural network (DWS-CNN) is presented to extract the features from X-ray images. Finally, a deep support vector machine (DSVM) is applied for classification.

## 1.3 Agenda

The remaining chapters in this report are structured as follows. Chapter 2 gives an overview of Convolutional Neural networks and Capsule Network architectures. Chapter 3 outlines all parameters, dropout method, and metrics which are investigated in this project. Chapter 4 depicts the experimental evaluation of the proposed scheme. Chapter 5 makes concluding remarks.

# Chapter 2

# Background

Deep neural networks are used in different fields such as language translation, plant disease detection [32]–[34], facial (expression) recognition [35]–[37], image processing, and speech recognition [37]–[40].

## 2.1 Artificial Neural Networks and deep learning

One of the popular fields in Computer Science is AI which is based on neurons network in the brain. Artificial Intelligence networks include different layers, and neurons in each layer which are connected to neurons in the other layers by weighted synapses. In the learning process of networks, these weights are adjusted by backpropagation [41], [42]. The weighted sum of the inputs $x_i$ and weights $w_i$ is calculated as follows:

$$x = \sum_{i=1}^{n} w_i x_i \tag{2.1}$$

To find the optimized weights by training the model, the defined loss function for the model should be minimized which means how well the model performed its predictions. There are different types of loss functions and they include the loss-related part which is calculated based on the predictions and the ground truth, and regularization part which is used to prevent overfitting based on the model.

## 2.2 Convolutional Neural Networks (CNNs)

Computer vision applications are applicable and critical in different fields which are not feasible by a human being, such as real time product defects detection in manufacturing or defense and security based on analyzing video. Neural Networks (NN) is a method which can use enormous amounts of data [46] and deep Convolutional Neural Networks (CNN) [39], [42] surpass in areas of Computer Vision. They have been used for tasks such as disease detection [32], [47], [48], facial (expression) recognition [49], image processing, and speech recognition [38]–[40], [50]. The pivotal role of CNNs is to extract features automatically and handle data in different forms. The main layers of Convolutional Neural Network architecture can be found in the following figure:

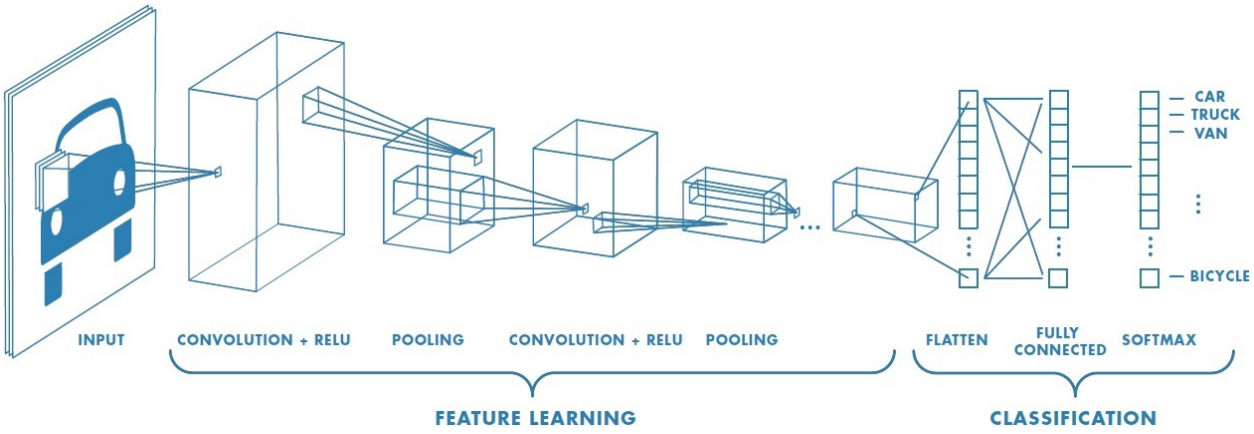



*Figure 1: Convolution Neural Network architecture*

Convolutional Neural Networks (CNN) [43] include convolutional layers, pooling layers, fully connected layers and also flattening part. A n×m kernel which is smaller than the input image is used to extract features from the input image by moving across the image based on a stride value. When a stride with a value greater than or equal to 2 is used, it results in dropping some features of the image. So, several kernels are used to achieve several feature extractions. ReLU [44], [45] is one of the popular activation functions which is applied after the convolutional layer for adding nonlinearity into the model and reducing model computational complexity.

### 2.2.1 Convolutional Layer

The most important part of CNN is the convolutional layer, which extracts features using a combination of linear and nonlinear processes [51]. In this layer, filters are applied to their inputs to generate feature maps. Each filter consists of weights which are trained during training models. Each convolutional layer has a bias which is used to map the feature independently. The number of weights in each convolutional layer is

$$P = W + \mathrm{B} = K^2 \times C \times N + N \tag{2.2}$$

Where $P$ is the number of weights, $W$ is the number of kernel weights, and $B$ is the number of the bias. $K$ is the kernel size, $C$ is the number of channels, and $N$ is the number of filters.

### 2.2.2 Padding

Padding is the process of adding elements on the edges of the first input image in the first layer to prevent the reduction of the input size as a result of the convolution process, which is called the border effect. The size of the output feature map is

$$O = \frac{W - K + 2P}{S} + 1 \tag{2.3}$$

Where $O$ is the feature map size, $K$ is the filter (kernel) size, $W$ is the input size, $P$ is the padding size, and $S$ is the stride which is related to the pooling layer.

### 2.2.3 Pooling Layer

The next layer is the pooling layer which is used for reducing the dimension of feature maps and as a result the memory requirements of the model can be reduced. It confirms that the same object in different forms of images can be recognized and also it results in spatial invariance, which is one of the major weaknesses of CNNs. This layer operates by sliding the pooling filter over the

convolved feature with a larger convolved map than the pooling filter. There are different types of pooling such as max pooling, min pooling, average pooling, and sum pooling [52].

#### 2.2.3.1 Max Pooling Layer

Max-pooling is one of the most popular pooling operations which extracts the best features of the feature map by keeping the largest value for each square segment and ignoring the other values in that segment. A popular max-pooling filter has a stride size of 2×2 and the feature map size is reduced by a factor of two [51].

#### 2.2.3.2 Average Pooling Layer

Average pooling is another common method which works by taking the average of all the values in each square segment of the feature map [53]. Some advantages of using an average pooling layer are lowering the number of learning parameters and taking inputs of varying sizes [51].

### 2.2.4 Activation functions

There is an activation function which controls the firing strength of a neuron. They add non-linearity to the network [54]. There are different types of activation functions such as sigmoid function, Rectified Linear Unit (ReLU) [55] (Eq. (2.4)), Soft-Max Activation Function [54], [56] (Eq. (2.6)) and the Hyperbolic Tangent function (tanh) (Eq. (2.8)).

#### 2.2.4.1 ReLu activation function

The ReLU activation function is useful when the vanishing gradient problem happens during training of the model. When the back-propagated gradient is so small, it cannot update the weights and this is the vanishing gradient problem. However, not having an upper limit on the output of ReLU is a disadvantage. Rectified Linear Unit (ReLU) formula is as follows:

$$\emptyset(x) = \max(x, 0) \tag{2.4}$$

There are the other versions of ReLU such as Leaky Rectified Linear Unit (Leaky ReLU) and Parametric Rectified Linear Unit (PReLU) (Eq. (2.5)) which solves the zero-gradient problem. Leaky ReLU outputs a small negative number for negative inputs which translate to a positive gradient and helps the neurons to recover when not being able to update.

$$\emptyset(x) = \max(0, x) + \beta \min(0, x) \tag{2.5}$$

Where $\beta$ is the parameter learned during training. When $\beta$ = -1, then $\emptyset(x) = x$ and this version of ReLU is called Absolute Value ReLU. When $\beta$ is very small, the activation function behaves as leaky ReLU [1].

### 2.2.4.2 Softmax

Softmax is usually used in the final layer of the CNN. Softmax maps the inputs to probabilities and sum of all probabilities is one. For calculation of Soft-Max activation function:

$$P(x) = \frac{e^{x^i}}{\sum_{j=1}^{n} e^{x^j}} \tag{2.6}$$

where $P(\emptyset(x))$ is the predicted probability for the *ith* class, *i* and *j* are related to the *ith* and *jth* classes and *n* is the number of output classes.

### 2.2.4.3 Sigmoid

The Sigmoid activation function outputs values between 0 and 1. The limited output of sigmoid is an advantage over ReLU. However, it can't solve the vanishing gradient problem when the input is small or large. Also, nonzero output of this function when input is zero, makes the gradient function move in different directions, which is a difficulty in optimization. Sigmoid function is calculated as follows:

$$\emptyset(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

Where $\emptyset(x)$ is constrained when $x \to \pm\infty$.

### 2.2.4.4 Tangent function (tanh)

tanh is given by:

$$\emptyset(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.8}$$

## 2.2.5 Fully Connected (FC) Layer

The main goal of a fully connected (FC) layer is to generate an output of 1D array of integers from a final feature map. During this process, weights of the final layer (feature map) are used to produce the desired outputs, which are the probability for each class in a classification task. The number of parameters in an FC layer is $(N + 1) \times M$, where $N$ is the number of input elements and $M$ is the output size.

## 2.2.6 Loss function

The equivalent of the cost function in ANNs is the loss function in CNNs. The most common loss function for classification is the cross-entropy function [57]. For regression, mean squared error is the best loss function.

# 2.3 Limitations of Convolutional Neural Networks

Convolutional neural networks are typically used for classification of high-dimensional data (images) and are useful for mitigating overfitting. However, there are some challenges in CNNs to recognize pose, texture and deformations of a whole or part of an image [8]. Using a pooling layer is the reason for losing some features in the images and also translation-invariant behavior,

which leads to inability in handling of rotation and scale-invariance without explicit data augmentation. They are not equivariant and equivalence. Therefore, to have effective performance and compensate for these issues, CNNs require a large amount of data based on the complexity of the task. In addition, training times for CNNs are longer due to the fact that they are more in depth [9]. Also, due to adversarial attacks such as pixel perturbations, CNNs [10] have wrong classifications [11].

## 2.4 Capsule Network

To overcome the aforementioned limitations of CNNs, Capsule Network can be used. The introduced Capsule Network architecture [8] is shown in Figure 2. In this method, there are a group of capsules, and each capsule includes neurons related to different information about any detectable object. This information is about position, rotation, scale, and so on. In an object, there is a high-dimensional vector space in which each dimension represents special features that can be understood intuitively. The main building block of the capsule network is related to the deconstruction of different hierarchical subparts of an object and to find a relationship between these subparts of the object. The main blocks of a capsule network are primary capsules, higher layer capsules and loss function, and all of these blocks have sub-operations which their descriptions are presented in the following sections.

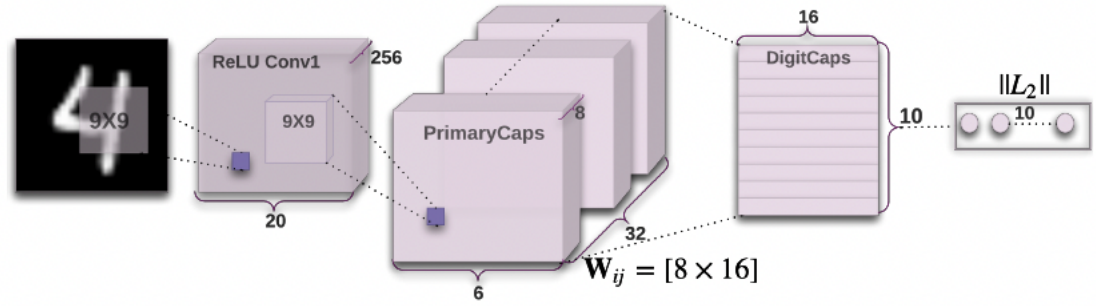



*Figure 2: CapsNet Encoder Architecture* [2]

## 2.4.1 Primary Capsules

In the first stage, a series of convolutional layers are used to extract an array of feature maps from the input image. To reshape feature maps into vectors, a reshaping function is applied. To keep the length of each vector equal to one, a nonlinear Squash function is applied in the last stage of this block. The maximum value of this function is one and the minimum value is zero [8]. This function can be described as follows:

$$d_j = \frac{\|p_j\|^2}{1 + \|p_j\|^2} \frac{p_j}{\|p_j\|} \qquad (2.9)$$

$$p_j = \sum_i a_{ij} \hat{v}_{ji}$$

$$\hat{v}_{ji} = A_{ij} \hat{v}_i \qquad (2.10)$$

$d_j$ is the vector output of capsule $j$ and $p_j$ is its input. The input capsule $p_j$ can be calculated by multiplying $a_{ij}$ which is set by the dynamic routing and the prediction vector $\hat{v}_{ji}$ which is the predicted output of the next capsules by multiplying the previous capsule's output $\hat{v}_i$ by a weighted matrix $A_{ij}$. $A_{ij}$ is the affine transformation matrix. This matrix is the main reason for robustness in the capsules.

The number of outputs in the capsule network are *N* vectors, which is the number of classes in the classification problem. Each element of these vectors is an instantiating parameter of the image. The length of each vector which can be calculated by $l^2$ norm is the existence probability of an entity in an image. The Squash function helps in normalizing the length of vectors between zero and one.

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i^2}} \qquad (2.11)$$

*x* is each element od vector.

## 2.4.2 Higher Layer Capsule

In this layer, an agreement technique is used to calculate the coupling coefficient $a_{ij}$, which is the amount of coupling between the higher-level capsule and the lower-level capsule and it is estimated using the softmax function:

$$a_{ij} = \frac{e^{c_{ij}}}{\sum_k e^{c_{ik}}} \tag{2.12}$$

Where $c_{ij}$ is the log probability. The initial value of $c_{ij}$ at the routing by agreement process is zero. In this process, the log probability $c_{ij}$ is updated based on the agreement between $d_j$ and $\hat{v}_i$, which is calculated as follows:

$$c_{ij} = d_j \hat{v}_i \tag{2.13}$$

To reconstruct the input image after higher layer capsule, an additional decoder network is used which recreates the input image by minimizing the squared error between the reconstructed image and the input image. There are three fully connected layers in this decoder, two rectified linear unit-activated units, and then the sigmoid-activated layer to decode the output vectors. The capsule network decoder can be found in figure 3.

## 2.4.3 Loss Function

In Capsule Network, a separate loss function called margin loss $L_{cap}$ is used for providing intraclass compactness and interclass separability in each digit capsule t. Therefore, the higher margin loss specifies that the prediction of each category in each capsule is not correct [58]. It is defined as

$$L_{cap} = W_t \max(0, n^+ - \|v_t\|)^2 + \lambda(1 - W_t)\max(0, \|d_t\| - n^-)^2 \tag{2.14}$$

Whenever class *t* is actually present, $W_t$ is one for correct predictions and otherwise it is zero. $n^+$, $n^-$, and λ are hyperparameters that should be tuned for the learning process. $\lambda$ is the weight for penalizing wrong predictions, and $n^+$ and $n^-$ are used to remove capsules with high or low probabilities from the margin loss. Also, during the reconstruction process in a decoder network, a reconstruction loss $R_t$ is computed, which is the mean square error between the input image and the reconstructed image:

$$R_T = \|I_R - I_I\|_2^2 \tag{2.15}$$

$I_R$ is reconstructed image and $I_I$ is input image data. The total loss function can be provided by:

$$L = (L_{cap} + \alpha R_t) \tag{2.16}$$

$\alpha$ is used for tuning the effect of reconstruction loss and importance of the margin loss in the total loss. Desirable loss function is the one with domination of the margin loss in the training process. In addition, this loss exploits as a regularizer to avoid overfitting in the training process.

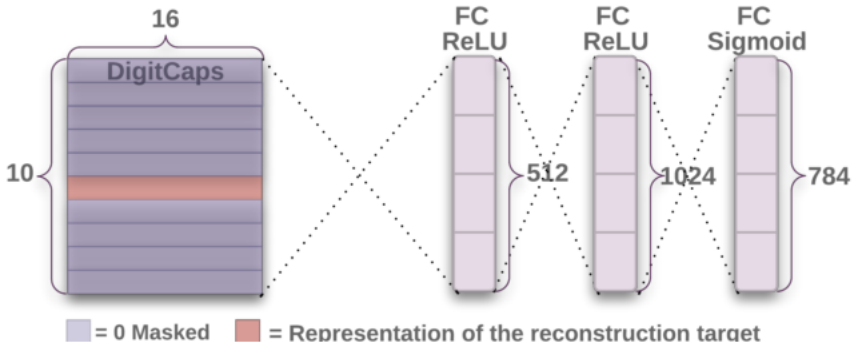



*Figure 3: CapsNet Decoder Architecture* [2]

### 2.4.4 Dynamic Routing Algorithm

Dynamic routing algorithm provides a stable training by determining the place of the output vectors in the first layer of capsules in their next layer. For each input, parameters for dynamic routing should be determined. In addition, this algorithm is a method for checking the degree of agreement between the predictions of the first layer.

*Table 1: Dynamic Routing Algorithm* [2]

**Algorithm 1** Dynamic Routing Algorithm

**procedure** ROUTING($u_{j|i}, l, r$)
  $b_{ij} \leftarrow 0$ ▷ for all capsules $i$ and $j$ in layers $l$ and $l+1$
  **for** r iterations **do**
    $c_i \leftarrow SoftMax(b_i)$ ▷ for all capsule $i$ in layer $l$
    $v_j = Squash(\sum_i c_{ij}\hat{u}_{j|i})$ ▷ for all capsule j in layer l+1
    $b_{ij} = b_{ij} + \hat{u}_{j|i}.v_j$ ▷ for all capsule i and j in layers l and l+1
  **return** $v_j$

### 2.5 Advantages of CapsNets over CNNs

CapsNet encodes the spatial information of the image and generates vectors for different categories. The relationship between the different levels of features such as edges and blobs and the whole objects and the amount of agreement are determined by using Dynamic Routing algorithm.

The other advantage of CapsNets is the robustness over affine transformations applied to input images, and it performs well even without training on a particular set of transformations. Capsule Networks also perform better on detecting overlapping images [8].

### 2.6 CapsNet Drawbacks

Capsule Networks show some drawbacks. Training time for Capsule networks are longer due to multidimensional matrix multiplications and the Dynamic Routing Algorithm. In addition, Dynamic Routing is a computationally expensive operation due to repetitive process for each set of input images. Also, the other reason for computational complexity is the number of capsules and too many weights which have to be learned for the classification of datasets.

# Chapter 3

# Hyperparameter tuning

## 3.1 Binary Classification Loss Functions

Deep Learning algorithms use different types of optimization algorithms to learn the objective. First, the objective which is called Loss Function, should be selected based on the distribution of labels. Then, by using the loss function, the error between the predicted output and the true output for the current state of the model must be estimated repeatedly and model weights can be updated in order to reduce the loss after each epoch. In this project, the problem is related to binary classification and examples are assigned one of two labels. So, for this purpose, three different loss functions for binary classification will be investigated.

### 3.1.1 Binary Cross-Entropy

The default loss function for binary classification is cross-entropy and it is used to estimate the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect cross-entropy value is 0 [59]. Binary Cross-Entropy is defined as:

$$\mathrm{BCE}(y, \hat{y}) = -(y\log(\hat{y}) + (1 - y)\log(1 - \hat{y})) \tag{3.1}$$

Where $y$ is the true value and $\hat{y}$ is the predicted value by the prediction model.

### 3.1.2 Weighted Binary Cross-Entropy

The other type of binary cross-entropy is Weighted Binary cross-entropy (WCE) [60]. This method is used when a dataset is skewed [61] and it gives weights by some coefficient to the positive examples. Weighted Cross Entropy can be defined as:

$$\mathrm{WBCE}(\mathrm{y},\hat{\mathrm{y}}) = -(\beta * \mathrm{y}\log(\hat{\mathrm{y}}) + (1-\mathrm{y})\log(1-\hat{\mathrm{y}})) \tag{3.2}$$

$\beta$ is used to tune false negatives and false positives. To reduce the number of false negatives, $\beta$ should be more than one, and to decrease the number of false positives, $\beta$ should be less than one.

### 3.1.3 Balanced Cross-Entropy

Balanced cross entropy (BCE) [62] is another variation of Weighted Cross Entropy. As it can be seen in the formula, both positive and negative examples [63] are weighted.

$$L_{BCE}(\mathrm{y},\hat{\mathrm{y}}) = -(\beta * \mathrm{y}\log(\hat{\mathrm{y}}) + (1-\beta) * (1-\mathrm{y})\log(1-\hat{\mathrm{y}})) \tag{3.3}$$

Where $\beta$ is $1 - \frac{y}{H*W}$

## 3.2 Evaluation Metrics

A classification model is trained by using training data to estimate the class label for test data. However, the results of trained models should be assessed and analyzed by different metrics such as accuracy, sensitivity, specificity, and F1-score. The other assessment technique is Receiver operating characteristics (ROC) and Precision-Recall curves, which demonstrate different interpretations of the classification performance.

### 3.2.1 Accuracy

One of the most commonly used methods for the classification performance is Accuracy (Acc), and it calculates a ratio between the correct classified samples to the total number of samples as follows [65]:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.6}$$

*TP* and *TN* are the number of correct positive and negative classified samples, and *FP* and *FN* are the number of incorrect positive and negative classified samples.

### 3.2.2 Sensitivity and specificity

Sensitivity and specificity are two evaluation metrics to estimate classification performance, especially when the dataset is imbalanced [66]. Recall or sensitivity or True Positive Rate (TPR), demonstrates the correctly classified positive samples to the total number of positive samples, whereas specificity or True Negative Rate (TNR) represents the ratio of the negative correctly classified samples to the total number of negative samples as they are defined in Eq. (3.7, 3.8) [65].

$$TPR = \frac{TP}{TP + FN} \tag{3.7}$$

$$TNR = \frac{TN}{FP + TN} \tag{3.8}$$

### 3.2.3 False positive and false negative rates

False Positive Rate (FPR) represents the proportion of negative samples that are incorrectly classified [67]. The False Negative Rate (FNR) is the proportion of positive samples that are

incorrectly classified and they are defined in Eq. (3.9, 3.10). Both FPR and FNR can be used for imbalanced data classifications due to not being sensitive to changes in data distributions [66].

$$FPR = 1 - TNR = \frac{FP}{FP + TN} = \frac{FP}{P} \tag{3.9}$$

$$FNR = 1 - TpR = \frac{FN}{FN + TP} = \frac{FN}{N} \tag{3.10}$$

### 3.2.4 Predictive values

Positive Prediction Value (PPV) or precision is expressed as the rate of correctly classified positive samples to the total number of positive predicted samples. On the other hand, Negative Predictive Value (NPV) or true negative accuracy (TNA) represents the rate of correctly classified negative samples to the total number of negative predicted samples. The formulas of these metrics are defined in Eq. (3.11, 3.12) [67]. These two metrics are sensitive to imbalanced data [66], [68].

$$PPV = Precision = \frac{TP}{FP + TP} = 1 - FDR \tag{3.11}$$

$$NPV = \frac{TN}{FN + TN} = 1 - FOR \tag{3.12}$$

### 3.2.5 F1-score

F1-score represents the harmonic mean of precision and recall as in Eq. (3.13) [65]. The value of this metric is varied from zero to one, and when F1-score is one, it specifies high classification performance.

$$F - measure = \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN} \tag{3.13}$$

### 3.2.6 Receiver operating characteristics (ROC)

The graphical metric to make a balance between true positives and false positives is the receiver operating characteristics (ROC) curve, which is a two-dimensional graph in which y-axis is related to TPR and the x-axis is based on FPR [69].

### 3.2.7 Area under the ROC curve (AUC)

The Area under the ROC curve (AUC) metric is used to estimate the area under the ROC curve for comparing the performance of different classifiers. The AUC score is between zero and one [70], [71].

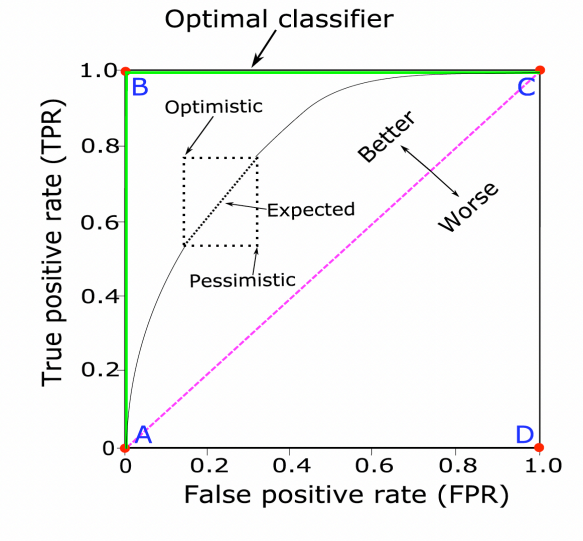



*Figure 4: ROC curve*

## 3.3 Optimization methods

For training a classifier model with the maximum accuracy based on a train dataset and finding the best weights, an optimization algorithm can be used to minimize the error between the predicted output and the true output. There are millions of parameters in each deep learning model and it raises the need to select a proper optimization algorithm based on the application. There are different types of optimization algorithms: Gradient Descent, Stochastic Gradient Descent,

Stochastic Gradient descent with momentum, Mini-Batch Gradient Descent, Adagrad, RMSProp, AdaDelta, Adam.

### 3.3.1 Adam Deep Learning Optimizer

Adaptive Moment Estimation (Adam) is an optimization technique to update network weights during training and it is widely used in deep learning research. This optimization technique is a combination of two other extensions of stochastic gradient descent (SGD), the 'AdaGrad' algorithm and the 'RMSProp' algorithm. This optimization algorithm has a faster running time, requires low memory, and less tuning, and it is efficient when the problem is involved with a lot of data or parameters. In SGD, a single learning rate is used through the training process while in Adam optimize, the learning rate updates for each network weight individually. Adam adapts learning rate based on the first moment (mean) as in RMSProp, and also the second moment of the gradients (the uncentered variance) [72].

#### 3.3.1.1 Momentum

In this algorithm, the 'exponentially weighted average' of the gradients makes the algorithm faster to converge towards the minima.

$$\omega_{t+1} = \omega_t - \alpha m_t \tag{3.14}$$

$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta \omega_t}\right] \tag{3.15}$$

$m_t$ is aggregate of gradients at time *t* [current] (initially, $m_t = 0$), $m_{t-1}$ is aggregate of gradients at time *t*-1 [previous], $\omega_t$ is weights at time t, $\omega_{t+1}$ is weights at time *t*+1, $\alpha_t$ is learning rate at time *t*, $\delta L$ is derivative of loss function, $\delta \omega_t$ is derivative of weights at time *t*, $\beta$ is moving average parameter (constant, 0.9).

### 3.3.1.2 Root Mean Square Propagation (RMSP)

To improve AdaGrad, which takes the cumulative sum of squared gradients, Root Mean Square prop (RMSprop) is used as an adaptive learning algorithm and it takes the 'exponential moving average'.

$$\omega_{t+1} = \omega_t - \frac{\alpha_t}{(\upsilon_t + \varepsilon)^{\frac{1}{2}}} * \left[\frac{\delta L}{\delta \omega_t}\right] x \tag{3.16}$$

$$\upsilon_{t+1} = \beta \upsilon_{t-1} - (1 - \beta) * \left[\frac{\delta L}{\delta \omega_t}\right]^2 \tag{3.17}$$

$\omega_t$ is weights at time, $\omega_{t+1}$ is weights at time $t$+1, $\alpha_t$ is learning rate at time $t$, $\delta L$ is derivative of loss function, $\delta\omega_t$ is derivative of weights at time $t$, $\upsilon_t$ is sum of square of past gradients. [i.e. sum $(\frac{\delta L}{\delta \omega_{t-1}})$] (initially, $\upsilon_t = 0$), $\beta$ is moving average parameter (constant, 0.9), $\epsilon$ is a small positive constant ($10^{-8}$).

Based on the features of the above optimizers, the Adam algorithm is built and the formula is defined as follows:

$$m_t = \beta_1 m_{t-1} - (1 - \beta_1) * \left[\frac{\delta L}{\delta \omega_t}\right] \tag{3.18}$$

$$\upsilon_t = \beta_2 \upsilon_{t-1} - (1 - \beta_2) * \left[\frac{\delta L}{\delta \omega_t}\right]^2 \tag{3.19}$$

$\beta_1$ and $\beta_2$ are decay rates of average of gradients in the above two methods ($\beta_1 = 0.9$, $\beta_2 = 0.999$). As it can be seen in the above optimizers, $m_t$ and $\upsilon_t$ are initially zero, and if $\beta_1$ and $\beta_2$ are equal to one, it causes to gain a tendency to be 'biased towards 0'. In Adam optimizer by computing 'bias-corrected' $m_t$ and $\upsilon_t$, this problem was fixed. In addition, it prevents high oscillations while reaching the global minimum by controlling the weights. The fixed $m_t$ and $\upsilon_t$ and the updated weights are as follows:

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{3.20}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{3.21}$$

$$\omega_{t+1} = \omega_t - \widehat{m_t}\left(\frac{\alpha}{\sqrt{\widehat{v_t}} + \varepsilon}\right) \tag{3.22}$$

## 3.4 Dropout

Deep neural networks are the most powerful machine learning models for researchers. They use a large number of parameters to learn complex functions which leads to overfitting the training data. There are various methods to prevent overfitting in the training process, such as weight loss regularization, reconstruction loss regularization, early stopping, and dropout. However, dropout makes the network more robust compared to other regularization methods due to the fact that different neurons in different layers of network are discarded, and in each iteration, it generates a new network while weights are shared between these networks. The main benefit of dropout is reduction of the co-adaptations between neurons which happens due to fixing mistakes by neurons on the training data and it leads to having a well-fitted network on training data and not having a generalized network on the test data [73].

### 3.4.1 Capsule Dropout

The dropout in the capsule network has to remove a vector rather than just some elements in the vector as it can be seen in Figure 5, because each capsule is a vector [74] and it leads to changes in the properties of the entity which the capsule represents. Therefore, to address this issue, a new dropout algorithm is used which is a new encoding method of mask without any variance in the direction of capsule by Bernoulli distribution to drop some capsules randomly.

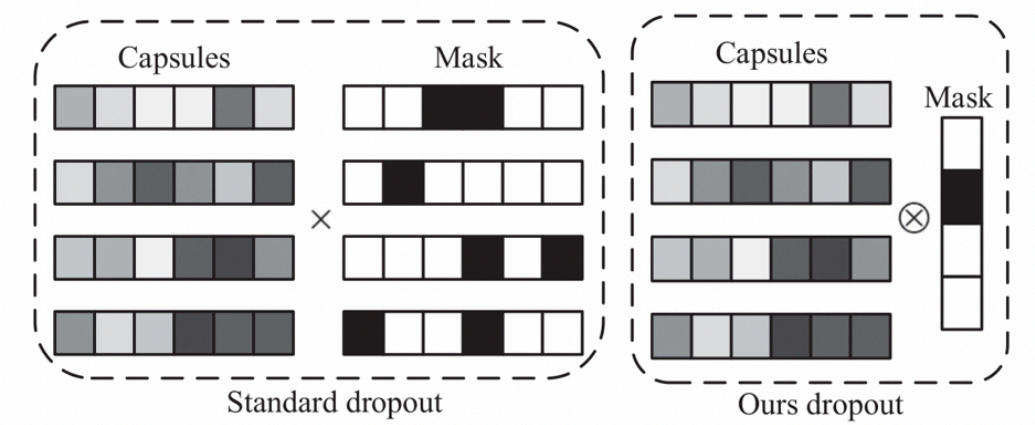



*Figure 5: The proposed capsule dropout has different encoding of the mask. The gray values represent the true values; the black is 0 and the white is 1. '×' means element-wise multiplication, '⊗' means broadcast multiplication*[73]

In this project, the inverted dropout method is used to discard some of the primary capsules and then the dynamic routing algorithm uses the remaining capsules. In this method, the removed capsules are considered a success, and a failure otherwise. In other words, if $p$ is the drop probability, the primary capsules are scaled by $1 \,/\, p - 1$ during the training. All the neurons corresponding to these capsules are divided by $1 - p$. This layer does not make any change to the capsules during inference.

## 3.5 Learning rate scheduler

There are different types of hyperparameters which can affect the performance of a model and one of the most important hyperparameters is learning rate. By this parameter, the amount of changes in the model can be controlled each time the model weights are updated. Choosing the too small learning rate may increase the time of the training process, while a too large learning rate may result in an unstable training process or finding a sub-optimal set of weights too fast. Hence, finding the best learning rates based on their effects on model performance is vital. In this project, two different types of learning rate schedulers are used as follows:

### 3.5.1 Cosine Annealing scheduler

Cosine annealing is a learning rate schedule which starts high and is decreased fast to a minimum value around zero before being increased rapidly. In this process, resetting the high value for learning rate is like a simulated restart with re-use of good weights (warm restart) instead of using a new set of random numbers for weights (cold restart) [75].

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + \cos(\frac{T_{cur}}{T_i}\pi)) \tag{3.23}$$

Where $\eta_{max}^i$ $and$ $\eta_{min}^i$ are ranges for the learning rate, and $T_{cur}$ is the number of epochs before the last restart. $T_i$ is the number of epochs.

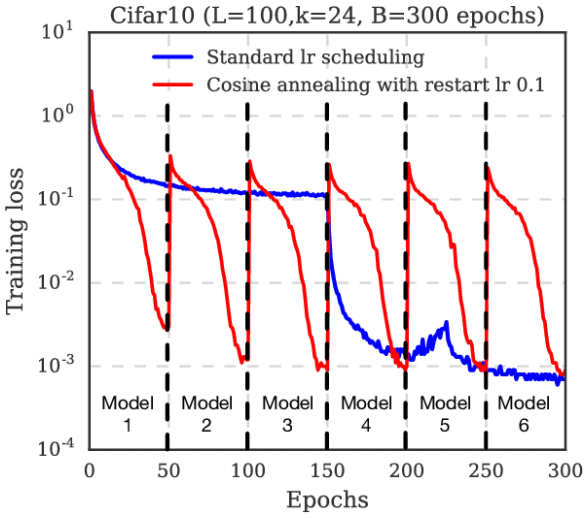



*Figure 6: Training loss based on Cosine Annealing Scheduler and Standard lr scheduling* [75]

### 3.5.2 Learning Rate on Plateau

The *Plateau* Learning Rate scheduler decreases the learning rate by a factor when there is not any improvement in a selected metric for a specified number of epochs (*patience* values). It can be seen that different "*patience*" values, which is the number of epochs before any drop in the value of learning rate and different factors can have effects on the training process.

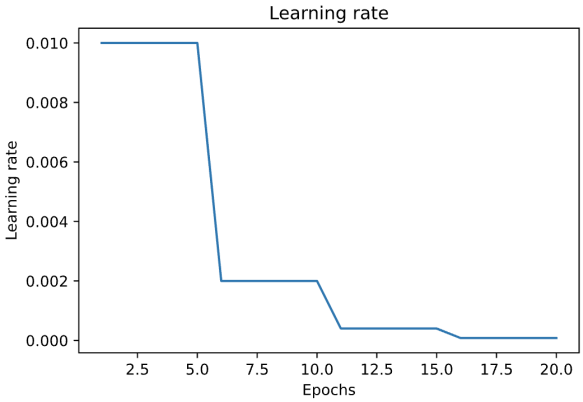



*Figure 7: Plateau Learning rate schedueler*

# Chapter 4

# Experimental Results

## 4.1 Dataset

The COVID-19 radiography database includes chest X-ray images for COVID-19 positive cases along with Normal and Viral Pneumonia images. This dataset includes 3616 COVID-19 positive cases with 10,192 Normal, and 1345 Viral Pneumonia 224 * 224 one channel images. As the main goal of this project is to detect positive COVID-19 cases (binary classification), the labels for COVID-19 positive images are one (positive) and for Viral Pneumonia and Normal images are zero (negative).

*Figure 8: Covid-19, Normal and Viral Pneumonia images*

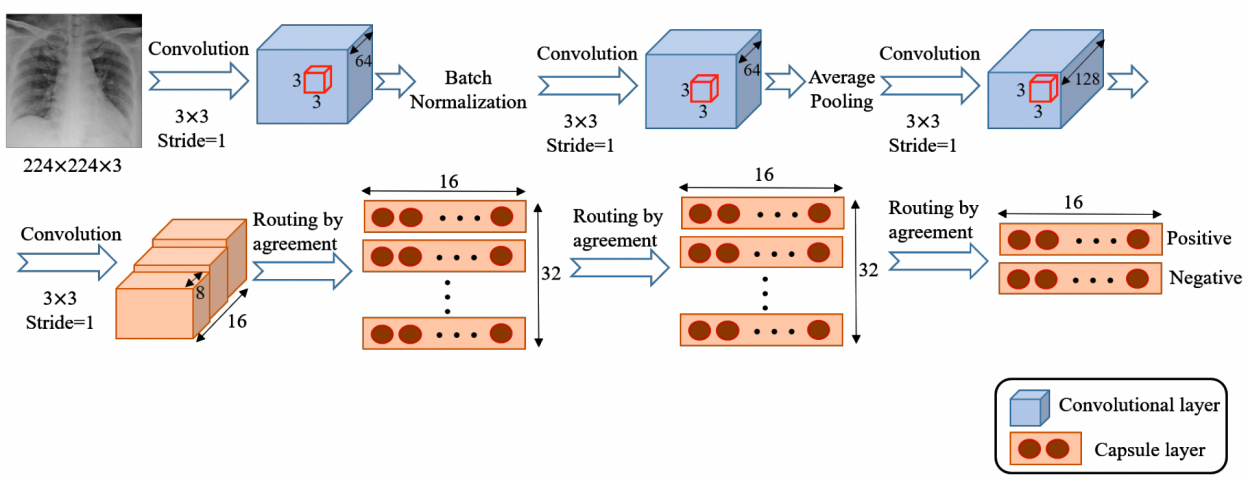



*Figure 9: COVID-CAPS architecture [14]*

## 4.2 Covid-19 Capsule Network

The input images as mentioned in the previous section are X-ray images with 224 * 224 for width and height dimensions and 3 RGB channels without performing any data augmentation due to using a Capsule Network-based architecture. In addition, due to having an imbalanced dataset which includes less numbers of positive cases, $N^+$, than the negative ones, $N^-$, more weight is given to positive samples in the loss function based on the proportion of the positive and negative cases, as follows:

$$loss = \frac{N^+}{N^+ + N^-} \times loss^- + \frac{N^-}{N^+ + N^-} \times loss^+ \tag{4.1}$$

Where $loss^+$ represents the loss of positive samples, and $loss^-$ represents the loss of negative samples.

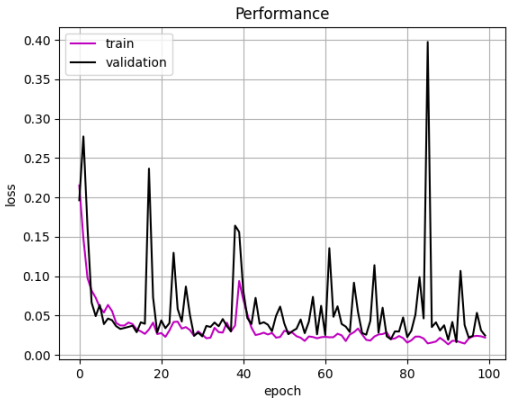
The architecture of the Covid-19 Capsule Network, which is used here, includes 4 convolutional layers and 3 Capsule layers. In the first layer, there is a convolutional block with 64 3*3 kernels moving with stride one. Then, Batch Normalization block is used, followed by the second convolutional layer like the first layer and 2*2 average pooling layer. In addition, two other convolutional layers in the third and fourth layers are added with 128 kernels, and then the output

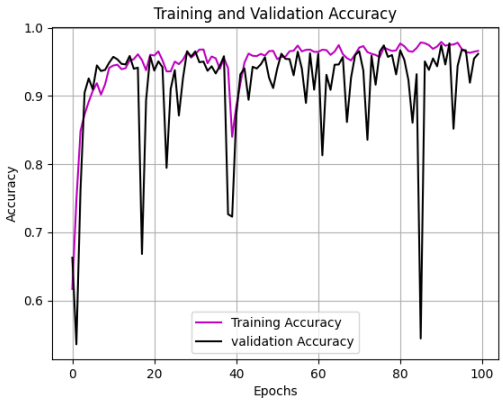
of the fourth layer is reshaped for the first capsule layer. Accordingly, there are three capsule layers by applying the routing by agreement process. Finally, for the last capsule layer, the network contains the instantiation parameters of the positive and negative COVID-19 classes and the probability of each class is estimated by the length of these two capsules.

In this project, Adam optimizer, different types of learning rate, 100 epochs, and a batch size of 16 are used. 80% of the dataset is used for training the model and 20% for validation of the model and selecting the best model based on the performance. Finally, the trained model is evaluated by different types of metrics such as accuracy, loss, precision, recall, and f1-score.

## 4.3 Experimental Results

The baseline architecture with learning rate 0.001 and using Margin Loss reached accuracy 0.962, precision 0.907, recall 0.937, F1-score 0.922 and ROC 0.95. One main purpose of this project is to improve all these metrics, performance, true positive and true negative predictions and decrease overfitting by using dropout method in capsule network. In addition, it can be seen in Figure 10 that there is a fluctuation in train and validation results of the main architecture which can be appeared due to using large learning rates. Therefore, different types of learning rate schedulers are used for solving this problem.



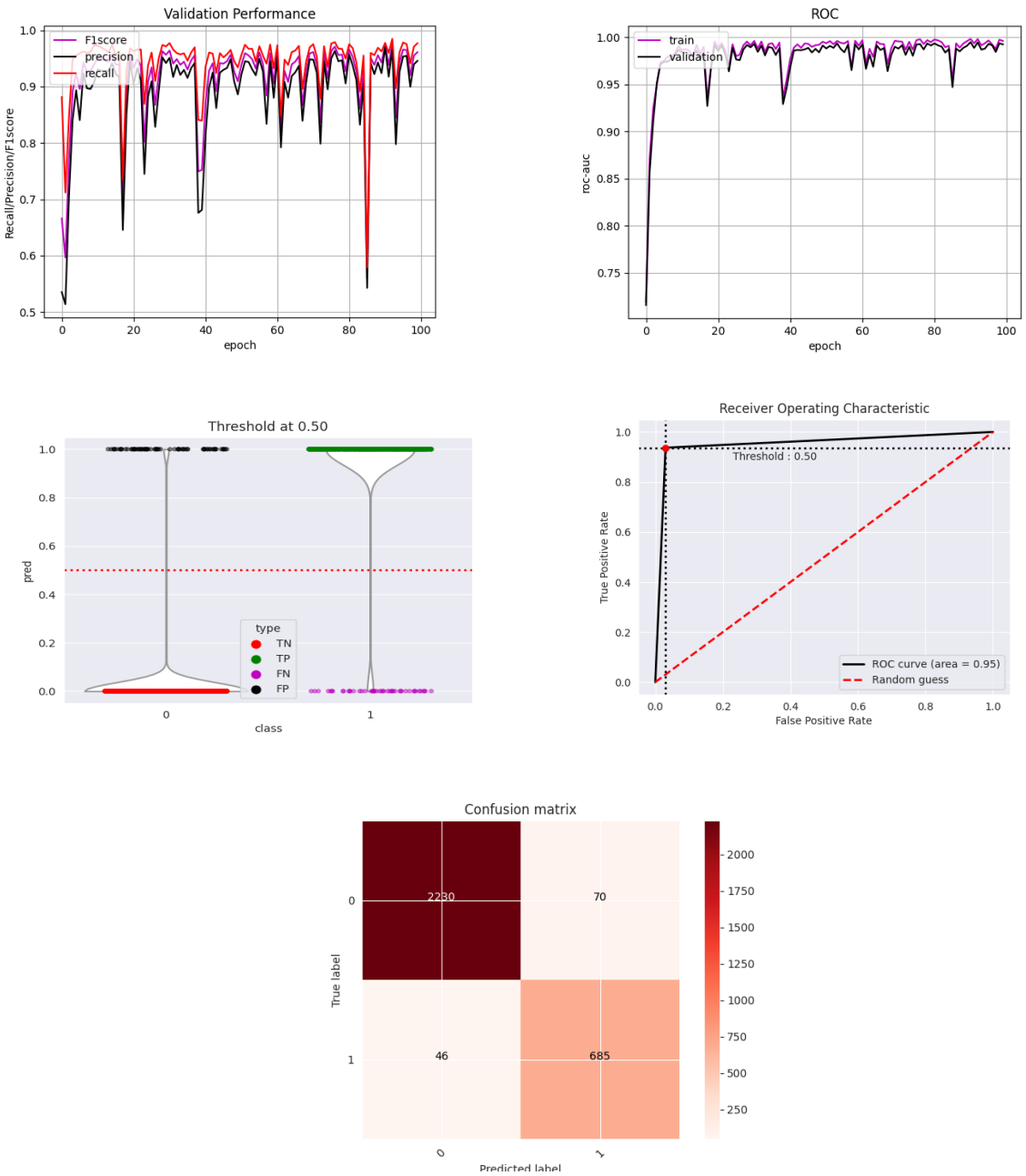
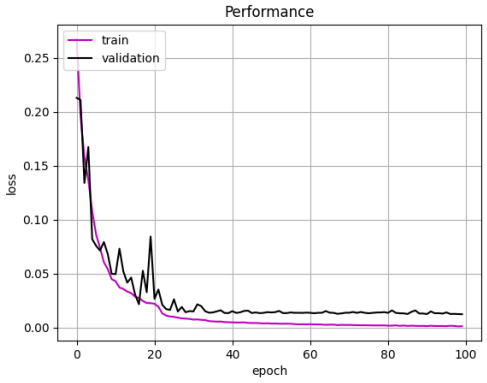
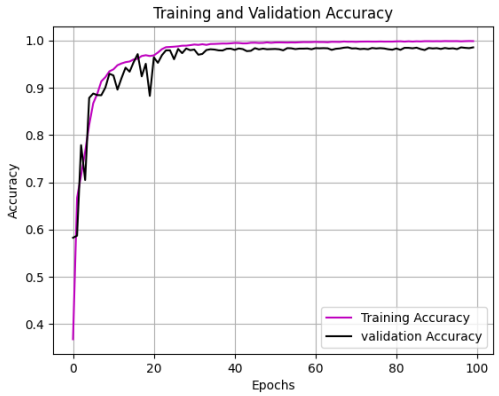


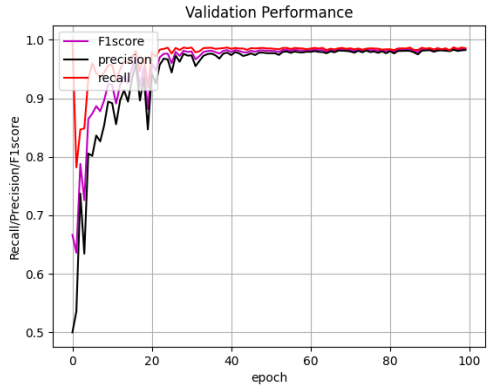

*Figure 10: Loss, accuracy, F1-score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for the main algorithm*

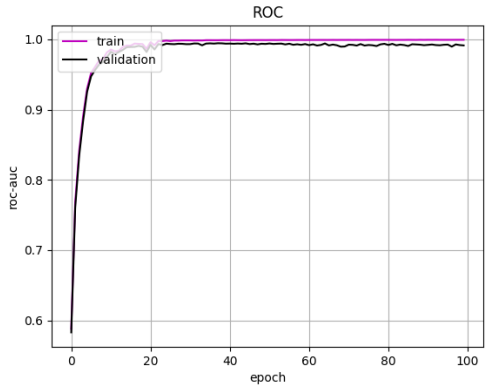
For the next step, *Plateau* Learning Rate scheduler function as a learning scheduler with initial learning rate 0.0001with factor 0.2 and patience 5 was applied. As a result, it is obvious that there is not that much fluctuation during the training and validation process. Moreover, after around 30

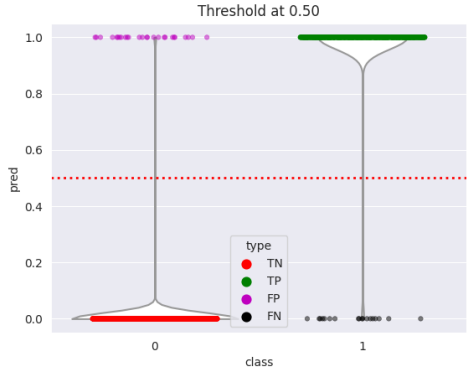
epochs, the network could reach good performance with train accuracy 1, ROC 0.98 and less false positive and false negative as it can be considered in Figure 11.

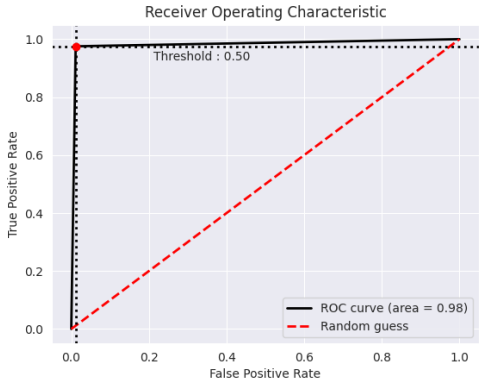


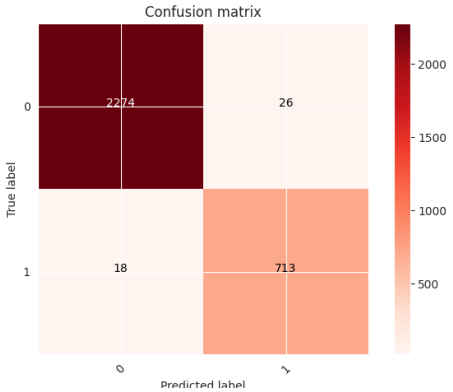
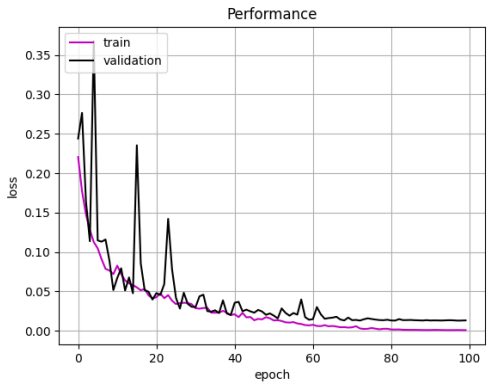



*Figure 11: Loss, accuracy, F1-score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for Plateau Learning Rate scheduler*

The other learning rate scheduler is cosine annealing scheduler with $\eta_{min}$ 0.00001 and $\eta_{max}$ 0.001 which was explained in section 3.5.1 and based on the results in Figure 12, there is some fluctuation until 60 epochs and network needed more epochs to reach the best performance rather than Plateau learning rate scheduler and also number of false negatives is more than the previous used scheduler. Therefore, the improvement by using learning rate scheduler is obvious and among these two schedulers, Plateau one represents better results in this problem.

In the next step, binary loss function was tested and Plateau learning rate scheduler was applied and based on the results in Figure 13, ROC curve area is less around 0.97, and the main differences in loss and accuracy graphs are the more overfitting and less performance rather than Margin Loss while the number of true negatives is better than the other tests.



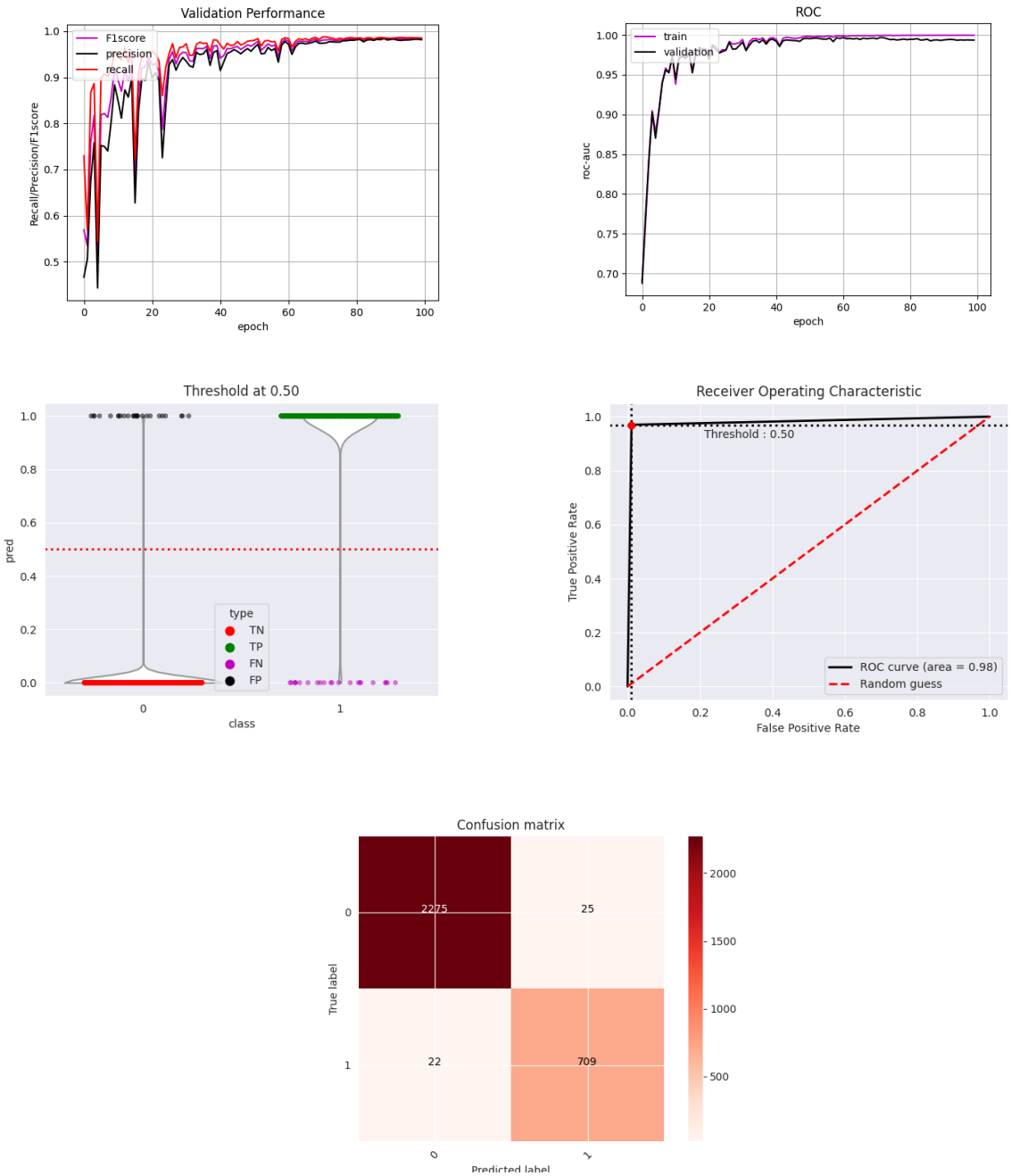
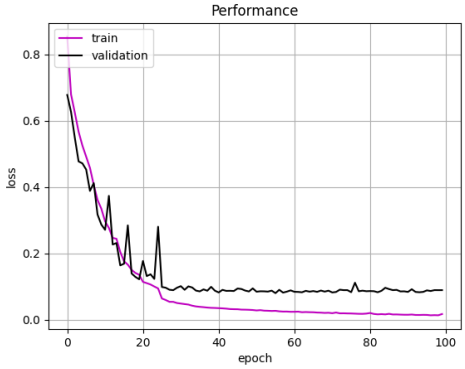
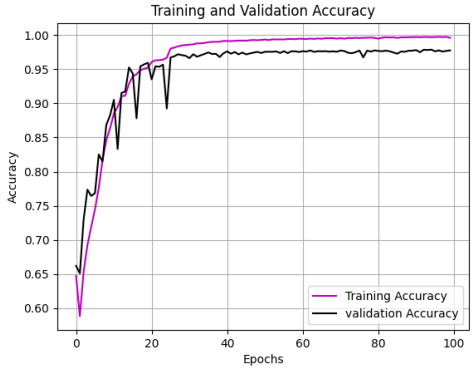
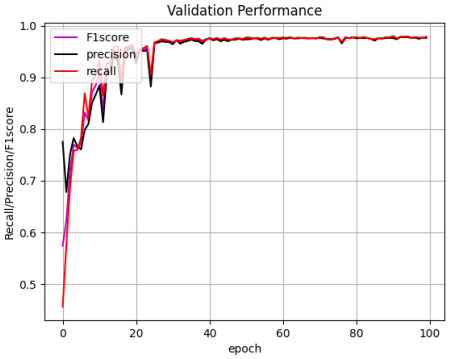
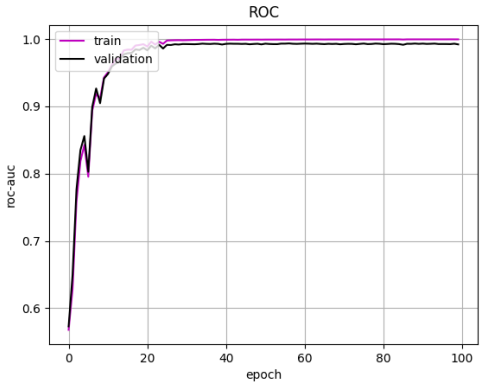



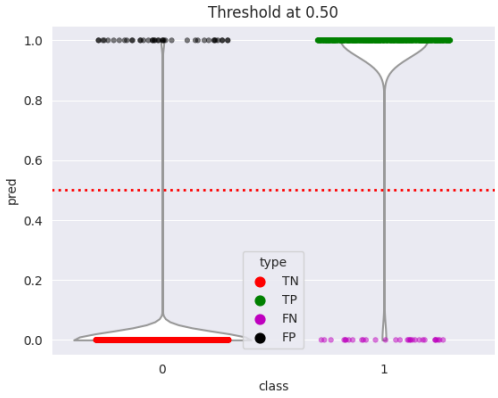

*Figure 12: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for cosine annealing scheduler*

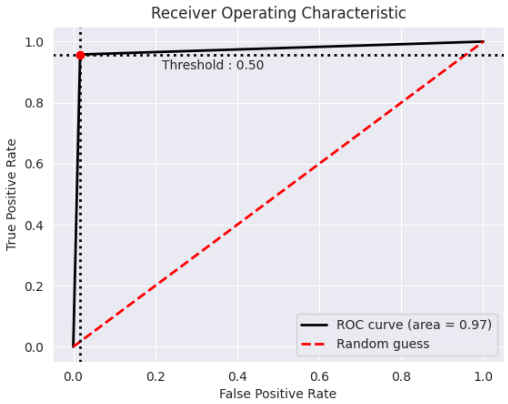

Performance
train
validation
loss
epoch


Training and Validation Accuracy
Accuracy
Epochs
Training Accuracy
validation Accuracy


Validation Performance
F1score
precision
recall
Recall/Precision/F1score
epoch


ROC
train
validation
roc-auc
epoch


Threshold at 0.50
pred
type
TN
TP
FN
FP
class


Receiver Operating Characteristic
Threshold : 0.50
True Positive Rate
False Positive Rate
ROC curve (area = 0.97)
Random guess

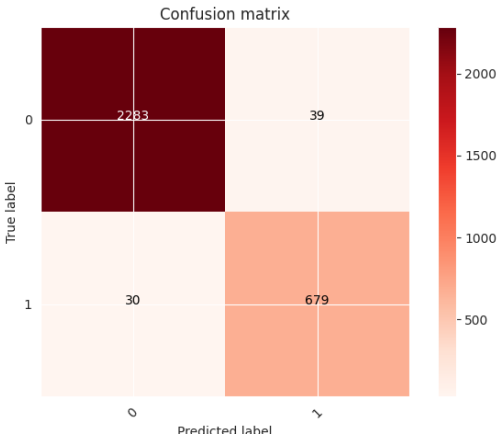
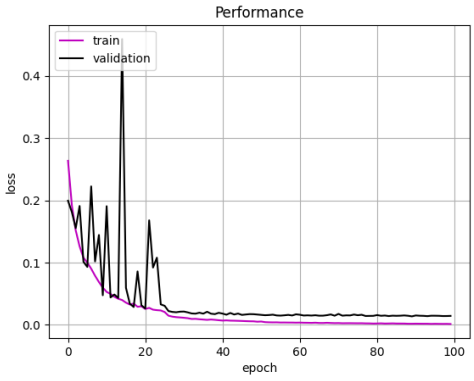



*Figure 13: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for Plateau Learning Rate scheduler and Binary classification*

In this step, different dropout rates were applied to gain any improvement in overfitting issue and it seems that by using lower dropout rate (less than 0.1) and the higher than this value presented worse fluctuation in all curves and overfitting. In addition, the number of true negatives with dropout is greater than without using dropout technique and the ROC curve area is 0.98 when dropout rate 0.1 was used.



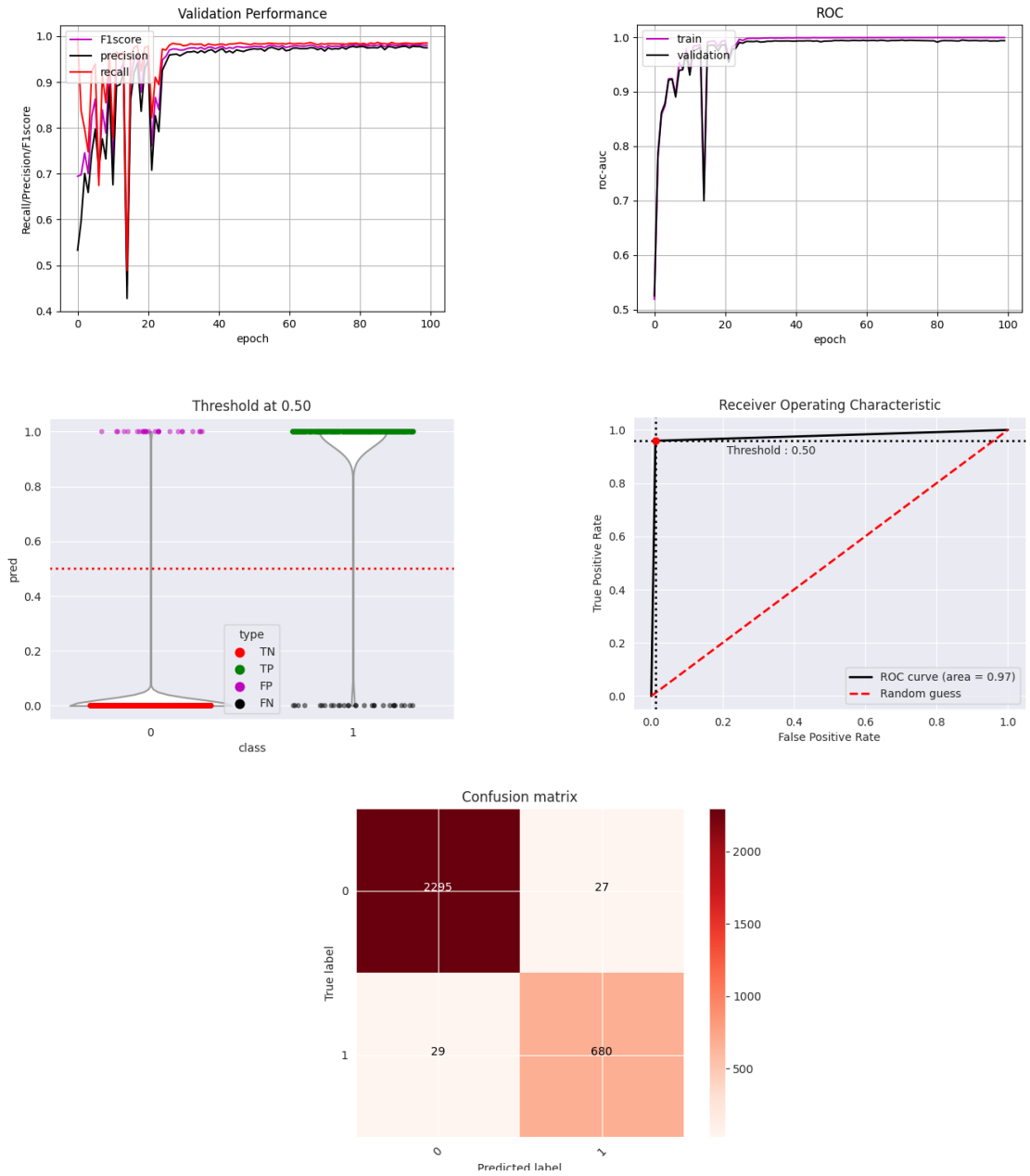
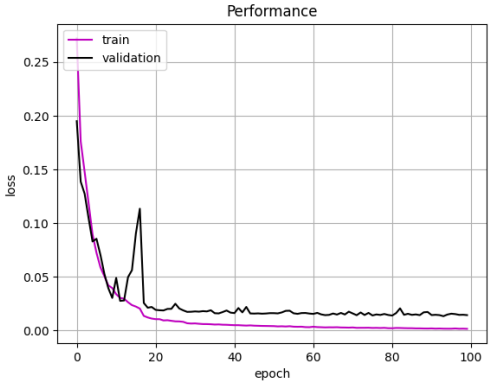
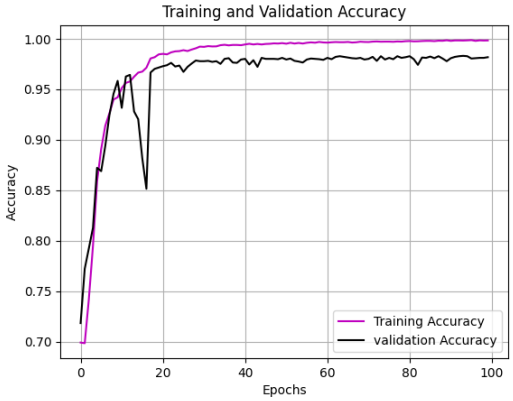
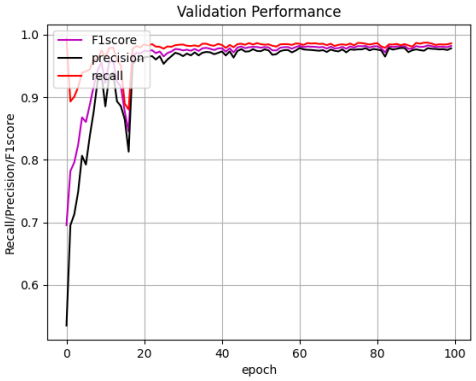
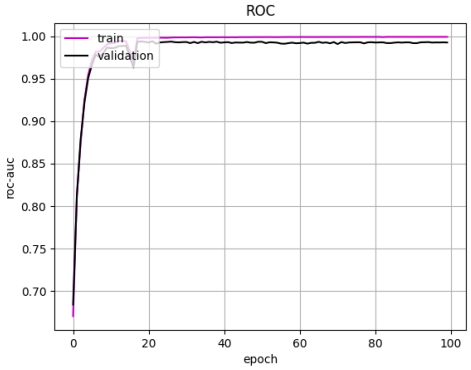



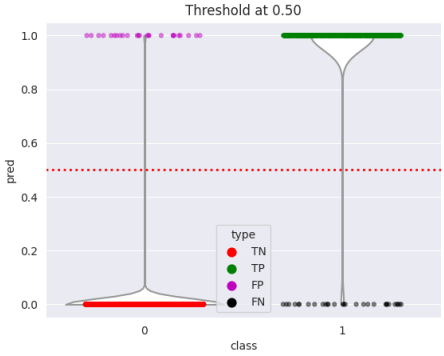

*Figure 14: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for margin loss, Plateau Learning Rate scheduler and dropout rate 0.01*

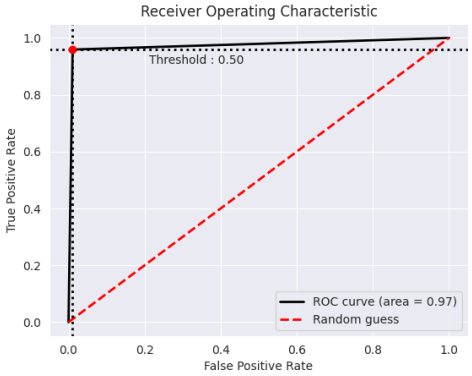

Performance
train
validation
loss
epoch


Training and Validation Accuracy
Accuracy
Epochs
Training Accuracy
validation Accuracy


Validation Performance
F1score
precision
recall
Recall/Precision/F1score
epoch


ROC
train
validation
roc-auc
epoch


Threshold at 0.50
pred
type
TN
TP
FP
FN
class


Receiver Operating Characteristic
Threshold : 0.50
True Positive Rate
False Positive Rate
ROC curve (area = 0.97)
Random guess

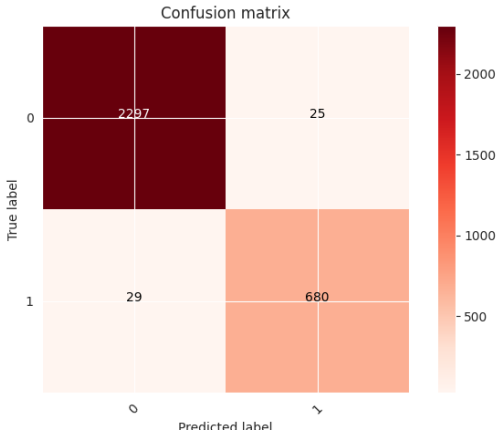
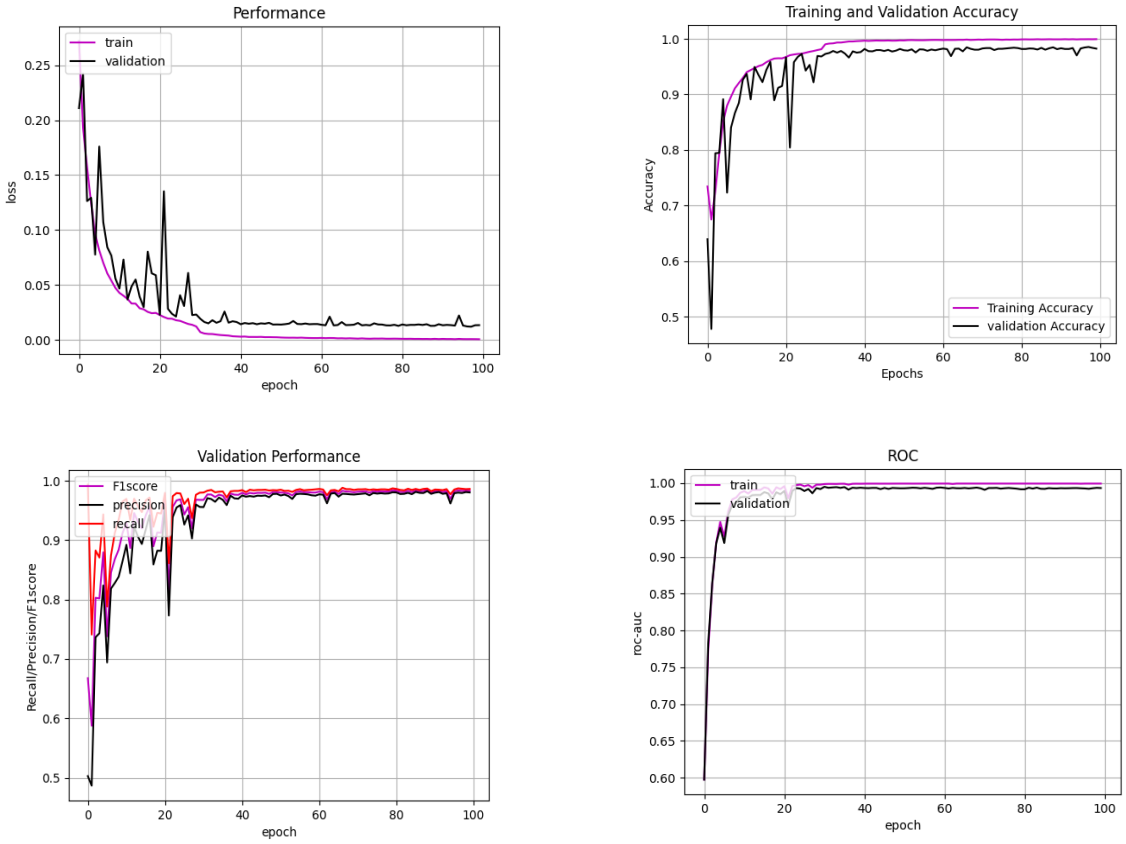



*Figure 15: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for margin loss, Plateau Learning Rate scheduler and dropout rate 0.05*



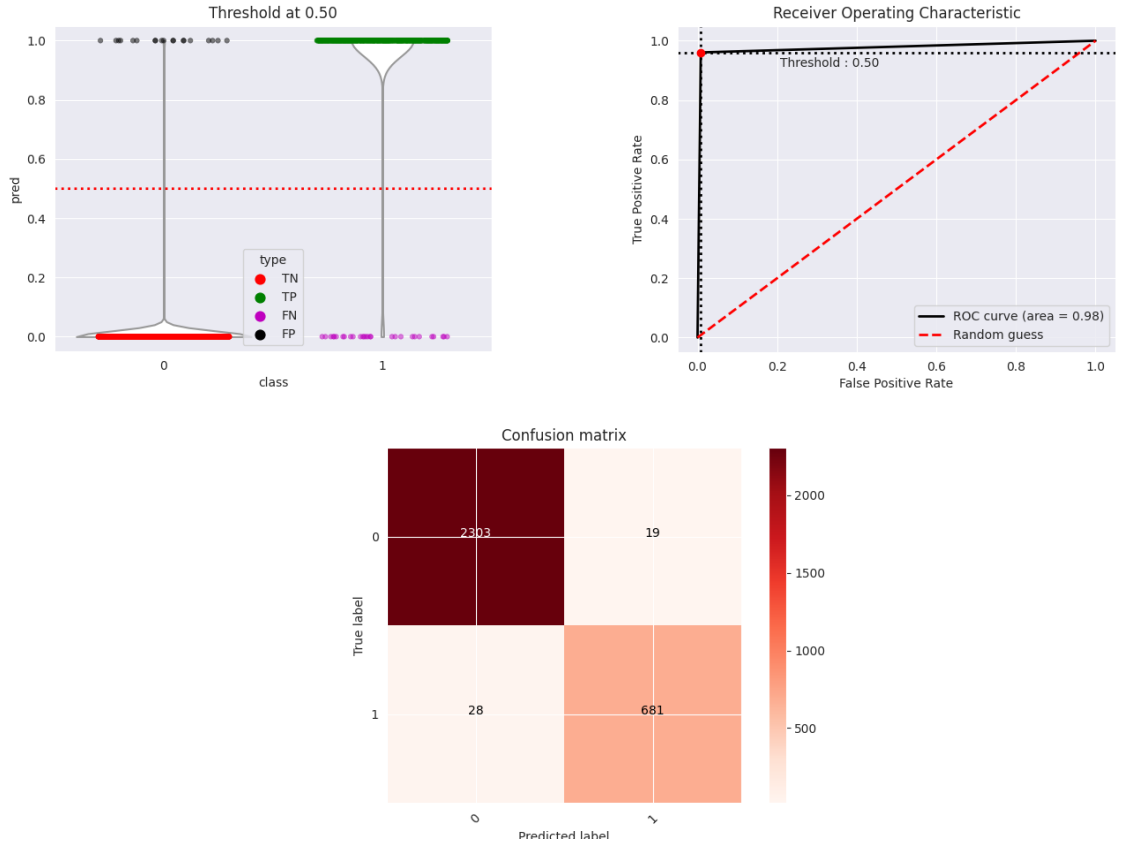
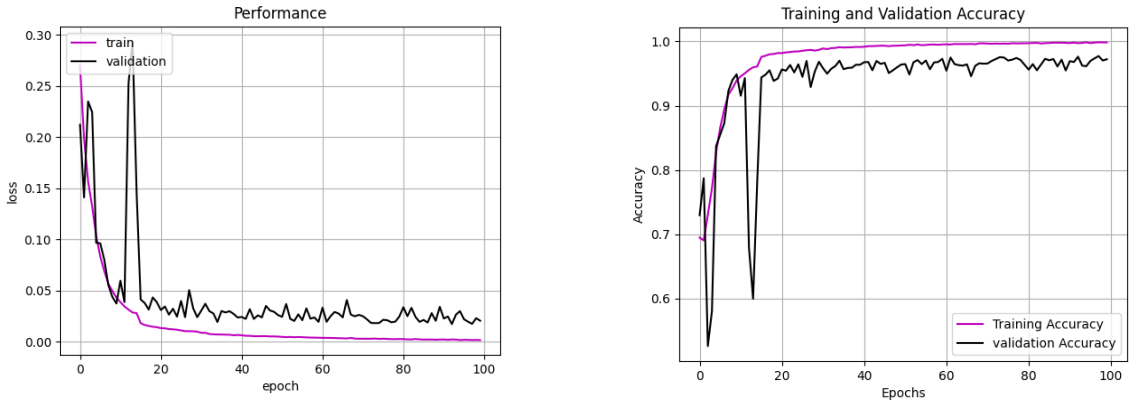



*Figure 16: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for margin loss, Plateau Learning Rate scheduler and dropout rate 0.1*



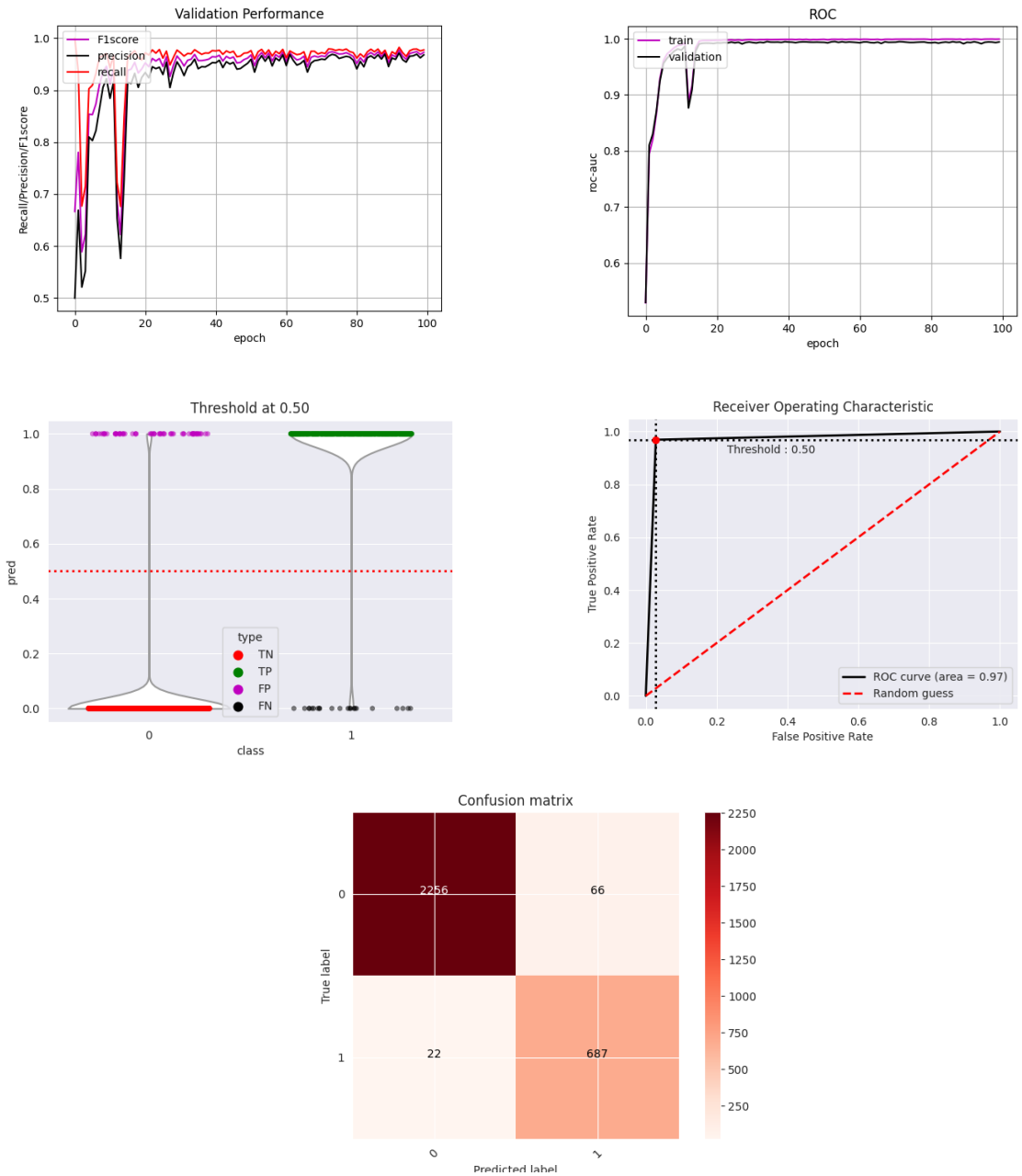
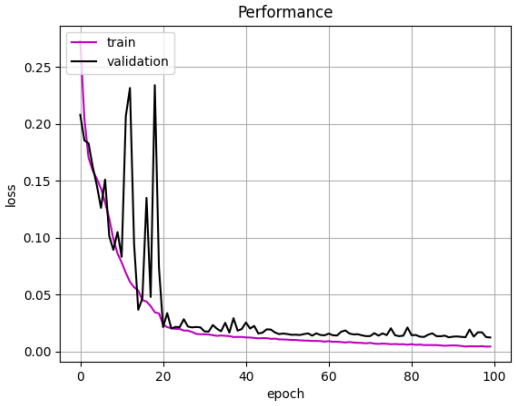


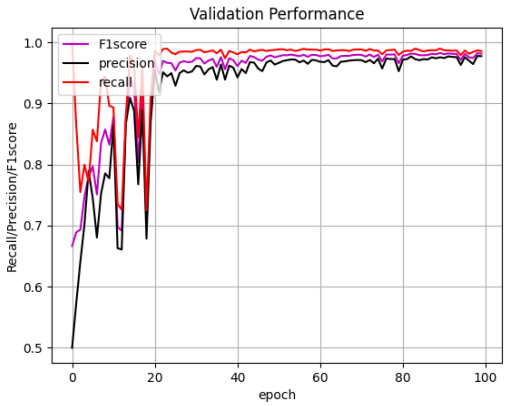

*Figure 17: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for margin loss, Plateau Learning Rate scheduler and dropout rate 0.3*

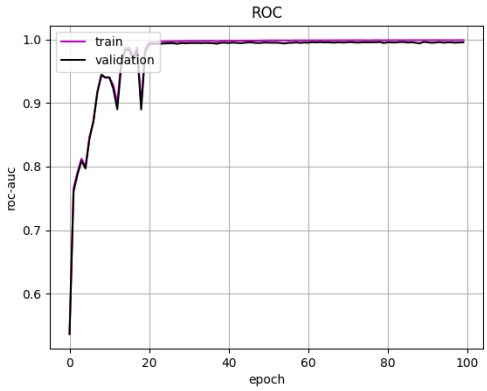
For the last experiment, there is an attempt to decrease the number of learning parameters by removing different Convolutional layers with the least effect on performance. Finally, by removing the last Convolutional layer exactly before capsule layer, the results for accuracy and loss are approximately promising without significant overfitting but it is presented fluctuation specially

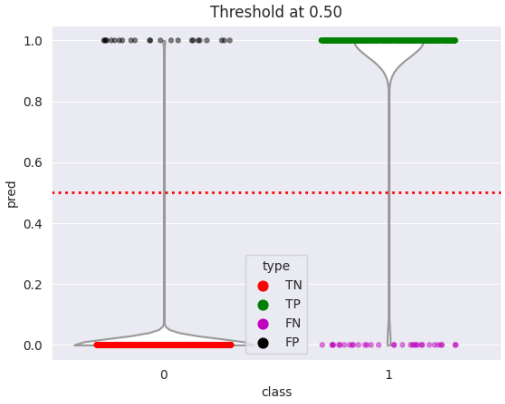
during first 20 epochs. ROC curve area is still 0.97 and even the number of true positive is in the highest situation among all experiments. In this new architecture, the number of learning parameters is 146,752 which makes the network faster while the number of the main architecture is 295,488.

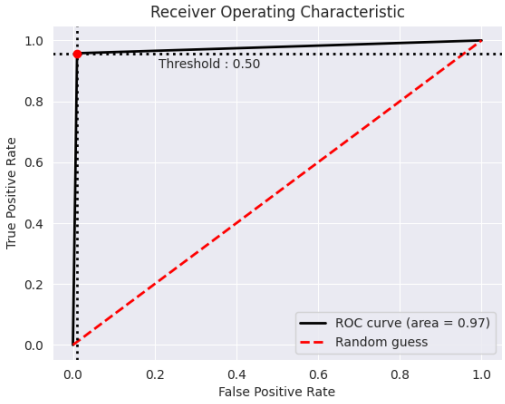






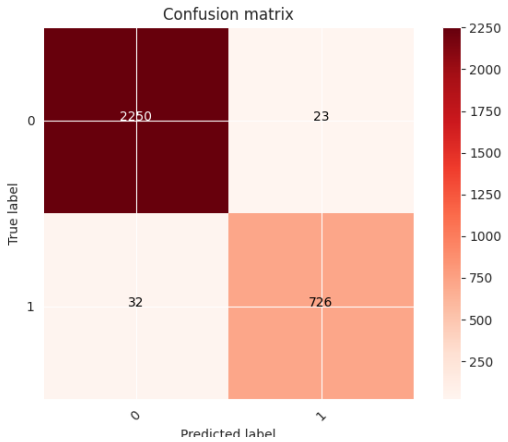



*Figure 18: Loss, accuracy, F1score, Precision, Recall, ROC-AUC, Distribution of predictions, ROC and confusion matrix for margin loss, Plateau Learning Rate scheduler and less capsule layer*

*Table 2: Comparison of results*

| | **loss** | **f1 score** | **Accuracy** | **Precision** | **Recall** | **roc-auc** |
|---|---|---|---|---|---|---|
| **Main** | 0.0247 | 0.9547 | 0.9617 | 0.9393 | 0.9706 | 0.9935 |
| **Plateau** | **0.0092** | **0.9883** | **0.9885** | **0.9859** | **0.9908** | **0.9948** |
| **Cosine** | 0.013 | 0.9835 | 0.9845 | 0.9816 | 0.9859 | 0.9937 |
| **Binary and plateau** | 0.0891 | 0.9776 | 0.9776 | 0.9763 | 0.9789 | 0.9921 |
| **Margin loss plateau dropout 0.05** | 0.0143 | 0.982 | 0.9819 | 0.9781 | 0.9861 | 0.9926 |
| **Margin loss plateau dropout 0.01** | 0.0145 | 0.98 | 0.9812 | 0.9749 | 0.9851 | 0.9939 |
| **Margin loss plateau dropout 0.1** | **0.0134** | **0.9832** | **0.9825** | **0.9803** | **0.9861** | **0.9934** |
| **Margin loss plateau dropout 0.3** | 0.0206 | 0.9727 | 0.9723 | 0.968 | 0.9775 | 0.9943 |
| **Less parameter algorithm** | **0.0125** | **0.9816** | **0.9818** | **0.9774** | **0.9858** | **0.9959** |

As it can be seen in table 2, by comparing different results, the lowest loss of 0.0092, and best accuracy of 0.9885, f1-score of 0.9883, precision of 0.9859, recall of 0.9908 and Area Under the Curve (AUC) of 0.9948 was achieved when Plateau learning rate scheduler and margin loss function was used in capsule network. On the other hand, among different dropout rates which were used to decrease the overfitting, dropout rate 0.1 shows better results. In the last part, by

removing one capsule layer and having far less number of trainable parameters in comparison to the main architecture, it still shows promising results with 146,752 of learning parameters.

# Chapter 5

# Conclusion

Since the end of 2019, people all around the world have been struggling with COVID-19. As a result, the rapid and accurate diagnosis of COVID-19 is crucial for avoiding the distribution of this disease and breaking the chain of transition. Therefore, using Computed Tomography (CT) scans and X-ray images is one of the methods to detect COVID-19 virus. However, using these human-centered diagnosis methods is challenging due to their overlap with other lung infections. To avoid any human-based inaccuracy in the identification of COVID-19 cases, different Deep Neural Network (DNN) algorithms have been developed, mainly based on CNNs. Although, there are some drawbacks in CNNs because of losing spatial information between image instances and require large datasets. Therefore, a combination of Capsule Networks and Convolutional Neural Network as an alternative framework is used in this project, and different parameters and any changes in the architecture are investigated to improve the performance and decrease overfitting and learning parameters of the network.

# Bibliography

[1] A. Rehman, T. Saba, U. Tariq, and N. Ayesha, “Deep Learning-Based COVID-19 Detection Using CT and X-Ray Images: Current Analytics and Comparisons,” IT Professional, vol. 23, no. 3, pp. 63–68, May 2021, doi: 10.1109/MITP.2020.3036820.
[2] G. Ciaparrone, F. Luque Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, “Deep learning in video multi-object tracking: A survey,” Neurocomputing, vol. 381, pp. 61–88, Mar. 2020, doi: 10.1016/j.neucom.2019.11.023.
[3] M. Asadi-Aghbolaghi et al., “A Survey on Deep Learning Based Approaches for Action and Gesture Recognition in Image Sequences,” in 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), May 2017, pp. 476–483. doi: 10.1109/FG.2017.150.
[4] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, “Deep Face Recognition: A Survey,” in 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Oct. 2018, pp. 471–478. doi: 10.1109/SIBGRAPI.2018.00067.
[5] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, “Image Steganography: A Review of the Recent Advances,” IEEE Access, vol. 9, pp. 23409–23423, 2021, doi: 10.1109/ACCESS.2021.3053998.
[6] O. Elharrouss, N. Almaadeed, and S. Al-Maadeed, “An image steganography approach based on k-least significant bits (k-LSB),” in 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Feb. 2020, pp. 131–135. doi: 10.1109/ICIoT48696.2020.9089566.
[7] N. Subramanian, I. Cheheb, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, “End-to-End Image Steganography Using Deep Convolutional Autoencoders,” IEEE Access, vol. 9, pp. 135585–135593, 2021, doi: 10.1109/ACCESS.2021.3113953.
[8] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,” Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.09829
[9] A. Shahroudnejad, A. Mohammadi, and K. N. Plataniotis, “Improved Explainability of Capsule Networks: Relevance Path by Agreement,” Feb. 2018, [Online]. Available: http://arxiv.org/abs/1802.10204
[10] J. Su, D. V. Vargas, and K. Sakurai, “One Pixel Attack for Fooling Deep Neural Networks,” IEEE Transactions on Evolutionary Computation, vol. 23, no. 5, pp. 828–841, Oct. 2019, doi: 10.1109/TEVC.2019.2890858.
[11] J. Su, D. V. Vargas, and K. Sakurai, “Attacking convolutional neural network using differential evolution,” IPSJ Transactions on Computer Vision and Applications, vol. 11, no. 1, p. 1, Dec. 2019, doi: 10.1186/s41074-019-0053-3.
[12] L. Wang, Z. Q. Lin, and A. Wong, “COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images,” Scientific Reports, vol. 10, no. 1, Dec. 2020, doi: 10.1038/s41598-020-76550-z.
[13] J. Zhang et al., “Viral Pneumonia Screening on Chest X-ray Images Using Confidence-Aware Anomaly Detection,” Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.12338
[14] P. Afshar, S. Heidarian, F. Naderkhani, A. Oikonomou, K. N. Plataniotis, and A. Mohammadi, “COVID-CAPS: A capsule network-based framework for identification of COVID-19 cases from X-ray images,” Pattern Recognition Letters, vol. 138, pp. 638–643, Oct. 2020, doi: 10.1016/j.patrec.2020.09.010.

55
[15] A. Abbas, M. M. Abdelsamea, and M. M. Gaber, “Classification of COVID-19 in chest Xray images using DeTraC deep convolutional neural network,” Applied Intelligence, vol. 51, no. 2, pp. 854–864, Feb. 2021, doi: 10.1007/s10489-020-01829-7.
[16] M. Siddhartha and A. Santra, “COVIDLite: A depth-wise separable deep neural network with white balance and CLAHE for detection of COVID-19,” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.13873
[17] A. H. Panahi, A. Rafiei, and A. Rezaee, “FCOD: Fast COVID-19 Detector based on deep learning techniques,” Informatics in Medicine Unlocked, vol. 22, p. 100506, 2021, doi: 10.1016/j.imu.2020.100506.
[18] H. Maghdid, A. T. Asaad, K. Z. G. Ghafoor, A. S. Sadiq, S. Mirjalili, and M. K. K. Khan, “Diagnosing COVID-19 pneumonia from x-ray and CT images using deep learning and transfer learning algorithms,” in Multimodal Image Exploitation and Learning 2021, Apr. 2021, p. 26. doi: 10.1117/12.2588672.
[19] K. Hammoudi et al., “Deep Learning on Chest X-ray Images to Detect and Evaluate Pneumonia Cases at the Era of COVID-19.” [Online]. Available: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
[20] M. Rahimzadeh and A. Attar, “A modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2,” Informatics in Medicine Unlocked, vol. 19, p. 100360, 2020, doi: 10.1016/j.imu.2020.100360.
[21] I. D. Apostolopoulos, S. I. Aznaouridis, and M. A. Tzani, “Extracting Possibly Representative COVID-19 Biomarkers from X-ray Images with Deep Learning Approach and Image Data Related to Pulmonary Diseases,” Journal of Medical and Biological Engineering, vol. 40, no. 3, pp. 462–469, Jun. 2020, doi: 10.1007/s40846-020-00529-4.
[22] A. I. Khan, J. L. Shah, and M. M. Bhat, “CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images,” Computer Methods and Programs in Biomedicine, vol. 196, p. 105581, Nov. 2020, doi: 10.1016/j.cmpb.2020.105581.
[23] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, “Automated detection of COVID-19 cases using deep neural networks with X-ray images,” Computers in Biology and Medicine, vol. 121, p. 103792, Jun. 2020, doi: 10.1016/j.compbiomed.2020.103792.
[24] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger.” [Online]. Available: http://pjreddie.com/yolo9000/
[25] F. Ozyurt, T. Tuncer, and A. Subasi, “An automated COVID-19 detection based on fused dynamic exemplar pyramid feature extraction and hybrid feature selection using deep learning,” Computers in Biology and Medicine, vol. 132, p. 104356, May 2021, doi: 10.1016/j.compbiomed.2021.104356.
[26] T. Mahmud, M. A. Rahman, and S. A. Fattah, “CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization,” Computers in Biology and Medicine, vol. 122, p. 103869, Jul. 2020, doi: 10.1016/j.compbiomed.2020.103869.
[27] S. Karakanis and G. Leontidis, “Lightweight deep learning models for detecting COVID-19 from chest X-ray images,” Computers in Biology and Medicine, vol. 130, p. 104181, Mar. 2021, doi: 10.1016/j.compbiomed.2020.104181.
56

[28] P. Kumar Sethy, S. Kumari Behera, P. Kumar Ratha, and P. Biswas, “Detection of coronavirus Disease (COVID-19) based on Deep Features and Support Vector Machine,” 2020. [Online]. Available: www.preprints.org
[29] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
[30] U. Ozkaya, S. Ozturk, and M. Barstugan, “Coronavirus (COVID-19) Classification using Deep Features Fusion and Ranking Technique.”
[31] D. N. Le, V. S. Parvathy, D. Gupta, A. Khanna, J. J. P. C. Rodrigues, and K. Shankar, “IoT enabled depthwise separable convolution neural network with deep support vector machine for COVID-19 diagnosis and classification,” International Journal of Machine Learning and Cybernetics, vol. 12, no. 11, pp. 3235–3248, Nov. 2021, doi: 10.1007/s13042-020-01248-7.
[32] A. K. Dey, M. Sharma, and M. R. Meshram, “Image Processing Based Leaf Rot Disease, Detection of Betel Vine (Piper BetleL.),” Procedia Computer Science, vol. 85, pp. 748–754, 2016, doi: 10.1016/j.procs.2016.05.262.
[33] K. Golhani, S. K. Balasundram, G. Vadamalai, and B. Pradhan, “A review of neural networks in plant disease detection using hyperspectral data,” Information Processing in Agriculture, vol. 5, no. 3, pp. 354–371, Sep. 2018, doi: 10.1016/j.inpa.2018.05.002.
[34] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification,” Computational Intelligence and Neuroscience, vol. 2016, pp. 1–11, 2016, doi: 10.1155/2016/3289801.
[35] W. Sun, H. Zhao, and Z. Jin, “A facial expression recognition method based on ensemble of 3D convolutional neural networks,” Neural Computing and Applications, vol. 31, no. 7, pp. 2795–2812, Jul. 2019, doi: 10.1007/s00521-017-3230-2.
[36] B. Fasel and J. Luettin, “Automatic facial expression analysis: a survey,” Pattern Recognition, vol. 36, no. 1, pp. 259–275, Jan. 2003, doi: 10.1016/S0031-3203(02)00052-3.
[37] S. Sukittanon, A. C. Surendran, J. C. Platt, and C. J. C. Burges, “CONVOLUTIONAL NETWORKS FOR SPEECH DETECTION.”
[38] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional Neural Networks for Speech Recognition,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pp. 1533–1545, Oct. 2014, doi: 10.1109/TASLP.2014.2339736.
[39] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[40] Q. Chen, J. Xu, and V. Koltun, “Fast Image Processing with Fully-Convolutional Networks,” Sep. 2017, [Online]. Available: http://arxiv.org/abs/1709.00643
[41] M. Nielsen, “Neural Networks and Deep Learning.” [Online]. Available: http://neuralnetworksanddeeplearning.com
[42] Y. LeCun et al., “Backpropagation Applied to Handwritten Zip Code Recognition,” Neural Computation, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.
[43] J. Wu, “Introduction to Convolutional Neural Networks,” 2017.
[44] C.-C. J. Kuo, “Understanding Convolutional Neural Networks with A Mathematical Model,” Sep. 2016, [Online]. Available: http://arxiv.org/abs/1609.04112

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” Feb. 2015, [Online]. Available: http://arxiv.org/abs/1502.01852
[46] O. Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.0575
[47] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification,” Computational Intelligence and Neuroscience, vol. 2016, pp. 1–11, 2016, doi: 10.1155/2016/3289801.
[48] K. Golhani, S. K. Balasundram, G. Vadamalai, and B. Pradhan, “A review of neural networks in plant disease detection using hyperspectral data,” Information Processing in Agriculture, vol. 5, no. 3, pp. 354–371, Sep. 2018, doi: 10.1016/j.inpa.2018.05.002.
[49] B. Fasel and J. Luettin, “Automatic facial expression analysis: a survey,” Pattern Recognition, vol. 36, no. 1, pp. 259–275, Jan. 2003, doi: 10.1016/S0031-3203(02)00052-3.
[50] S. Sukittanon, A. C. Surendran, J. C. Platt, and C. J. C. Burges, “CONVOLUTIONAL NETWORKS FOR SPEECH DETECTION.” [Online]. Available: http://www.iscaspeech.org/archive
[51] A. Patil and M. Rane, “Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition,” 2021, pp. 21–30. doi: 10.1007/978-981-15-7078-0_3.
[52] G. Goos et al., “LNCS 6354 - Artificial Neural Networks … ICANN 2010,” 2010.
[53] M. Lin, Q. Chen, and S. Yan, “Network In Network,” Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.4400
[54] S. Pattanayak, Pro Deep Learning with TensorFlow. Apress, 2017. doi: 10.1007/978-1-4842-3096-1.
[55] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” 2011.
[56] B. Gao and L. Pavel, “On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.00805
[57] M. Kwabena Patrick, A. Felix Adekoya, A. Abra Mighty, and B. Y. Edward, “Capsule Networks – A survey,” Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 1, pp. 1295–1310, Jan. 2022, doi: 10.1016/j.jksuci.2019.09.014.
[58] A. F. M. Saif, T. Imtiaz, S. Rifat, C. Shahnaz, W.-P. Zhu, and M. O. Ahmad, “CapsCovNet: A Modified Capsule Network to Diagnose COVID-19 From Multimodal Medical Imaging,” IEEE Transactions on Artificial Intelligence, vol. 2, no. 6, pp. 608–617, Aug. 2021, doi: 10.1109/tai.2021.3104791.
[59] Yi-de Ma, Qing Liu, and Zhi-bai Quan, “Automated image segmentation using improved PCNN model based on cross-entropy,” in Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004., pp. 743–746. doi: 10.1109/ISIMP.2004.1434171.
[60] V. Pihur, S. Datta, and S. Datta, “Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach,” Bioinformatics, vol. 23, no. 13, pp. 1607–1615, Jul. 2007, doi: 10.1093/bioinformatics/btm158.
[61] Y. Ho and S. Wookey, “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling,” IEEE Access, vol. 8, pp. 4806–4813, 2020, doi: 10.1109/ACCESS.2019.2962617.

[62] S. Xie and Z. Tu, “Holistically-Nested Edge Detection,” in 2015 IEEE International Conference on Computer Vision (ICCV), Dec. 2015, pp. 1395–1403. doi: 10.1109/ICCV.2015.164.
[63] S. Pan et al., “Diagnostic Model of Coronary Microvascular Disease Combined With Full Convolution Deep Network With Balanced Cross-Entropy Cost Function,” IEEE Access, vol. 7, pp. 177997–178006, 2019, doi: 10.1109/ACCESS.2019.2958825.
[64] R. C. Moore and J. Denero, “L 1 AND L 2 REGULARIZATION FOR MULTICLASS HINGE LOSS MODELS.”
[65] A. Sattar and B. Kang, “Lecture Notes in Artificial Intelligence 4304 Subseries of Lecture Notes in Computer Science,” 2006.
[66] V. García, R. A. Mollineda, and J. S. Sánchez, “Theoretical analysis of a performance measure for imbalanced data,” in Proceedings - International Conference on Pattern Recognition, 2010, pp. 617–620. doi: 10.1109/ICPR.2010.156.
[67] D. M. W. Powers and Ailab, “EVALUATION: FROM PRECISION, RECALL AND FMEASURE
TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION.”
[68] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” Information Processing & Management, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.
[69] K. Hajian-Tilaki, “Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation,” 2013.
[70] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” Pattern Recognition, vol. 30, no. 7, pp. 1145–1159, Jul. 1997, doi: 10.1016/S0031-3203(96)00142-2.
[71] C. E. Metz, “Basic principles of ROC analysis,” Seminars in Nuclear Medicine, vol. 8, no. 4, pp. 283–298, Oct. 1978, doi: 10.1016/S0001-2998(78)80014-2.
[72] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980
[73] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu, “MS-CapsNet: A Novel Multi-Scale Capsule Network,” IEEE Signal Processing Letters, vol. 25, no. 12, pp. 1850–1854, Dec. 2018, doi: 10.1109/LSP.2018.2873892.
[74] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” 2014.
[75] I. Loshchilov and F. Hutter, “SGDR: Stochastic Gradient Descent with Warm Restarts,” Aug. 2016, [Online]. Available: http://arxiv.org/abs/1608.03983