SUTD

ISTD

50.039 Theory and Practice of Deep Learning

Small Project Report

Group 16
Xie Han Keong (100 3876)
Jisa Mariam Zachariah (100 3638)

1. Introduction

The small project is a project for students taking the 50.039 Theory and Practice of Deep Learning module in SUTD to get hands-on experience in designing, implementing, and training deep learning models to solve real world problems. In this project, we were tasked to develop a deep learning model to classify chest X-ray images to detect if a patient has pneumonia or not and whether the pneumonia is caused by Covid-19. As this project is a primer for a big project later on, the focus of this project is on model design and training and less on data preparation. As such, a curated dataset was provided together with guided notebooks on how to implement custom PyTorch datasets and data loaders that are needed for training our models. The objective of this project is to propose, train, and evaluate a deep learning model with an emphasis on some of the important concepts and good practices that are useful for many projects related to deep learning.

2. Dataset and Data Loader

The dataset that was provided consists of 5,854 chest X-ray images in grayscale. The images have size 150 by 150 pixels and are labelled with 3 classes: Normal, Infected without Covid, and Infected with Covid. Figure 1 shows a sample of images in each class. Also, the dataset was pre-split into train, test, and validation datasets. Figure 2 shows the distribution of all the images in each class across these datasets. Although the validation dataset is perfectly balanced, the train and test dataset have greater weight in the ‘Infected without Covid’ class as compared to ‘Normal’ and ‘Infected with Covid’. We address this by using a weighted sampler in our data loaders to sample data more representatively.

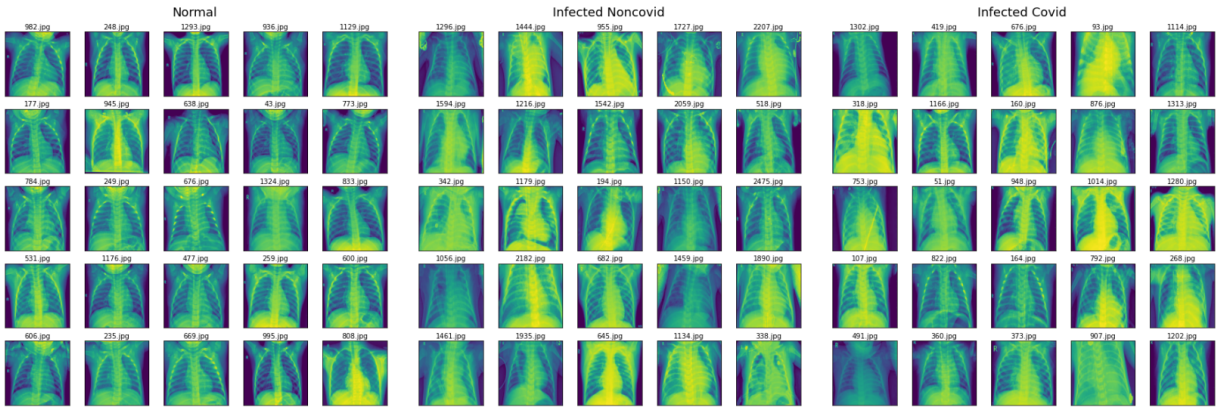

Figure 1: A random sample of 25 X-ray images from each class

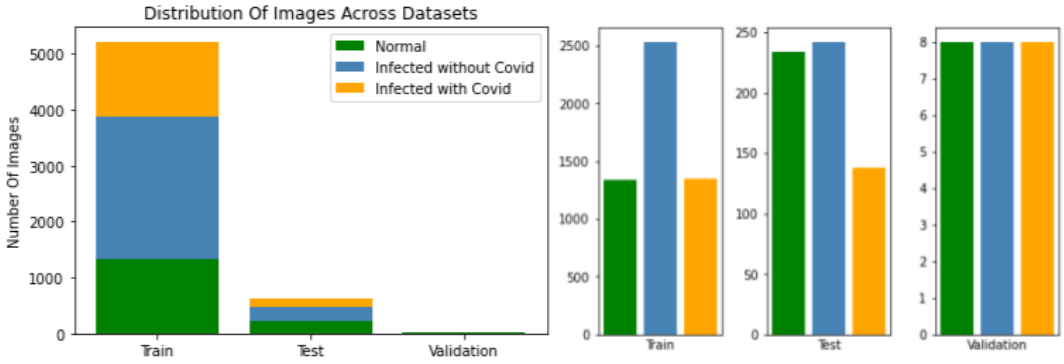

Figure 2: Distribution of images from each class across datasets

Data Augmentation. We implemented a custom dataset and data loader to load these images into batches and performed data augmentation during training by applying two transformations to the images. Firstly, a random clockwise or counter-clockwise rotation of 5 degrees and a horizontal translation of -0.1 to 0.1 of the original image size was made. This is because we observed that many of the images have empty spaces on the left and right and some are also rotated at a slight angle. Also, a similar approach was adopted from a recent paper in 2020 that used CNNs for pneumonia detection [4]. Following that, a random crop between 0.9 to 1.0 of the original image size and a random aspect ratio between 0.9 to 1.1 of the original aspect ratio was made, and the result was resized back to 150 by 150 pixels. This is because we observed that there were many images with varying body sizes. Also, a similar approach was popularly used to train the InceptionV1 networks. Finally, the image values were normalized by converting them into tensors, which is the required data format for PyTorch. Figure 3 shows a sample of augmented images from the training dataset. Using this approach, we drew out 6,400 samples from the augmented training dataset to train each model. We chose not to apply color adjustments such as brightness or contrast jitter because that may affect the predicted class of the image. We

also chose not to apply flipping because the relative position of the lungs is important for pneumonia detection and should be preserved. However, we considered test-time augmentation as a good approach for improving the performance of the model, but we were not able to implement it within the time frame of the project.

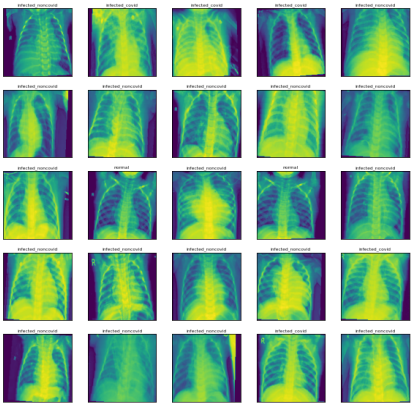

Figure 3: A random sample of 25 augmented images from the training dataset

3. Proposed Model

Two approaches for classifying the images are a 3-class classifier that classifies all three classes directly, or a dual binary classifier that classifies an image as normal or infected first, and then classifies the infected images as infected without Covid or infected with Covid. We think that a dual binary classifier will be better because the difference between non-Covid and Covid images is very subtle whereas the difference between normal and infected images is very obvious. It is not possible to take advantage of this observation when training a 3-class classifier. However, a binary classifier can be trained for these two classification tasks with the corresponding level of depth and complexity required to learn the features required to tell apart the subtle differences between non-Covid and Covid images. In order to verify this hypothesis, we decided to train both a 3-class classifier and a dual binary classifier to compare their performance on the validation set.

Model Architecture. We decided to adopt the DenseNet architecture since it was found that DenseNet outperformed other deep CNNs such as AlexNet, ResNet, and SqueezeNet in both normal/pneumonia and bacterial/viral pneumonia detection tasks [4]. The advantage that DenseNet has over other CNNs is that convolutional layers are able to access the feature maps from all preceding layers in each dense block. This allows the model to use low-level and mid-level features to construct higher level features deep inside its network. This allows the model to extract better high-level features and learn more accurately.

Implementation. We wrote an implementation of DenseNet with reference to the PyTorch source code [3]. Even though a DenseNet201 model was trained in [4] using pre-trained weights, we could not use pre-trained weights as transfer learning is not allowed in this project. As such, we trained our DenseNet models from scratch. Table 1 shows the DenseNet parameters that we used for our models.

Model Parameters	3-Class Classifier	Dual Binary Classifier	
		Normal/Infected Classifier	Non-Covid/Covid Classifier
growth_rate	18	18	18
block_config	(6, 12, 32, 32)	(6, 12, 24, 16)	(6, 12, 32, 32)
num_init_features	36	36	36
bn_size	4	4	4
drop_rate	0	0	0
num_classes	3	2	2

Table 1: DenseNet parameters used for each model

Model Parameters. We decided to follow [2] and set the growth rate to 32 to ensure that adequate features are being added in each dense layer. At the same time, we set the number of initial features to twice the growth rate. Also, we set the bottleneck size to 4 to adequately reduce the number of input features maps at the start of each convolutional layer to compress the size of the convolutional layers and prevent overfitting. In addition to that, no drop rate was employed during training. The number of classes chosen adheres to the corresponding classification tasks.

Block Configuration. Since the differences between normal and infected images are very obvious, we decided to choose the DenseNet121 block configuration which was the smallest block configuration discussed in [2] for the normal/infected classifier. For the 3-class classifier and the non-Covid/Covid classifier, we chose the DenseNet169 block configuration to allow better high-level features to be learnt to enable the model to identify the subtle differences between non-Covid and Covid images more accurately.

Training Details. We selected a mini-batch size of 64 instead of 256 in [2] because the training dataset only has 5,216 samples. We wanted a smaller mini-batch size to allow the model to converge faster and generalize better, and we also wanted to ensure that there are enough mini-batches for each training epoch. This is one reason why we decided to increase the number of training samples to 6,400 using data augmentation to allow for 100 mini-batches per epoch. We employed early stopping with a patience of 30 to determine the optimum number of epochs to train for without overfitting. We decided to use Adam for optimization even though classic SGD was used in [2] as it typically converges faster than other optimizers. Due to lack of time, we only tuned the learning rate and set all other

hyperparameters such as beta and the weight decay to the default values. The learning rate that we found performed the best was 0.0001. The model parameters were initialized using Kaiming He initialization, which is the default PyTorch implementation for Linear and Conv2d layers. This is to prevent the problem of exploding or vanishing gradients during backpropagation especially for deep neural networks such as DenseNet, which is a deep CNN, by keeping the variance of all layers equal while taking into account the use of non-linear activation functions like ReLU. The class weights of the training dataset were used to calibrate the negative log loss function to prevent the model from learning any biases due to the imbalance of classes in the training dataset.

In our training function, tqdm was used to display the progress over all epochs and the number of iterations in each epoch. In each epoch, the model was first trained using the train data loader, and then evaluated using the test data loader. The loss and other metrics such as the accuracy, precision, recall, and F1 score were calculated by the weighted average at the end of each iteration and averaged across all iterations. The scores on the train and test datasets were reflected together with the timestamp and the time elapsed after every epoch. Also, the minimum test loss was tracked as well as the number of 'bad' epochs where the test loss was higher than the minimum test loss. Every time a new minimum test loss is achieved, the model is saved and the 'bad' epochs are reset. However, if the number of 'bad' epochs surpasses the patience threshold, early stopping is activated and the training is terminated, and the minimum test loss and the epoch at which it was achieved is displayed.

4. Results

Figure 4 shows the learning curves of the models that we trained. The normal/infected classifier (top-left) was trained to classify both non-Covid and Covid images as infected. It was able to achieve an accuracy of 0.833 on the validation set after 6 epochs.

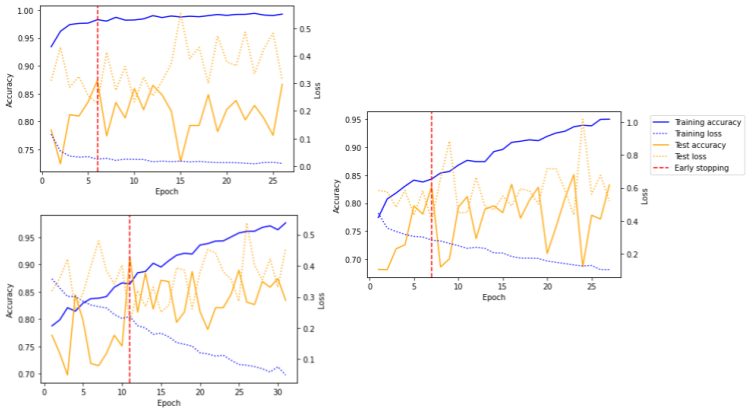

Figure 4: Learning curves of each model

The non-Covid/Covid classifier (bottom-left) was trained only on non-Covid and Covid images. It was able to achieve an accuracy of 0.625 on the validation set after 11 epochs. The 3-class classifier (right) was trained on all three classes at once, and it achieved an accuracy of 0.583 on the validation dataset after 7 epochs. Overall, the binary classifiers were able to achieve higher accuracies than the 3-class classifier. This is expected as binary classification is an easier task than 3-class classification.

After the two binary classifiers finished training, they were combined into a single dual binary classifier. The dual binary classifier classifies images by first passing them through the normal/infected classifier. The outputs are used to find which images are classified as infected, and these images are passed into the non-Covid/Covid classifier unchanged. The outputs are then used to determine whether these are infected without Covid or infected with Covid images. The dual binary classifier only managed to achieve an accuracy of 0.542, which is slightly worse than the 3-class classifier (See Table 2). This does not support our hypothesis that the dual binary classifier is the better choice when predicting chest X-ray images infected with or without Covid-19. We think it is because the error rate of the non-Covid/Covid classifier affected the overall error rate of the dual binary classifier. Better performance could be obtained if the accuracy of the non-Covid/Covid classifier is improved.

Model Validation score	Dual Binary Classifier	3-Class Classifier
Accuracy	0.542	**0.583**
Recall	0.582	0.613
F1-score	0.561	0.593

Table 2: Results of each model on the validation dataset

Metrics. Accuracy may not be the best metric for detecting patients infected with Covid-19, especially because the number of non-Covid images in the training dataset outweighs the number of Covid images. Because Covid-19 is a very contagious virus, it poses a danger and a high risk to society in the case of a false negative. Therefore, recall is a better metric for this problem, since it measures how many infected with Covid images were correctly classified out of all possible infected with Covid images. The 3-class classifier was also able to achieve a higher recall of 0.613 as compared to the dual binary classifier.

Limitations. One constraint of the dataset is that there are several images of very young and thin patients who are most probably children (See Figure 5). This might have affected the learning of the models as the training dataset was not specifically curated for images of children, which limits learning features that enable detection of a Covid infection in a

smaller and thinner body frame like that of a child. There could be possible improvement if the model is trained on more of such images. Also, one downside to the validation dataset is that it only consists of 24 images. Hence, the results on this small dataset might not accurately represent the model's performance on new data. Finding the results on a larger validation set might be a more accurate way of evaluating the model's performance in detecting pneumonia with or without Covid-19.

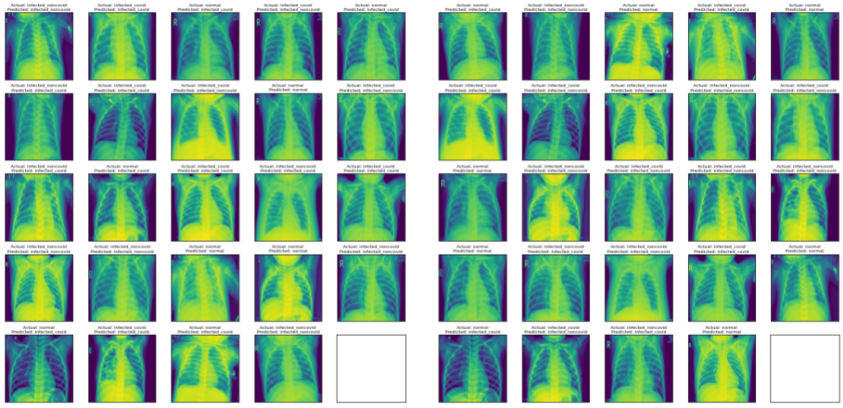

Figure 5: Actual and predicted labels of both models on the validation dataset

References

[1] Detecting Pneumonia Using CNN In Pytorch. (n.d.). Retrieved March 21, 2021, from https://www.kaggle.com/c/detecting-pneumonia-using-cnn-in-pytorch/overview/description

[2] Huang, G., Liu, Z., Maaten, L. V., & Weinberger, K. Q. (2018, January 28). Densely Connected Convolutional Networks [Scholarly project]. Retrieved March 20, 2021, from https://arxiv.org/pdf/1608.06993.pdf

[3] Pytorch. (2020, November 19). Pytorch/vision. Retrieved March 21, 2021, from https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py

[4] Rahman, T., Chowdhury, M. E., Khandakar, A., Islam, K. R., Islam, K. F., Mahbub, Z. B., . . . Kashem, S. (n.d.). Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection Using Chest X-ray [Scholarly project]. Retrieved March 20, 2021, from https://arxiv.org/ftp/arxiv/papers/2004/2004.06578.pdf