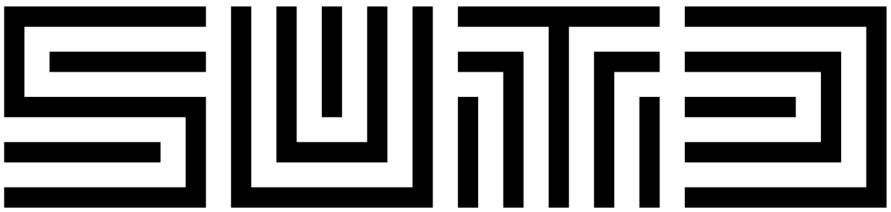


SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.039 Deep Learning

Project Report

Tao Sihan - 1005515

Wang Siyang - 1005485

GitHub Repo: https://github.com/sihvn/DeepLearningProject

Table of Contents

1. Introduction

The Chest X-ray (CXR) examination is widely utilised as a cost-effective means of diagnosing issues within the cardiovascular and pulmonary systems. Despite its prevalence, clinical interpretation of chest X-rays can pose challenges.This project aims to develop a deep learning model for detecting the presence of Pneumothorax in patients.

2. Data Preparation

2.1. Dataset

The NIH Chest X-ray Dataset [1] comprises 112,120 X-ray images with disease labels from 30,805 unique patients, encompassing 14 disease labels: Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema. Emphysema, Fibrosis, Effusion, Pneumonia, Pleural_thickening, Cardiomegaly, Nodule, Mass and Hernia. All images are in 1024 x 1024 resolution.

2.2. Pre-processing

X-ray images labelled with Pneumothorax will be the positive class, while all others will be considered the negative class.

Out of the 112,120 X-ray images, only 5,302 have labels containing Pneumothorax, which accounts for about 4.7% of the entire dataset. This results in a significant imbalance between the positive and negative class, potentially leading to high accuracy for predicting the majority class while failing to correctly identify the minority class. To address this issue, we undersampled the majority class to match the size of the minority class, ensuring a balanced distribution. After undersampling, we shuffled and split the dataset into training, validation and test dataset in an 8:1:1 ratio.

Training dataset size	Validation Dataset size	Test Dataset Size
8484	1060	1060

Figure 1: Training, Validation and Test Dataset size

3. Task

Based on the dataset, there were two possible tasks: one for a 14-label multi-label classification task, or a binary classification task on a single label. While the paper that did the multi-label classification task was able to achieve a respectable AUROC score of 0.84, the corresponding F1 score was much lower at 0.43. This was then validated by replicating the same experiment on our own machine.

For the binary classification task, we chose the label Pneumothorax out of the other 13, as one of the students has domain expertise from being a 4 time patient with Pneumothorax. In comparison, the paper that carried out the binary classification for Pneumothorax was able to achieve an F1 score of 77%.

While the multi-label classification model was able to discern between the positive and negative samples in the dataset, it was not able to achieve good precision or recall on all of the 14 labels at the same time. Hence, for this project, we will focus on a binary classification task for Pneumothorax, while attempting to replicate the performance of the related research paper with significantly less computing resources and less labels.

4. Model

During the initial phase, two model architectures were considered: Vision Transformers and CNNs. Since the original two papers that we used as reference were only worked with CNNs, we wanted to see if recent advancements in transformers could be useful in improving the performance of the models.

4.1. Transformers vs CNNs

Using MaxVit and Google's ViT-Base16 models that were downloaded from torchvision, we compared it against our implementation of DenseNet-121 that carried out the 14-label multi-label classification task.

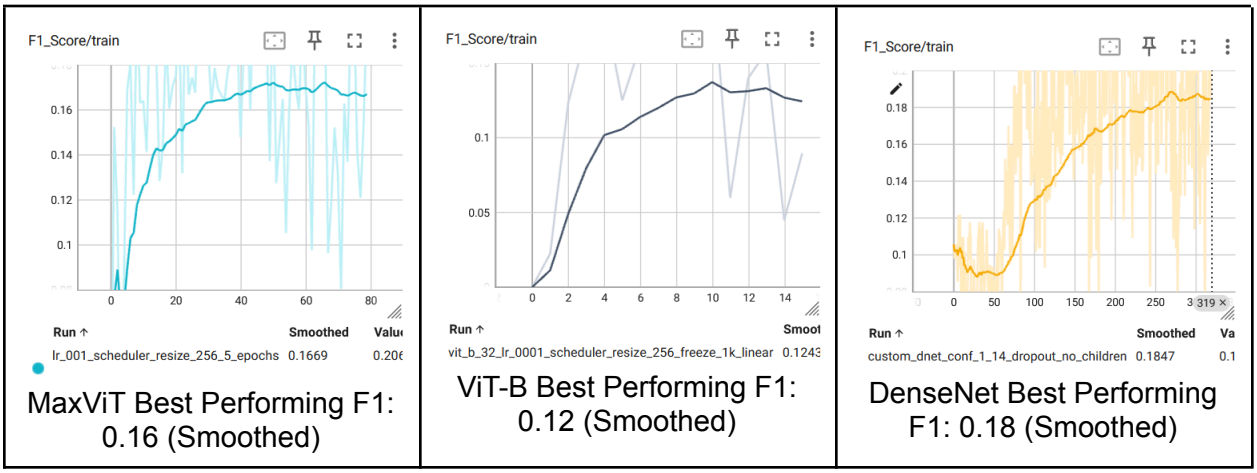


Figure 2: Comparison between MaxViT, ViT-B and DenseNet

Through experimental results over a short number of epochs, DenseNet-121 was shown to have a better F1-score, compared to MaxViT and ViT-B. Therefore, for the rest of the project, DenseNet was chosen as the model architecture. The task at this point was then further simplified to a Binary Classification problem, as mentioned in the previous section.

4.2. Building custom DenseNet

Our project implements a convolutional neural network (CNN) architecture inspired from DenseNet-121 [2]. DenseNet extends the logic of Skip Connections in ResNets further by connecting each layer in a Dense Net block to one another. This allows for the model to build up to a deep depth, without losing initial information.

After the DenseNet model architecture model was chosen, we worked to re-write the model using PyTorch modules, instead of importing the model architecture from Torchvision. After implementing the basic model, experiments were also carried out to verify the F1 score of various modifications of the DenseNet model.

4.2.1. Standard DenseNet-121 Model

The standard DenseNet-121 Model consists of an initial convolution and pooling stem. The pictures then pass through a series of four DenseBlock and Transition Layers. Each DenseBlock and Transition Layer pair work together to increase the number of channels and reduce the feature map dimensions respectively.

Last but not least, it passes through a Classification Layer. In the classification layer, the output from the last Transition Layer is passed through an Average Pooling layer to reduce the size of each feature map down to 1. The channels are then flattened to return a vector, which is then passed through a fully connected layer and a sigmoid function in the end.

Further experiments on this model are described in Section 5.

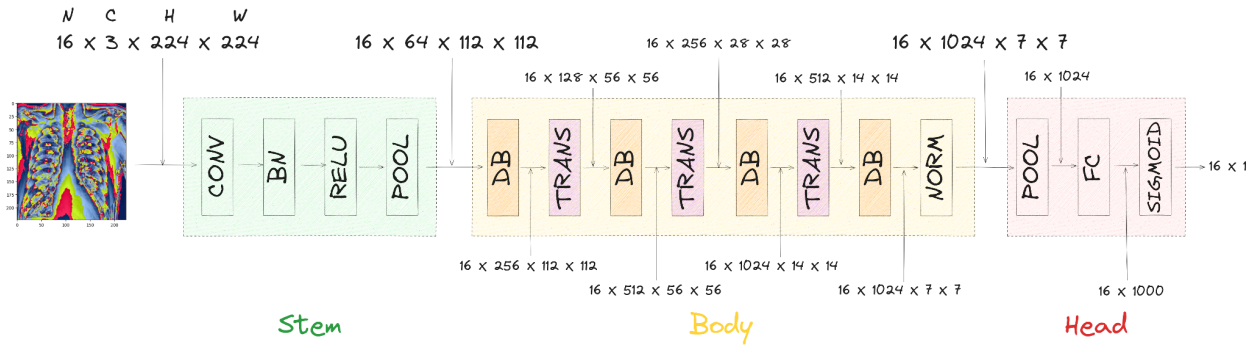


Figure 3: Overall diagram of the DenseNet Model, where batch size = 16

4.2.2. Add non-image data

Besides the pictures, the dataset also provided discrete data (# of follow ups, age, gender). The model was modified to take in non-image data after the image has passed through convolutional layers. Since pneumothoraces tend to occur more often in males of young age, and more likely to occur in patients with previous occurrences, it would have been helpful to add this information into the model with the images. Although the correlation in the dataset was not strong, the bar graphs reflected the trends as described previously.

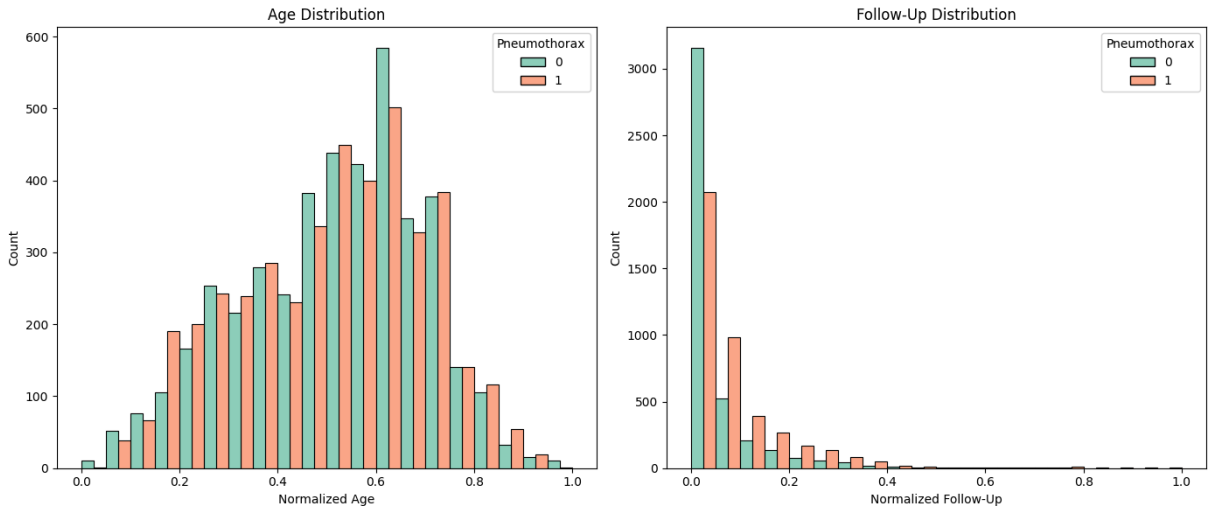


Figure 4: Age and # of follow-up distribution between Chest X-ray image with Pneumothorax (label 1) and without Pneumothorax (label 0)

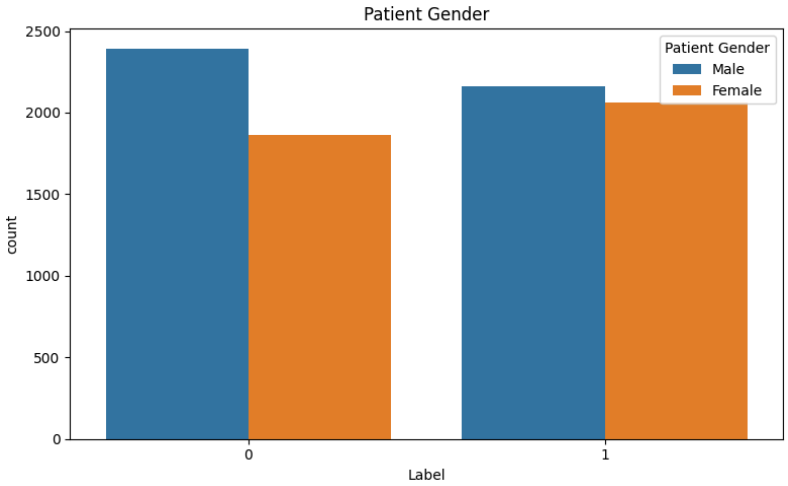


Figure 5: Gender distribution between Chest X-ray image with Pneumothorax (label 1) and without Pneumothorax (label 0)

The information from discrete data is concatenated to the image after it has passed through all convolutional layers and flattened into a vector. The discrete data was concatenated with the image data in various ways, such as 1) directly, and 2) after passing through fully connected layer(s).

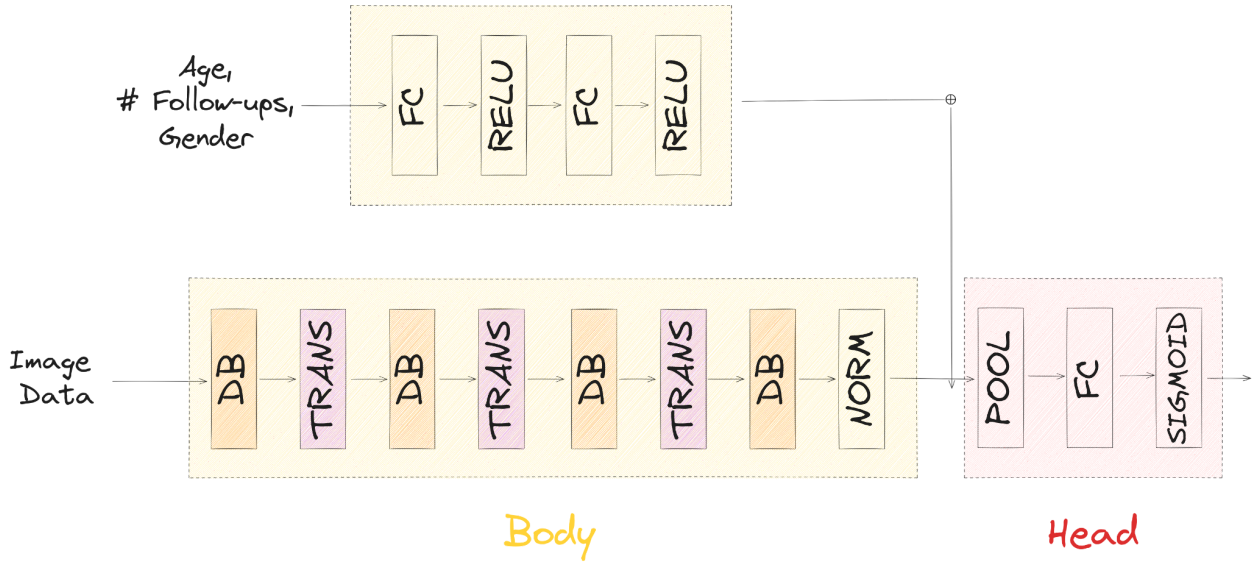


Figure 6: Concatenation of non-image data with image data, after it passes through fully connected layers

Based on our experiment, the new model underperformed compared to the original image only model. (**0.55 vs. 0.6 F1 score**) This could be due to an un-optimal design of the fully connected network for the non-image data, or it could also point towards the low utility of including the non-image data in the model training.

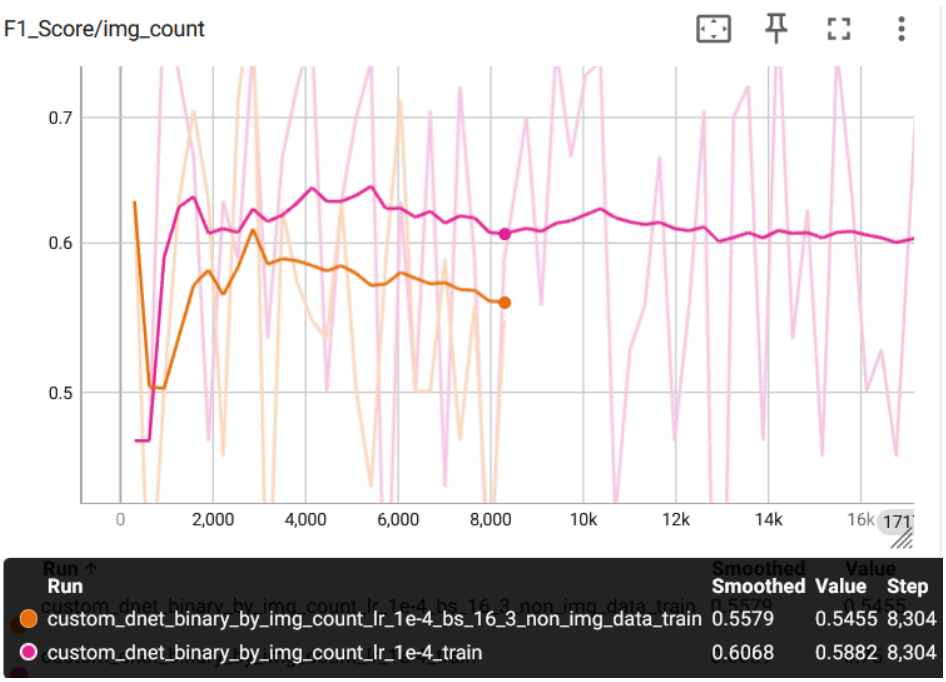


Figure 7: Results of vanilla DenseNet model vs model trained with non-image data, on Train Dataset

4.2.3. Take in data as grayscale images

Initially, the images were converted from grayscale to RGB and then normalised with ImageNet mean and standard deviation. However, since the original pictures were in grayscale format, it might be better to instead normalise each image to have a mean of 0 and a standard deviation of 1. In addition, since we had been experimenting with different variations of the Densenet model, we could not leverage on the pre-trained weights in ImageNet, which were less helpful in this case as it does not contain chest X-Ray images.
Normalisation was done for each image instead of across the dataset to avoid leakage of information from the test/val dataset into the train dataset.

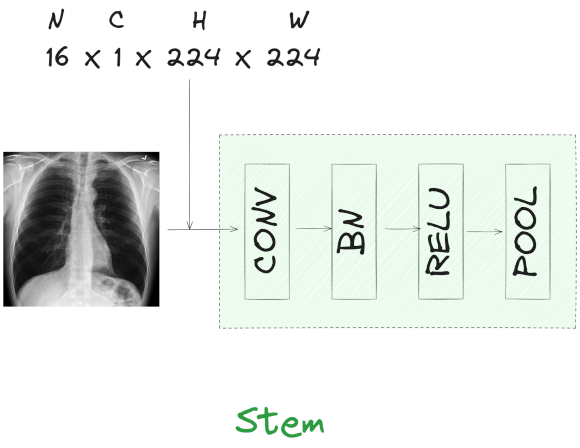


Figure 8: Use grayscale images instead of RGB -- note that there is only 1 channel

Based on our experiment, the new model underperformed compared to the original RGB-image trained model on the train dataset (**0.50 vs. 0.54 F1 score**).

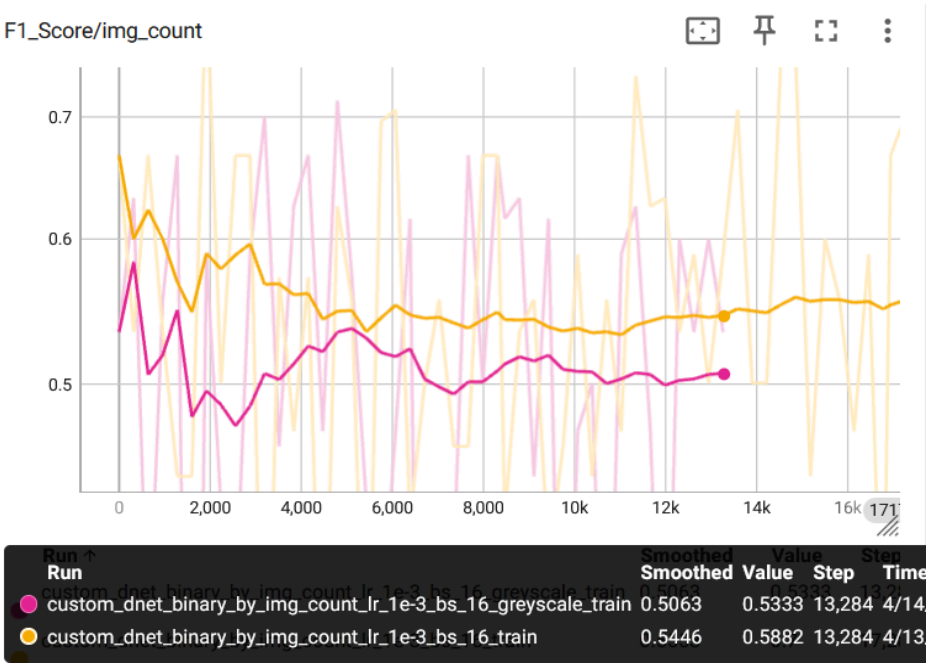


Figure 9: Results of vanilla DenseNet model vs model trained with grayscale images, on Train Dataset

4.2.4. Take in data with larger dimensions

Initially, the DenseNet-121 was chosen also because of the small input size (224 by 224 pixels), which helped to reduce the time taken to train per epoch. However, given that the original images had a much larger dimension (1024 by 1024 pixels), many details were lost when the images were downsampled. At the same time, pneumothoraces present themselves in many different sizes, and a larger initial input size can help the model to better generalise to the different cases.

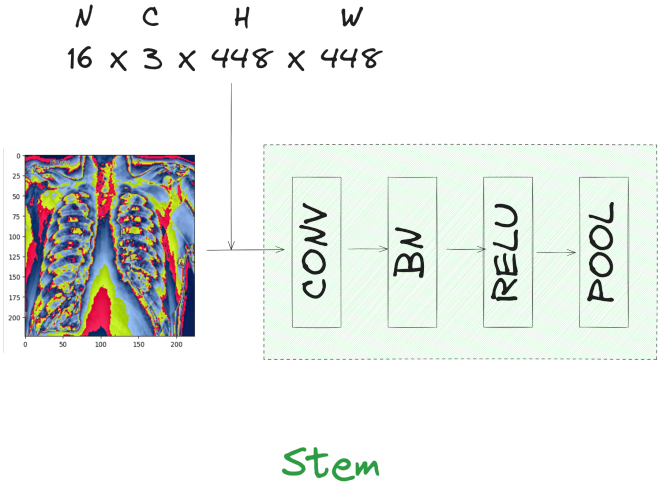


Figure 10: Use higher resolution images from the start -- note the increase Height and Width

Based on our experiment, the new model underperformed compared to the model trained with smaller images initially. (**0.49 vs. 0.57 F1 score**)

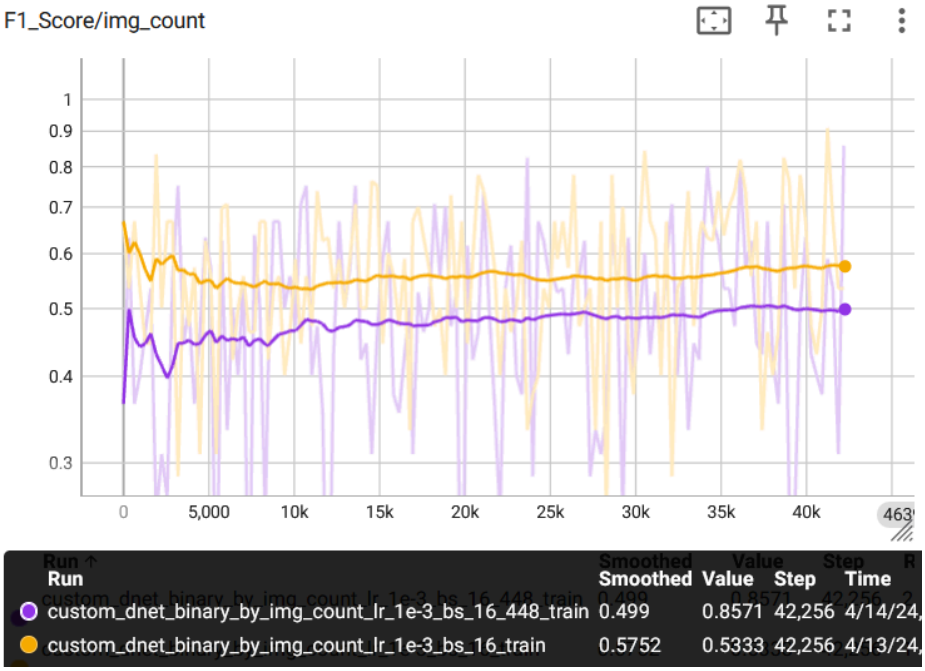


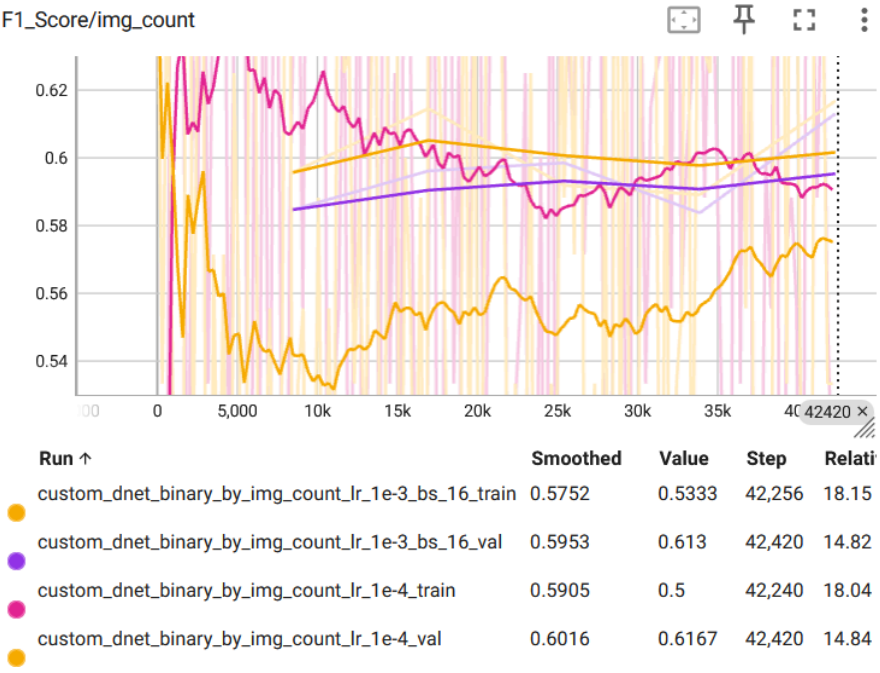
Figure 11: Results of vanilla DenseNet model vs model trained larger images, on Train Dataset

5. Training

Each of the models were then trained for at most 5 epochs as a proxy of potential model performance after sufficient number of epochs. Based on these results, we also tried tuning the hyper-parameters on the best performing model during training.

5.1. Learning rate

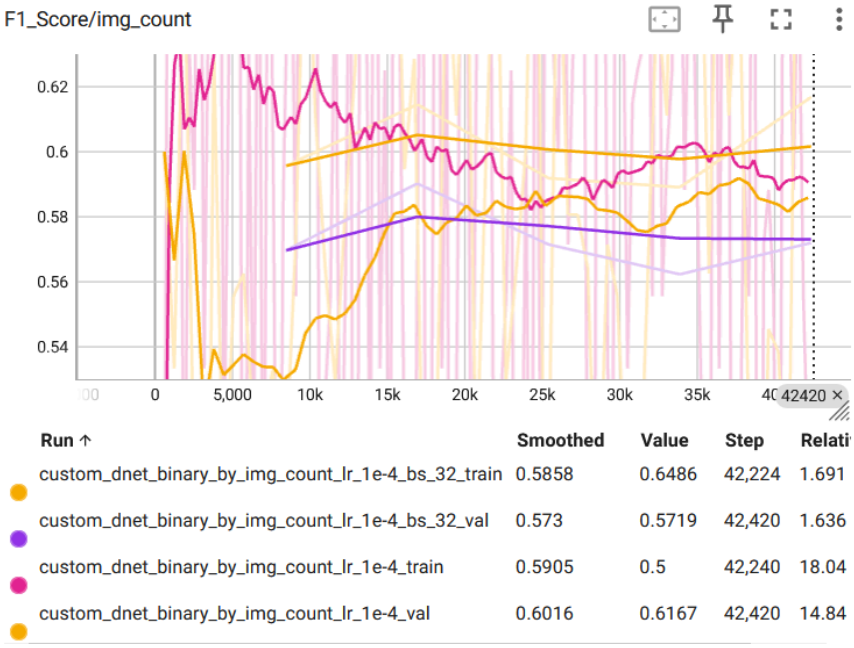
We experimented with learning rates 1e-4 and 1e-3. While both did not achieve a good F1 score, the score in 1e-4 was the marginally better performing model, with a higher F1 score of 0.6 on the validation dataset, compared to 0.59 by the model with 1e-3.



Learning Rate	Epochs	Train F1 Score	Validation F1 Score
0.001	5	0.57	0.59
0.0001	**5**	**0.59**	**0.6**

5.2. Batch Size

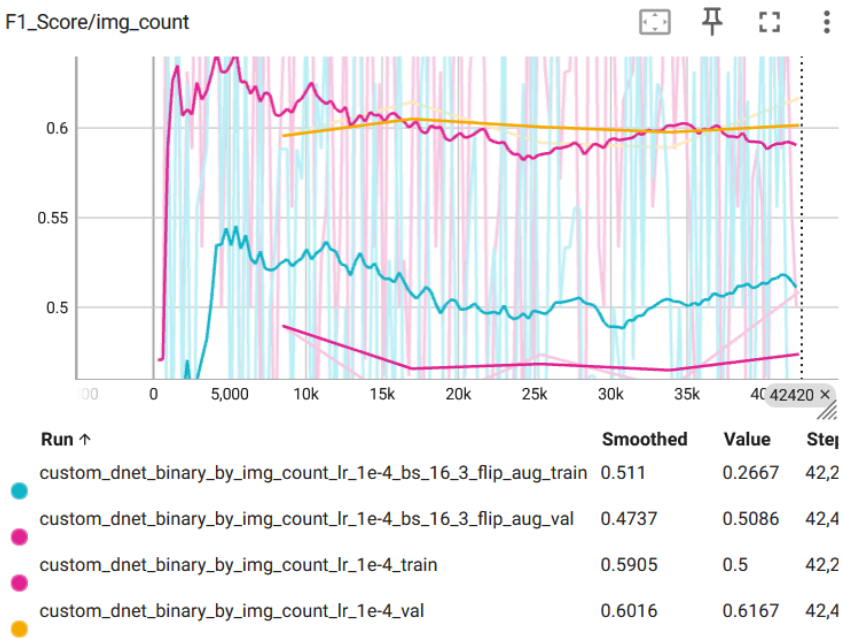
We experimented with batch sizes of 16 and 32. While both did not achieve a good F1 score, the score by model with batch size 16 was the marginally better performing model, with a higher F1 score of 0.6 on the validation dataset, compared to 0.57 by the model with batch size 32.



Batch Size	Epochs	Train F1 Score	Validation F1 Score
16	**5**	**0.59**	**0.6**
32	5	0.58	0.57

5.3. Augmentations

We attempted to implement Random Horizontal Flip as suggested in the original paper. While both did not achieve a good F1 score, the score by model with batch size 16 was the marginally better performing model, with a higher F1 score of 0.6 on the validation dataset, compared to 0.57 by the model with batch size 32.



With Flip?	Epochs	Train F1 Score	Validation F1 Score
Yes	5	0.511	0.47
No	**5**	**0.59**	**0.60**

Besides Random Horizontal Flip, it is difficult to find other augmentations due to the indistinct visibility of pneumothoraces on the X-Ray images, as shown below here during our Exploratory Data Analysis.

6. Results

During the testing of various modified densenets, we have arrived at the conclusion that the vanilla DenseNet-121 was still the best model architecture, and that the following hyperparameters were best suited for the task.

Learning Rate	Batch Size	Optimizer	Criterion	Epochs	Train F1 Score	Validation F1 Score	Test F1 Score
1e-4	16	Adam	BCELoss	50	0.61	0.60	0.59

7. Steps

7.1. Preparing Dataset

The function ***get_data_loaders*** is located in file ***preprocessing.py***. We imported this function into ***Trainer.ipynb.***
Run the first two cells in Trainer.ipynb to generate the training, validation and test dataset.

7.2. Exploratory Data Analysis

The data is then inspected in eda.ipynb, where we can look at examples of positive and negative cases, and look at the correlation between selected non-image.

7.3. Custom Model

The class ***dense_net*** is located in file ***custom_densenet.py.*** We imported this class into ***Trainer.ipynb*** in order to train the model***.***
Run the first four cells in ***Trainer.ipynb*** to view the model structure.

7.4. Training the model

Run all cells in ***Trainer.ipynb to train the model.*** Some training hyperparameters are exposed in the train helper function that is defined in the ***train_chexnet.py file.***

7.5. Evaluating Dataset

Run all cells in ***evaluation.ipynb*** to get the F1 score on the held-out test dataset, and view examples of some of the errors. (False Negative, False Positive)

8. Challenges faced and future Improvements

Due to time constraints, we encountered challenges while experimenting with transformers. Thus, in the future we would like to continue our attempts with transformers to improve the performance. Additionally, we plan to explore other architectures such as ResNet and EfficientNet.

We observed an unusually quick convergence on an average F1 score of 0.5 to 0.6 on each of the batches in the training dataset. Despite adjusting the learning rate and batch size, the model performance did not improve significantly. Therefore, we felt that the model's limiting factor was the model structure itself. Therefore, we allocated more time on refining the model, with the idea of tweaking the hyperparameters once a better performing model architecture was found.

9. References

[1] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., ... & Ng, A. Y. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*.

[2] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

[3] Zoogzog. (2018). chexnet. GitHub repository. *https://github.com/zoogzog/chexnet*