CPEN 355 - Final Project Report

Alex L. Lassooij
#38574752
Department of Electrical & Computer Engineering
alasso01@mail.ubc.ca

1 Problem Definition

Among the many viral diseases that plague the world's population, viral pneumonia is one of the most prevalent, causing severe and fatal complications for those who fall victim to it. It is estimated that 200 million people contract the disease annually, and is shown to cause many deaths of young children in developing countries and elderly people in developed countries.
Pneumonia can be caused by various respiratory viruses including human metapneumovirus, coronaviruses and human bocavirus. In particular, the novel coronavirus disease, COVID-19, caused by the SARS-Cov-2 virus discovered in 2019, has spread around the world and since induced a global health crisis. The COVID-19 virus causes pneumonia, but is known to move differently through its host's lungs compared to the aforementioned viruses, which is why is persists longer and causes more damage to the lungs [4]. Therefore, we generally differentiate between pneumonia caused by other common respiratory viruses (Viral Pneumonia) and the SARS-CoV-2 virus (COVID Pneumonia).
In the battle to cure patients of pneumonia infections, the diagnosis through X-Rays of the chest plays a key role. They help doctors identify the disease, allowing them to take required action.
In the following project, we will design a convolutional neural network to perform three category classification on X-Ray images in order to distinguish those infected with COVID Pneumonia and Viral Pneumonia, and those who are healthy. A highly accurate, automated classifier can serve as a useful auxiliary tool for clinicians, radiologists and doctors.

2 Dataset

The dataset consists of 317 gray-scale X-Ray images collected by researchers at the University of Montreal in 2019. To use it for this project, we download it through Kaggle's Public API. The dataset has already been divided into a training and testing set with 215 and 66 examples, respectively. The table below summarizes the count of each class, showing a slight class imbalance :

Covid	Normal	Viral
137	90	90

Table 1: Class Frequencies

Note that the number of samples is quite small, which will be taken into account and address in later steps.

3 Method

The design of the model will involve the following steps :

3.1 Data Preprocessing & Exploration

- Preparing and transforming data for input to a CNN
- Exploring data to observe and trends or patterns in data

3.2 Designing Model

- Selecting general model architecture
- Roughly choosing types, amounts and dimensions of layers based on widely known guidelines

3.3 Cross Validation & Tuning

- Fine-tuning hyperparameters and evaluating performance with K-Fold Cross Validation
- Introducing additional hyperparameters and components

3.4 Final Model Training & Testing

- Train final model on entire training set with selected hyperparameters
- Evaluate performance on unseen test set

4 Experiments

4.1 Network Architecture

To get a general idea on what architecture to select, I took inspiration from VGG16, which is widely known as one of the best CNNs for applications in computer vision [2]. It consists of 13 convolutional layers and 3 fully-connected layers. For our small dataset consisting only of grayscale X-Ray images of patients' chests, the VGG16 model is likely too complex.
Instead, a highly simplified version of VGG16 will be used as the network architecture, where the convolution kernel size decreases and the number of output channels increases with each convolutional layer.

4.1.1 Convolutional Layers

I started off with small with three convolutional layers where each had a kernel size of 3x3. Initially, the model was slightly overfitting, which might have been caused by the small amount of layers. I tried five convolutional layers which roughly cuts down the number of parameters in the first fully connected layer by a factor of 16. This lowered the level of over-parameterization in the model, leading to less overfitting
Furthermore, I used a kernel size of 5x5 for the first two layers to extract features in larger steps, resulting in no loss in performance. One layer of padding was added to reduce the loss of data in pixels on the perimeter of each image.

4.1.2 Batch Normalization

A significant jump in loss reduction was achieved by applying batch normalization after each convolution. Batch normalization combats the issue where the distribution of the inputs to each layer changes during training, reaching saturation regions and leading to loss of information, also known as internal covariate shift. An even further jump towards faster convergence was attained by adding a momentum of 0.90 to each batch normalization layer.

4.1.3 Fully Connected Layers

Initially, three fully connected layers were added to the model. The first FC layer had, by far, a much higher amount of parameters than the two following layers. Removing one FC layer and leaving just two in the model did not significantly affect its performance, suggesting it may have been redundant.

4.1.4 Dropout Layers

Introducing a dropout layer with a 50% dropout probability after the fully connected layer produced a strong decrease in overfitting. Now, the model was learning "less" from the training set, improving its ability to generalize [1].

4.1.5 Activation Functions

The Sigmoid and Tanh activation functions are known to be useful for binary classification and regression, but produced poor results during training. This is due to their limited output range and vanishing gradient for activation values near their saturation regions.
The Rectified Linear Unit activation function solves the vanishing gradient issue and performed much better. However, the images contain many dark pixels (around the patient's body and non-infected areas of the lungs) which are

represented by negative values after normalization. With a regular ReLU, this would cause a high amount of neuron death. By using a Parametric ReLu activation function, we avoid neuron death by multiplying negative values with a small weight instead of zero. This weight is very small compared to the weight for the positive region, which is desirable as the positive (bright, white coloured) pixels characterize the infected regions in the image.
After the last fully connected layer, softmax activation was used to produce a set of class probabilities that sum up to 1.0.

4.2 Loss Function

To calculate our loss, we use cross entropy loss as it is well suited computing loss for multi-class classification problems.

4.3 Optimizer

Stochastic Gradient Descent with mini-batches was used to minimize the loss of our model's output. I experimented with learning rates ranging from $[0.0005, 0.0015]$. The model's performance was extremely sensitive to small changes in the learning rate, but through trial and error, a learning rate of 0.00075 proved to be the best choice. Additionally, a momentum of 0.95 [5] was used to push the gradient into the "right" direction to accelerate convergence, especially in later stages of training.

4.3.1 Learning Rate Scheduler

A common occurrence was a highly varying error in later epochs, caused by a high learning rate that leads to oscillation around local minima. To avoid this, I let the learning rate decay [6] in steps of 4 by a factor of 0.85.

4.4 Epochs

The loss decreased sharply within the first 5 epochs, and decayed much slower in the following epochs. Between 25-30 epochs, the loss only decreased slowly, so I avoided training the model for more than 24 epochs to avoid overfitting the model.

4.5 Batch Size

Typical batch sizes range vary from a low of 32 and can reach sizes of up to 256. Since we are dealing with such a small dataset, using a large batch size lead to overfitting (generalization gap) and convergence to sub=optimal minima. Too small of a batch size caused underfitting and high variance in the estimate of the gradient, which was observed as a highly fluctuating accuracy and loss, leaving the model unable to converge to an optimal solution. For this model, a batch size of 8 was too small, and 32 caused the model to overfit, so I selected a final batch size of 16.

5 Results

5.1 Final Model

The final model consists of five convolutional layers and two fully connected layers, with a total of $1,582,547$ parameters. The diagram on the right shows a visual model with log-scaling applied.

Layer	# Parameters
Conv1	464
Conv2	12,928
Conv3	18,688
Conv4	74,240
Conv5	295,936
FC1	1,179,904
FC2	387

Table 2: Number of Parameters

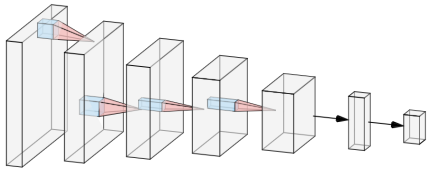

Figure 1: Scaled Network Model

5.2 Model Performance

Several hyperparameters and design choices were tuned, one at a time, through an iterative process, where each time the model's performance was evaluated through 5-fold cross validation. Finally, the model was evaluated on an unseen test

dataset of 66 samples. Both the validation and test accuracy curves rise sharply in the earlier epochs and stabilize after roughly 10 epochs. The accuracy continues to rise slowly, with slight fluctiations between each epoch. Typically, the model achieved a 92-94% cross validation accuracy and a 95-99% test accuracy, even reaching 100% accuracy in some instances. The slightly lower performance during cross validation is likely due to the smaller size of the datasets used.

	Accuracy	Loss
Validation	$93 \pm 1\%$	0.61 ± 0.2
Test	$97 \pm 2\%$	0.58 ± 0.3

Table 3: Model Evaluation Metrics

Pictured below are plots of the accuracy and loss obtained from (1) Cross Validation and (2) Testing :

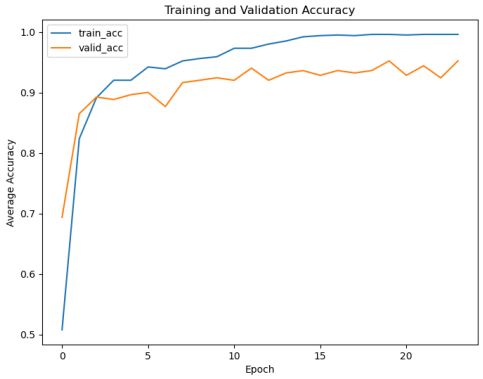

Figure 2: Validation Accuracy

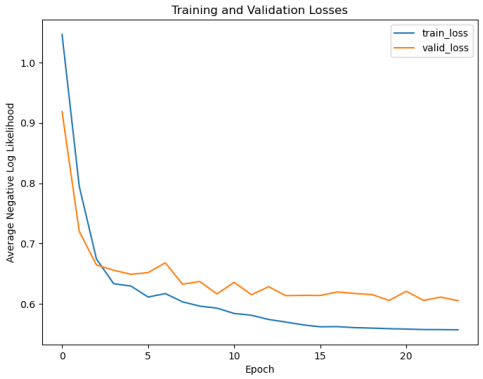

Figure 3: Validation Loss

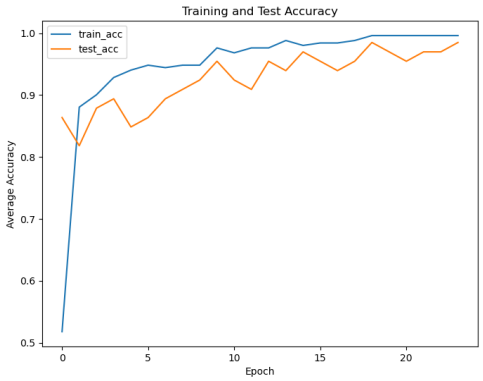

Figure 4: Test Accuracy

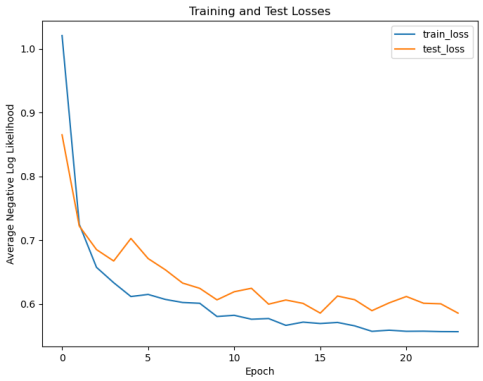

Figure 5: Test Loss

5.3 Conclusion

The model's performance is satisfactory, considering its simplicity compared to other computer vision models, as well as the limited size of the training dataset. With an average accuracy of $\approx 97\%$, it is reasonable to suggest its use in assisting clinicians and radiologists in practice. However, further optimization may have been achieved through applying ensemble methods, such as combining this model with a residual neural network. One concern is the high number of parameters in the first FC layer. Adding convolutional layers for dimensionality reduction showed to cause more overfitting. Possibly, Principal Component Analysis may be considered to extract important features and reduce the dimensionality of the model [3]. Lastly, it would be desirable to train the model on a larger dataset, as it would better represent the true population of interest and yield more reliable results. The notebook containing the model can be found here.

6 Implantation Environment

The model was trained in a local conda environment with Metal Performance Shaders (MPS) backend enabled for GPU training acceleration. Relevant libraries and their versions are listed below.

Package	Version
jupyterlab	4.0.9
kaggle	1.5.16
matplotlib	3.8.2
numpy	1.26.2
pandas	2.1.3
prettytable	3.9.0
python	3.10.13
pytorch	2.0.1
scikit-learn	1.3.2
scipy	1.11.4
torchvision	0.15.2
tqdm	4.66.1

Table 4: Library Versions

7 References

1. Hinton, Geoffrey E., et al. "Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors." arXiv.Org, Cornell University, 3 July 2012, arxiv.org/abs/1207.0580.
2. Kong, Lingzhi, and Jinyong Cheng. "Classification and Detection of COVID-19 X-Ray Images Based on DenseNet and VGG16 Feature Fusion." Biomedical Signal Processing and Control, U.S. National Library of Medicine, Aug. 2022, www.ncbi.nlm.nih.gov/pmc/articles/PMC9080057/.
3. Pradana, Ahmad Rizqi, et al. "Application of PCA-CNN (Principal Component Analysis – Convolutional Neural Networks) Method on Sentinel-2 Image Classification for Land Cover Mapping." Ijaers.Com, International Journal of Advanced Engineering Research and Science , 15 Aug. 2022, ijaers.com/issue-detail/vol-8-issue-9/.
4. Ruuskanen, Olli, et al. "Viral pneumonia." The Lancet, vol. 377, no. 9773, 2011, pp. 1264–1275, https://doi.org/10.1016/s0140-6736(10)61459-6.
5. Sutskever, Ilya, et al. "On the Importance of Initialization and Momentum in Deep Learning." Computer Science, University of Toronto, Proceedings of Machine Learning Research, 2013, www.cs.toronto.edu/ fritz/absps/momentum.pdf.
6. You, Kaichao, et al. "How Does Learning Rate Decay Help Modern Neural Networks?" OpenReview, ICLR, 5 May 2023, openreview.net/forum?id=r1eOnh4YPB.