EE 416 – Image Processing and Computer Vision **(UH Manoa, Fall 2020)**

Final Report

December 2, 2020

Instructor: Il Yong Chun *Name: Dino Mariano, Frendy Lio Can, Keenan Lee*

Introduction

The goal of this project was to create and train a fully-connected network (FCN) model (utilizing a pre-trained ResNet-50 base) to be able to classify CT-scans of lungs into three categories, "Healthy", "Other", and "Covid".

The initial dataset consisted of 4171 images, containing an uneven distribution of 757 "Healthy" images, 1247 "Other" images, and 2167 "Covid"images. Dimensions between images differed significantly from each other.

Question a)

The question-box is filled by:

$$x_i^{[2]} = \sum_{j=1}^{3} z_j^{[1]} * c_i^{[2]} \quad , x_i : i = 1, \cdots, 5$$

The number of parameters is: $3 * 5 * 5$ (First Layer) + $5 * 5 * 5$ (Second Layer) = 200.

Question c)

Pre-processing

Before training, several operations were performed on the images. All images were resized to 224x224 and converted to grayscale. The reason for this resizing was to meet ResNet-50 requirements of a 3x224x224 image and was performed via scaling using the resize function of the OpenCV library.

We chose to scale the images, as we found that cropping would result in the loss of too much information. The grayscale transformation was for transportability. The training was done on the Google Colab platform and new sessions required a re-import and processing of data, so saving images as grayscale made everything quicker.

Input images were also augmented, with all images being rotated three times (90, 180, and 270) as well as being flipped horizontally to increase the size of the dataset.

To improve our data, we could have performed additional augmentation to the "Healthy" and

“Other“ images, which would have evened out the distribution of data; however, we were not able to do so due to time constraints.

When training the model, we needed to include five-fold cross-validation. To load the data into our model, we first split the images into 5 different folders. The split was done randomly, and each class of image was split separately to ensure a uniform distribution of classes across splits. Labels were stored in a CSV, and each folder was loaded as a separate dataset.

To implement our five-fold cross-validation, we created five separate train/test sets, concatenating four of the datasets for training and using the last dataset as the testing split.

When images were loaded into our dataset class, two duplicate channels were added to each image, bringing the image to the required 3x224x224 from the original 1x224x224.

Additionally, images were normalized and standardized. Mean and the standard deviation were calculated individually for each of the 5 datasets. This was done for convenience, as we found that the recommended way, standardizing both training and testing data on values of mean and standard deviation calculated from the training set [2], was difficult with our implementation of five-fold cross-validation.

We found this method acceptable because it did not result in data leakage and we found that the mean and standard deviation across different datasets varied by only around ± 0.003.

ResNet-50

We found that the best results where by freezing the first 7 children of the pre-trained ResNet-50 model. Increasing the number of frozen children led to a lower accuracy while freezing less led to the model overfitting much quicker with lower testing accuracy.

Forward Network

Our final Feed Forward Network consisted of three linear layers, separated by batch normalization and ReLU layers. We learnt that applying batch normalization after the activation function is a general rule [1].

In our case, we found no significant difference between placing batch normalization layers before or after ReLU. We did, however, achieve slightly better accuracy placing batch normalization before the ReLU function; thus, our final model consisted of the following:

1. Linear layer
2. Batch normalization
3. ReLU
4. Linear layer

5. Batch normalization
6. ReLU
7. Linear layer

We decided on a model with three linear layers as we found that (without tuning) only having two layers led to major underfitting, and having four linear layers led to quick overfitting and low overall accuracy.

Optimizer

In our final model, we decided to use Stochastic Gradient Descent (SDG) as our optimizer (with momentum = 0.2). However, we were able to achieve more than 85% accuracy with both ADAM and SDG, so our reason for choosing SDG was not based on anything significant.

Loss Function

We kept the default Cross-Entropy Loss function.

Hyperparameters

After trial and error, we concluded the following:

- Batch Size: 32
- Number of neurons in layer 1 (l1Tol2Features): 96
- Number of neurons in layer 2 (l2tol3Features): 64
- Epochs: 10
- Learning rate: 0.08
- The number of children frozen in ResNet-50: 7
- Momentum: 0.2

Data

Our best model had an accuracy of 90.6%.

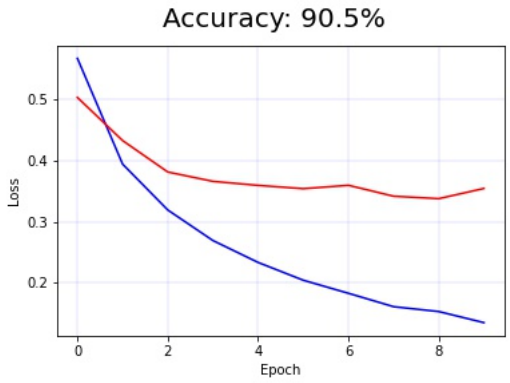

Figure 1: First folding

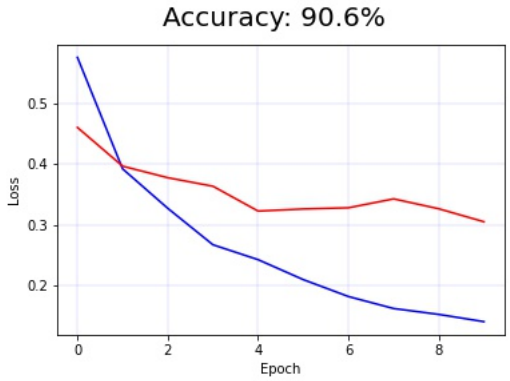

Figure 2: Second folding

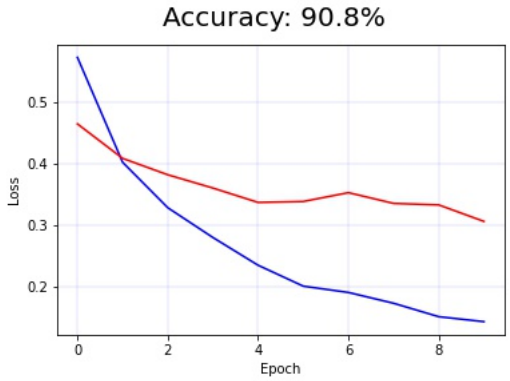

Figure 3: Third folding

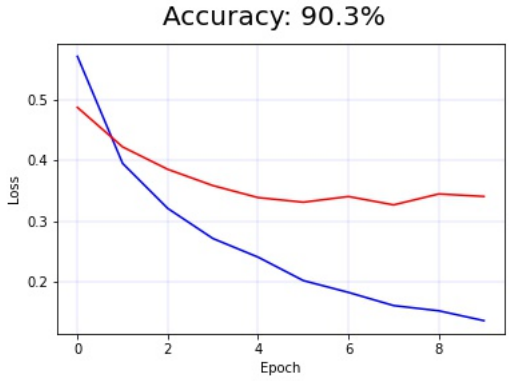

Figure 4: Fourth folding

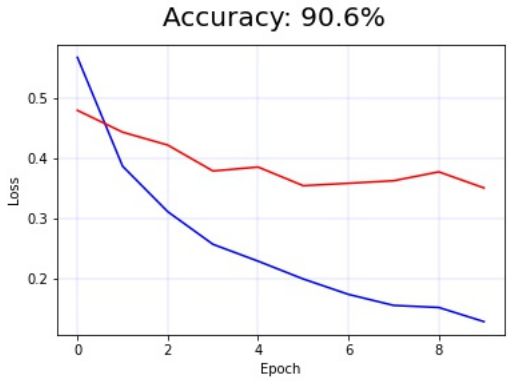

Figure 5: Fifth folding

References

[1] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: (Mar. 2015). URL: http://arxiv.org/abs/1502.03167.

[2] G. Myrianthous. "Feature Scaling and Normalisation in a nutshell". In: *Medium* (Jul. 05, 2020). URL: https://medium.com/analytics-vidhya/feature-scaling-and-normalisation-in-a-nutshell-5319af86f89b.