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 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 sets. 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.

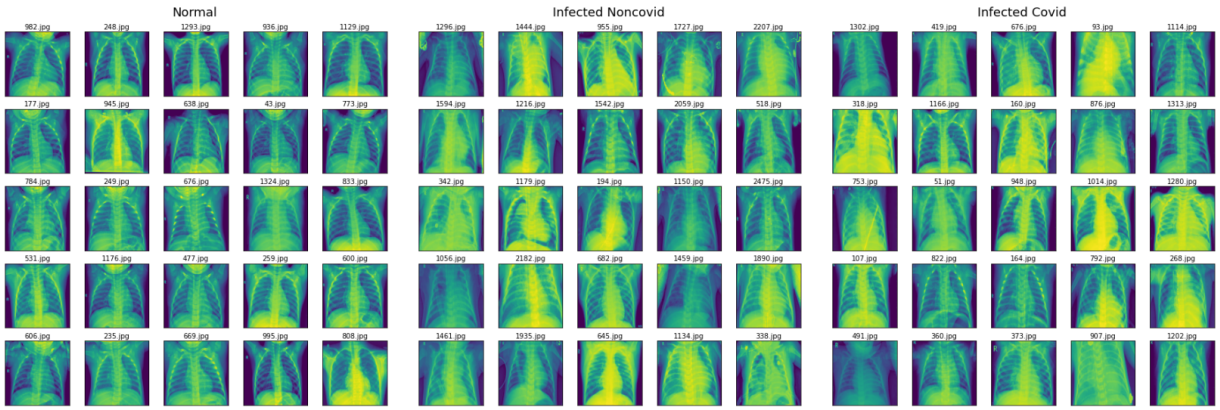

Figure 1: A random sample of 25 X-ray images from each class

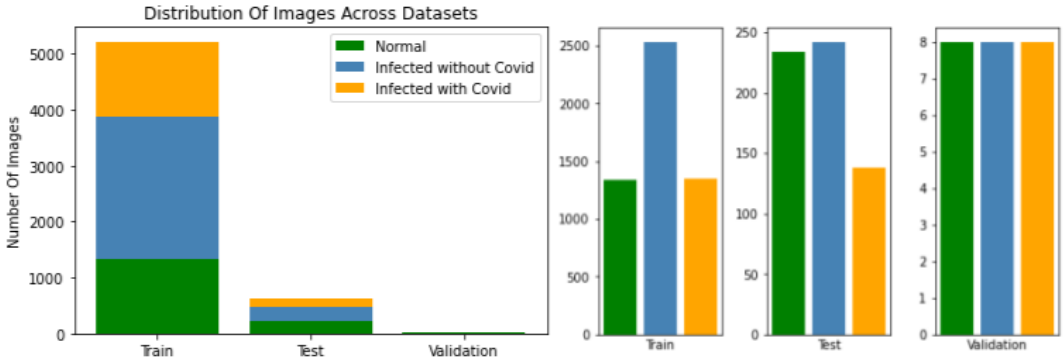

Figure 2: Distribution of images from each class across datasets

Data Augmentation. We implemented a custom dataset and data loader to load these images into batches and performed data augmentation during training by applying three transformations to the images. Firstly, a random crop between 0.08 to 1.0 of the original image size and a random aspect ratio between 3/4 to 4/3 of the original aspect ratio was made, and the result was resized back to 150 by 150 pixels. This approach was popularly used to train the Inception networks. Secondly, a random translation of -0.1 to 0.1 of the original image size was made in both the horizontal and vertical direction. This approach was adopted from a recent paper in 2020 which used CNNs for pneumonia detection [4]. Lastly, a color jitter of a random factor between 0.9 and 1.1 was applied to the brightness, contrast, and saturation. This was suggested in the guided steps in a Kaggle competition on pneumonia detection using CNNs [1]. Finally, the image values were normalized by converting them into tensors, which is the required data format for PyTorch. We chose not to apply cropping or flipping because the relative position of the lungs is important for pneumonia detection and should be preserved. Also, 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.

3. Proposed Model

Two approaches for classifying the images are a 3-class classifier which classifies all three classes directly, or a dual binary classifier which will classify an image into 'Normal' or 'Infected' first, followed by classifying infected images into 'Infected without Covid' or 'Infected with Covid'. We think that a dual binary classifier is better because the difference between 'Infected without Covid' and 'Infected with Covid' images is very subtle whereas the difference between 'Normal' and 'Infected' 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 subtle features between 'Infected without Covid' and 'Infected with Covid'. 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 were not able to train a DenseNet201 model as we did not have enough compute resources and we could not use pre-trained weights as transfer learning is not allowed in this project. As such, we created custom DenseNet models and trained them 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	16	8	8
block_config	(4, 8, 16, 12)	(4, 8, 16, 12)	(6, 12, 24, 18)
num_init_features	32	16	16
bn_size	4	4	4
drop_rate	0	0	0
num_classes	3	2	2

Table 1: DenseNet parameters used for each model

Growth Rate. We chose a growth rate of 16 and 8 instead of 32 like in [2] for two reasons. Firstly, this growth rate was used in models trained on ImageNet pictures which are 224 by 224 pixels and in RGB. However, our model is going to be trained on images which are 150 by 150 pixels and in grayscale. This means there is less information to learn, which means that the number of additional feature maps in each layer can be reduced. Using the same logic, we set half the growth rate for the binary classifiers as compared to the 3-class classifier since less features are needed to classify between two classes.

Block Configuration. We chose a block configuration of (4, 8, 16, 12) over the classic block configurations for DenseNet for the same reason we chose a smaller growth rate. Since there is only 1 input channel for grayscale images as compared to 3 for RGB images, the amount of information is correspondingly only 1/3. Thus, we decided to reduce the block sizes by roughly 1/3 as fewer features need to be learnt. The reason why the non-Covid/Covid classifier has a block configuration of (6, 12, 24, 18) is to allow more high-level features to be learnt to enable the model to identify the subtle differences between non-Covid and Covid X-ray images more accurately.

Other Model Parameters. We decided to follow the reasoning in [2] and set the number of initial features to twice the growth rate. For this reason, the bottleneck size was also set to 4 to reduce the number of input features maps at the start of each convolutional layer, and no drop rate was employed during training. The number of classes were set to adhere to the corresponding classification tasks.

Training Details. We selected a batch size of 64 instead of 256 in [2] for two reasons. Firstly, the training dataset only has 5,216 samples as compared to the millions of images in ImageNet that were used to train the classic DenseNet. We need to ensure that there are enough mini-batches for each training epoch to allow for smoother training. Secondly, we did not have enough compute resources to experiment with batch sizes larger than 64. We chose 50 epochs instead of 90 in [2] because we determined from experiments that any higher would lead to overfitting. We decided to use Adam for optimization even though classic SGD was used in [2] as it typically converges faster than other optimizers. We chose an initial learning rate of 0.01 and employed a scheduler to decrease the learning rate by a factor of 10 every 20 epochs. This is similar to [2] where the learning rate was initially set to 0.1 and decreased every 30 epochs by a factor of 10. Other parameters such as beta and the weight decay were not tuned and were set to the default values. 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.

We wrote a training function that uses tqdm to display the progress over all epochs and the number of iterations in each epoch. In each epoch, the model is 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 are calculated at the end of each iteration and averaged across all iterations. The scores on the train and test datasets are reflected together with the timestamp and the time elapsed after every epoch. Finally, we let the training function save the model and the results of training every 30 epochs to ensure that progress is not lost if training is interrupted.

4. Results

Figure 3 shows the learning curves of the models that we trained. The normal/infected classifier (top-left) was trained to classify both infected without Covid and infected with Covid images as infected. Amazingly, it was able to achieve an accuracy of 1.0 on the validation set. The non-Covid/Covid classifier (bottom-left) was trained only on infected without Covid and infected with Covid images. From the curve, we can observe that the learning was not very stable. This is expected as the differences between infected without Covid and infected with Covid images are more subtle and thus the features are harder to learn. Also, we can observe slight overfitting starting from epoch 35. Nonetheless, it was able to achieve an accuracy of 0.833 on the validation set. The 3-class classifier (right) was trained on all three classes at once. Despite slight overfitting starting from epoch 25, the model was still able to achieve an accuracy of 0.792 on the validation dataset. 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.

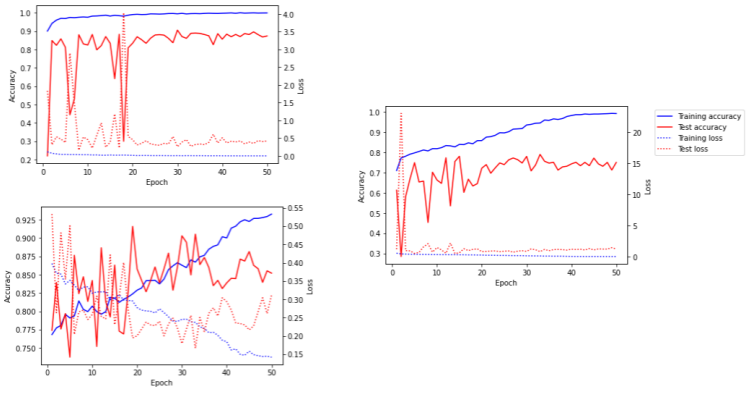

Figure 3: Learning curves of each model

After the two binary classifiers finished training, we combined them 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 managed to achieve an accuracy of 0.833, which is slightly better than the 3-class classifier (See Table 2). This proves that our hypothesis is correct, and the dual binary classifier is the better choice when predicting chest X-ray images infected with or without Covid-19. Figure 4 shows the actual and predicted labels of the dual binary classifier (left) and 3-class classifier (right) on the validation dataset.

Model Validation score	Dual Binary Classifier	3-Class Classifier
Accuracy	**0.833**	0.792
Recall	0.846	0.776
F1-score	0.829	0.760

Table 2: Results of each model on the validation dataset

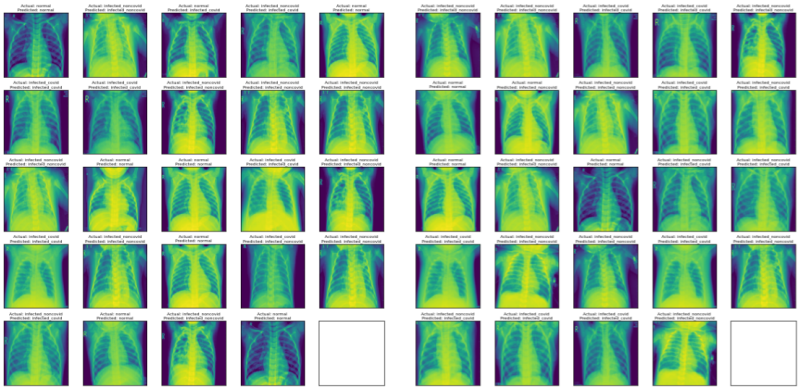

Figure 4: Actual and predicted labels of both models on the validation dataset

Metrics. Accuracy may not be the best metric for detecting patients infected with Covid-19, especially because the number of infected without Covid images in the training dataset outweighs the number of infected with 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. Fortunately, the dual binary classifier was able to achieve a higher recall of 0.846 as compared to the 3-class classifier as well.

Limitations. Even though the dual binary classifier was able to achieve good results, it was not able to predict all the images in the validation dataset correctly. There is even one infected with Covid image which was wrongly classified as infected without Covid, which is a very bad situation to have. One possible reason for this specific shortcoming is that the image is of a very young and thin patient, which most probably is a child. This can be considered as an anomaly as the model was not specifically trained on images of children, and thus the features it learnt might not be able to pick up signs of a Covid infection in a smaller, thinner body frame in 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 predicting pneumonia with or without Covid-19.

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