Singapore University of Technology and Design

60.001: Applied Deep Learning

Multi-Label Respiratory Disease Prediction

Abigail Tan	1005347
Tay Kay Wee	1005468
Mandis Loh Zhi Cheng	1005297

Table of Contents

Introduction

This paper introduces the application of deep learning techniques in medical imaging, focusing on the classification of thorax diseases from chest X-ray images. While chest X-ray technology is widely used for disease detection, the interpretation of these images by radiologists presents challenges due to the complexity of thoracic conditions and the scarcity of qualified professionals. Leveraging deep learning models offers promising potential to enhance diagnostic efficiency and accuracy. However, addressing the complexity of multi-label disease classification remains a significant hurdle. We aim to investigate the feasibility of lightweight deep learning models for multi-label disease detection, comparing state-of-the-art architectures like ResNet, AlexNet, and DenseNet with custom models. The findings from this study aim to contribute valuable insights to the development of improved diagnostic tools, ultimately advancing patient care in lung disease detection.

How to Run the Project

The project repository can be found at this link, and trained models can be found here.

Setup

- Ensure you have Python 3.9 or 3.10 installed. Create and activate your desired virtual environment.
- Install Python dependencies.
- We highly recommend that you have CUDA installed on your system. You can follow the instructions provided in the PyTorch documentation here.

Download Dataset

The dataset used in this project is the NIH Chest X-ray Dataset of 14 Common Thorax Disease Categories. To download the dataset, please run the ./src/download_data.py file. You will need about 42GB of space to download the entire dataset.

Run the Models

The notebooks used to run the models, as well as the definitions themselves are located in the /notebooks/ directory. The notebooks are split by the different models and are named as such:
AlexNet.ipynb contains our implementation of the state-of-the-art AlexNet model
AlexNet_variation.ipynb contains our implementation of our custom AlexNet model

All the notebooks contain both training and evaluation sections and parameters for each model can be adjusted within the notebooks themselves.

The notebooks are organized into three different types:

- Models Successfully Run:
    - AlexNet
    - AlexNet_variation
    - DenseNet_variation
    - ResNet
- Models Unable to Run:
    - DenseNet
    - ResNet_variation
    - VGG
- Data Exploration:
    - data_processing
    - exploration

Loading Trained Models

The models we trained can be found here. We have included the saved Pytorch model weights as well as the training and validation losses which we have presented in our report. To reproduce the results, please save the desired model to the ./models folder in the root directory and navigate to the corresponding model notebook in the ./notebooks directory. From there, run the testing code found in the notebook.

Literature Review

Deep learning, characterized by its multi-layered neural network architectures, has revolutionized image classification by automatically extracting hierarchical features from raw data. As it can learn intricate patterns and representations directly from the data, it has shown to yield better accuracy and performance in image classification tasks compared to traditional machine learning tasks (Yu, 2022).

In today's medical imaging, chest X-ray medical imaging technology is one of the most cost-effective and non-invasive approaches to detecting most thorax diseases. It is also the most accurate method of detection (Almezhghwi, Serte, Al-Turjman, 2021). However, while X-rays themselves are easy to execute, the interpretation of these chest X-rays presents various challenges.

Radiologists visually examine CXR images, a task that requires significant time and resources, particularly in regions facing a shortage of qualified medical professionals. Additionally, the low resolution of CXR images, combined with similarities in disease signs and potential lack of experience or focus during interpretation, poses challenges for radiologists, potentially resulting in life-threatening diagnostic errors (Ait Nasser & Akhloufi, 2023).

Given the importance of early and accurate detection in improving patient outcomes, there is growing recognition of the potential for artificial intelligence (AI) to help radiologists make the diagnosis process more efficient by reducing human error and effort. Specifically, researchers have turned to deep learning models as a means to enhance the classification of chest X-ray images.

Many current studies on the use of deep learning models to identify thorax diseases using X-ray images have indicated great performance in binary classifications, particularly in the detection of pneumonia and Covid-19 (Malik et al., 2023). However, given that patients may have multiple thoracic diseases concurrently, the capacity of deep learning models to detect more than one disease per X-ray image—referred to as Multi-Label Classification—is pivotal in disease identification from X-ray images. Multi-label classification proves more challenging than single-label classification because it addresses the problem of imbalanced numbers of positive and negative labels while also extracting multiple object features from a single sample (Jin et al., 2023).

Our project aims to contribute to the advancement of medical imaging practices by testing if a lightweight model for multi-label classification of disease detection for X-ray images is feasible. We will be testing state-of-the-art deep learning models against our own custom, lightweight models inspired by these state-of-the-art models. Through this exploration, we hope to provide insights that can potentially guide the development of improved diagnostic tools, ultimately enhancing patient care in the field of lung disease detection.

Methodology

Our dataset is taken from the United States National Institutes of Health (NIH). It is a chest X-ray dataset of 14 common thorax disease categories, with a total of 112,120 frontal-view X-ray images of 1024 by 1024 resolution, with text-mined fourteen disease image labels.

We downloaded the data from NIH using a script due to the large size of the dataset and did some data exploration to gain insights on how best to process it. We then explored several state-of-the-art models to use as a baseline for comparison against our own models. This was done by training and evaluating each model, as well as carrying out parameter fine-tuning.

Data Acquisition

A diverse dataset comprising thoracic imaging scans, including X-rays and CT scans, was acquired from multiple medical institutions with appropriate ethical approvals. The dataset was meticulously curated to ensure a representative distribution of thoracic diseases, including but not limited to pneumonia, tuberculosis, and lung cancer. Preprocessing techniques such as normalization and augmentation were applied to enhance the robustness and generalization of the models.

The following data was provided to us on the NIH website:

1. Images: We had to first download the dataset from the NIH website. We used a python script to download and extract the images locally. The images were named in the following format:" {patient id}_{follow up number}".

2. Data Entry (shown in figure 1): CSV file containing the Image index, Disease labels, Patient ID, Follow up IDs, Gender of Patient, Image Size

	Image Index	Finding Labels	Follow-up #	Patient ID	Patient Age	Patient Gender	View Position	OriginalImage[Width	Height]	OriginalImagePixelSpacing[x	y]
0	00000001_000.png	Cardiomegaly	0	1	57	M	PA	2682	2749	0.143	0.143
1	00000001_001.png	Cardiomegaly|Emphysema	1	1	58	M	PA	2894	2729	0.143	0.143
2	00000001_002.png	Cardiomegaly|Effusion	2	1	58	M	PA	2500	2048	0.168	0.168
3	00000002_000.png	No Finding	0	2	80	M	PA	2500	2048	0.171	0.171
4	00000003_001.png	Hernia	0	3	74	F	PA	2500	2048	0.168	0.168
...	...	...	...	...	...	...	...	...	...	...	...
112115	00030801_001.png	Mass|Pneumonia	1	30801	38	M	PA	2048	2500	0.168	0.168
112116	00030802_000.png	No Finding	0	30802	28	M	PA	2048	2500	0.168	0.168
112117	00030803_000.png	No Finding	0	30803	42	F	PA	2048	2500	0.168	0.168
112118	00030804_000.png	No Finding	0	30804	29	F	PA	2048	2500	0.168	0.168
112119	00030805_000.png	No Finding	0	30805	26	M	PA	2048	2500	0.171	0.171

112120 rows × 11 columns

Figure 1: Data Entry CSV file

3. Test_list: Text file containing the image index for the test dataset
4. Train_val_list: Text file containing the image index for the training dataset
5. BBox List: CSV file containing the Image Index, the disease label, and coordinates of the bounding box

Data Exploration

In order to better understand the data, we did some data exploration. We first plotted a graph to see the distribution of labels across the dataset. As shown in figure 2 below, Hernia has the least occurrence, while Infiltration has the highest occurrence. Our exploration revealed a mere 227 instances of Hernia within the extensive dataset, comprising a total of 112,120 cases.

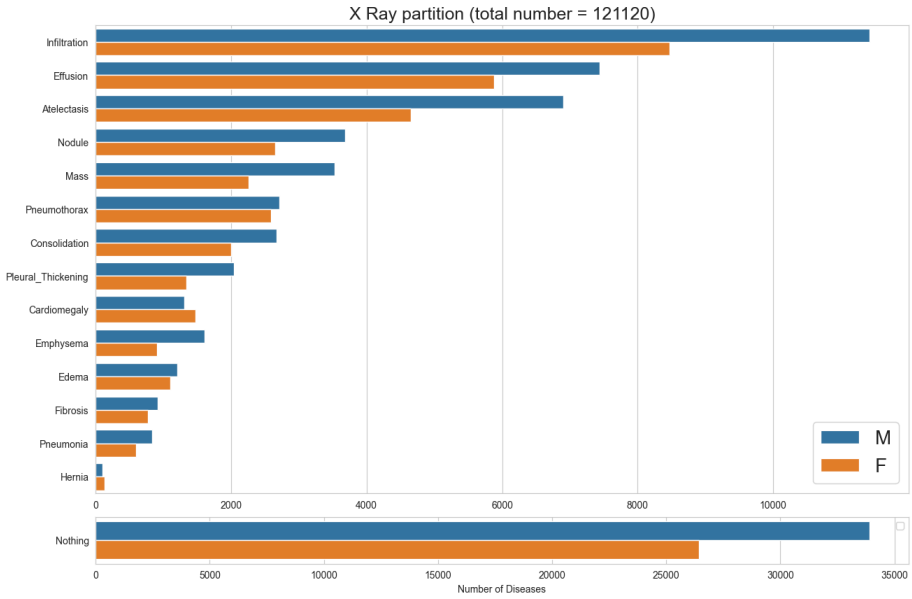


Figure 2: Distribution of Disease Classes in the Dataset

We discerned that the task at hand constitutes a multi-label classification challenge, wherein each chest imaging instance may fall under multiple categories. We conducted an analysis, shown in figure 3, to gain insights into the distribution of single and multiple pathology distribution within each class. Our findings show that the "No Findings" class predominantly comprises the dataset provided, uniquely belonging to a singular class.

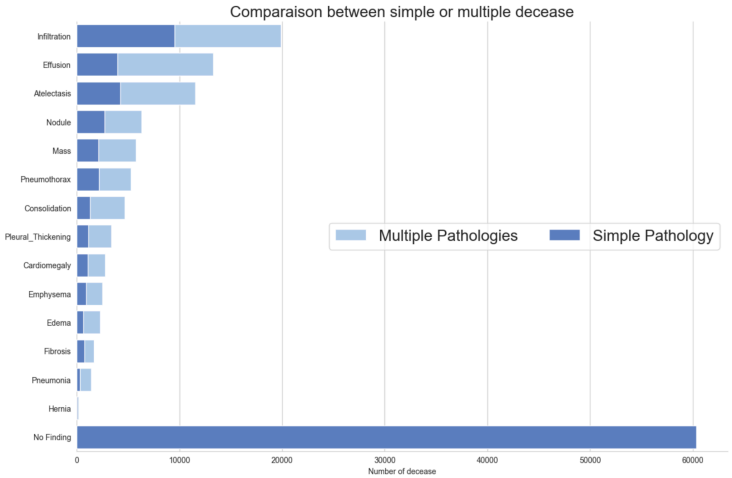


Figure 3: Distribution of Pathologies within Each Class in the Dataset

Data Processing

The data processing pipeline encompasses a series of crucial steps aimed at transforming raw medical imaging data into a structured format conducive to ML model training and evaluation. These steps not only involve consolidating and formatting the image data but also entail crucial preprocessing and labeling tasks essential for effective model training. Here are the steps we took to process the data

1. Consolidate all information by adding image data to dataframe

We organized the image data into a structured format using a Pandas DataFrame. This lets us easily access and manage information about each image. Each row in the DataFrame represents a specific image, identified by its index. We included relevant columns like patient details and diagnosed diseases for each image. Additionally, we added a "path" column containing the relative location of the image file within the dataset directory. This allows us to quickly load the images for further analysis.

To build this DataFrame, we focused on the image files and the "Data Entry" CSV files.
We excluded the "text_val_list" and "test_val_list" text files because they predefined training and test data splits based on the image index. Our goal was to have more control over how the data was divided and avoid relying on predefined configurations. Additionally, there was no separate validation list provided, hence using the text files would have excluded validation data altogether. We also excluded the BBox List as it contained coordinates of the bounding boxes which are mainly used for object detection and are not relevant to our current multi-label classification task.

2. Downsampling dataset

As the dataset was large, it required significant computational resources to train deep learning models. Downsampling was thus conducted to reduce the data size, making the training process faster and less resource-intensive. Duplicate entries were removed to ensure each data point is counted only once and avoid biasing the model towards certain data points.

For each unique disease, a column was added to indicate the presence of the disease. These columns encode the presence or absence of each disease for each image. A value of 1 indicates the presence, and 0 indicates absence. This allows us to iterate through the data to select 300 samples per disease. We included only 300 samples as the training deep learning models require significant computational resources. Using a smaller dataset can potentially reduce training time and memory usage, making the training process more efficient. In addition, this would help create a more balanced dataset. This is important because deep learning models can struggle with imbalanced datasets where some diseases are much more frequent than others. By ensuring a similar number of samples for each disease, the model gets a better chance to learn effectively for all types of diseases.

3. Reformatting the data in the ‘Finding Labels’ columns

The labels in the 'Finding Labels' column were stored in a single string format, with each label separated by a delimiter like '|'. This format might be difficult for a machine learning model to understand and process directly. Thus, we needed to split the labels by the delimiter ('|') to create a separate entry for each disease or finding within the label. This reformatting transforms the data into a more usable format, typically a list or array.

4. Splitting Data into Training, Validation, and Test Dataset
    a. Used the train_test_split function
    b. Stratified based on 'Finding Labels' Column

We then split the dataset into training data, test data, and validation data, where 20% of the data is for testing, 64% for training, and 16% for validating. The majority of the data (64%) is allocated for training the model as it is the data used by the model to learn.

A significant portion (20%) is reserved for testing to evaluate the model’s performance. This unseen data would assess its ability to generalize and make accurate predictions on new cases. The remaining 16% is used for validation. This serves as an intermediate step during training. The model is evaluated on the validation set to monitor its progress and identify potential issues like overfitting before final evaluation on the test set.

The data has imbalanced classes, where some diseases like "Hernia" are much less frequent than others. Thus, we decided to apply stratified splitting to ensure the training, validation, and test sets maintain the same proportion of each disease category as the original dataset.
This prevents biased predictions, where some diseases are heavily overrepresented. In addition, it ensures the model is exposed to a balanced representation of all diseases during training and testing. This leads to better generalization (performing well on unseen data) and a more reliable evaluation of the model's performance across different disease types.

5. Creation of DataSet object
    a. Inputs: image file path, list of labels
    b. Output: Image tensor, multi-hot encoded tensor

We initialized 3 CustomDataset objects that handle loading and preprocessing image data along with its corresponding labels for training, validation, and testing. The DataSet object takes a list of labels as input and converts them into a multi-hot encoded tensor. Multi-hot encoding is a technique used for representing labels in multi-label classification problems. Each label has a corresponding position in the tensor, and a value of 1 indicates the presence of that label, while 0 signifies its absence. Pillow library was also used to read the image data from the provided file path as the output of Pillow's image loading functions can be easily converted into PyTorch tensors. The image is then converted to a tensor and normalized before being outputted

6. Load data with DataLoader object

Lastly, we loaded the data with the DataLoader object to streamline data loading and minimize I/O operations, improving training speed. Also, It automatically groups data samples into batches, simplifying the training loop logic. This helps us to manage memory usage by loading data in batches instead of loading the entire dataset at once. For this project, a batch size of 16 was used. While we had explored a larger batch size of 32 which can potentially lead to faster training due to better hardware utilization, it was not possible due to the increased memory needed.

Final Dataset Size

Total size: 4427 samples
Train dataset: 2832 samples
Validation dataset: 709 samples
Test dataset: 886 samples

Architecture of Models

We explored various state-of-the-art models and used these models as inspiration to create our own smaller, less computationally expensive variations. Through this experimentation process which will be covered in the Experiments section, we narrowed our scope down to focus on the three models covered below.

Model Inputs and outputs

For each sample, our model takes in an image tenor and output an array of length 15, where every element is binary and represents the following classes:

1. Atelectasis
2. Cardiomegaly
3. Consolidation
4. Edema
5. Effusion
6. Emphysema
7. Fibrosis
8. Hernia
9. Infiltration
10. Mass
11. No Finding
12. Nodule
13. Pleural Thickening
14. Pneumonia
15. Pneumothorax

While our image size is 1024 x 1024, Pytorch's implementation of the AlexNet and ResNet model does the following preprocessing on our image:

1. The images are resized to 256 x 256
2. A central crop of 224 x 224 is applied.
3. The image values are first rescaled to [0, 1] and normalized with a mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225]

The following section only covers the architecture of our top 3 models. Other models we tried that were unsuccessful can be found in Appendix A.

State-of-the-Art Models

ResNet18

ResNet18 is a 72 layer architecture with 18 deep layers which makes use of skip connections to resolve the issue of vanishing gradient faced by deep networks (*Residual Neural Network - an Overview* | *ScienceDirect Topics*, n.d.). Figure 4 below shows the original ResNet18 architecture.

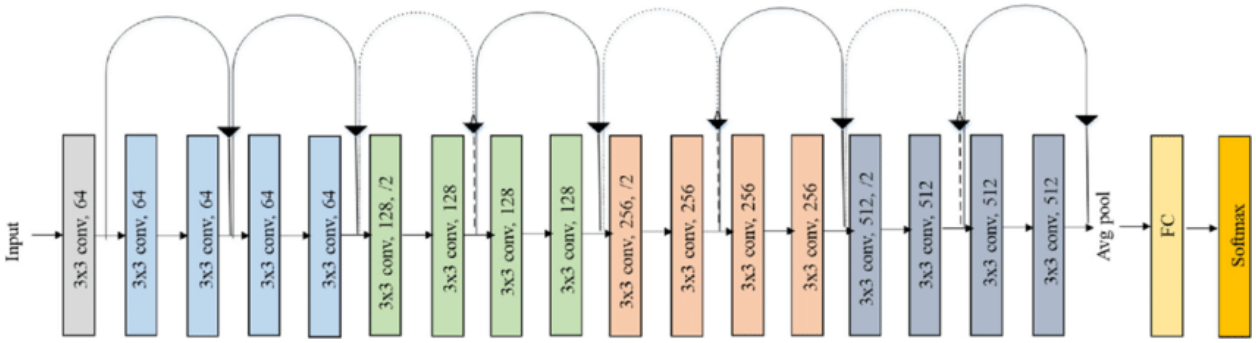


Figure 4: Architecture of original ResNet18 (Iqbal, 2019)

In our implementation in PyTorch, we kept the original architecture's input size of 3 x 224 x 224 pixels and changed the final connected layer to output 15 classes. Pytorch's implementation of ResNet18 does not come with the final SoftMax layer, hence we have replaced the final layer with a sigmoid layer to obtain the probabilities of the sample belonging to each class.

AlexNet

The network consists of 8 layers with weights, of which the first five are convolutional layers and the last three are fully connected. The output of the last fully connected layer is given to a SoftMax function (Krizhevsky et al., 2012). Figure 5 below shows the original AlexNet architecture.

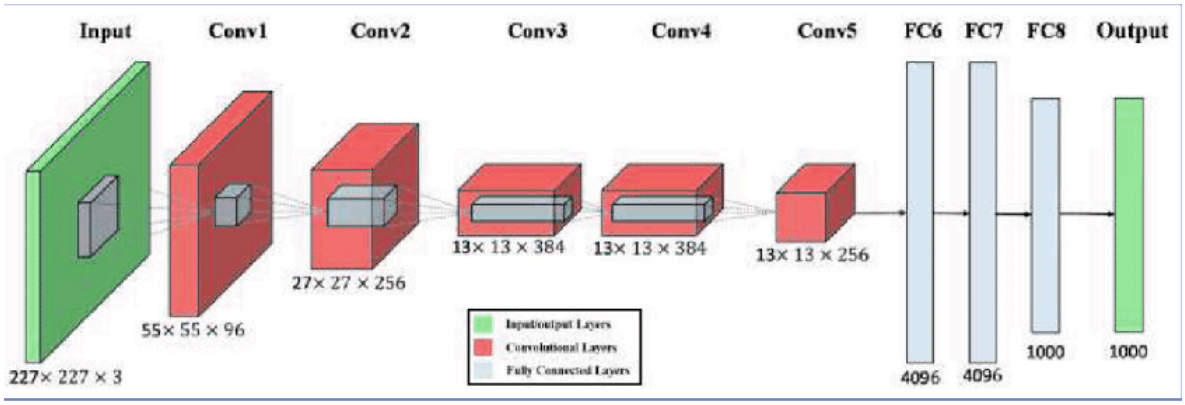


Figure 5: Architecture of original AlexNet (Putzu et al., 2020)

In our implementation, we kept the original architecture's input size of 3 x 224 x 224 and changed the final connected layer to output 15 classes. Pytorch's implementation of AlexNet does not come with the final SoftMax layer, hence we have replaced the final layer with a sigmoid layer to obtain the probabilities of the sample belonging to each class.

Custom Models

AlexNet Variation

We first initialized the weights randomly and followed the basic architecture of the original AlexNet model with five convolutional layers and three fully connected layers. As batch normalization is typically used to help stabilize and speed up the training process by normalizing the activations of each layer, we wanted to experiment if it would be useful for our project hence we added batch normalization layers after each convolutional layer. Figure 6 shows the architecture of our custom AlexNet model.

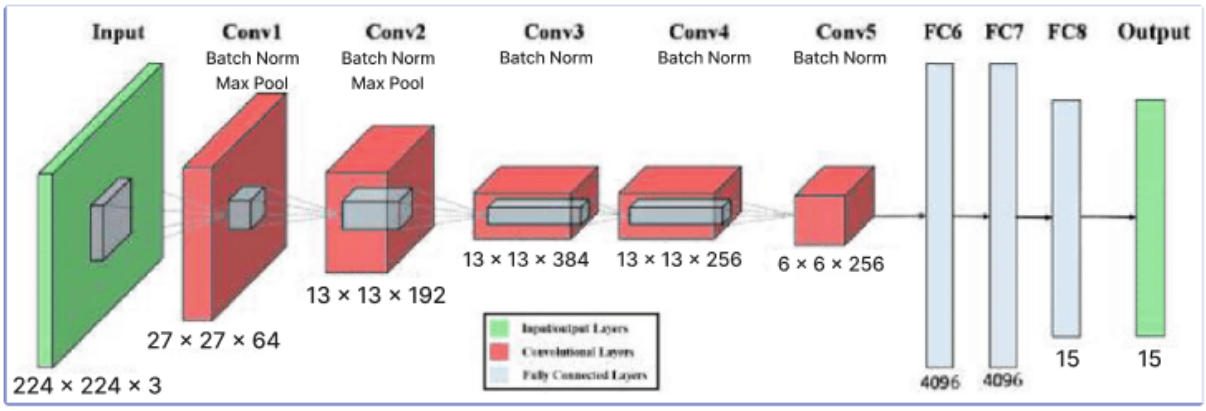


Figure 6: Architecture of the custom AlexNet

Training the Models

For the state-of-the-art models, we used transfer learning by freezing all layers and adjusting the weights of the final fully connected layer. All the models (state-of-the-art and our custom models) were trained using the same training function and parameters. We used the **Adam optimiser** as the default optimiser as it is highly efficient and converges quickly. We also used **Binary Cross-Entropy (BCE) loss** across all models and for every model, we experimented with two learning rates: 0.001 and 0.0005.

For every epoch, our training function trains the model and outputs the average BCE loss across the dataset loaded into the function. We also evaluate the BCE loss within the epoch on our validation dataset to track how well the model performs on unseen datasets. We track both the training and validation losses for every epoch and save the model with the lowest loss on the validation dataset. The detailed training and validation loss of each model can be found in Appendix B.

Insights

From the loss graphs below which show the loss value of the 3 models we explored across 15 epochs, the loss value for the graph with a learning rate of 0.0005 is lower than graphs with 0.001 across all the models when the learning rate is less than 0.0005.

The validation loss curve for Resnet is decreasing more than its training loss curve, while the validation loss curve for AlexNet is decreasing at a similar rate as the training loss curve. This suggests that both the models are not overfitting. As for the custom AlexNet, the validation and training loss curves for both learning rates appear to flatten out after 7 epochs, which suggests that the model is no longer learning effectively thereafter.

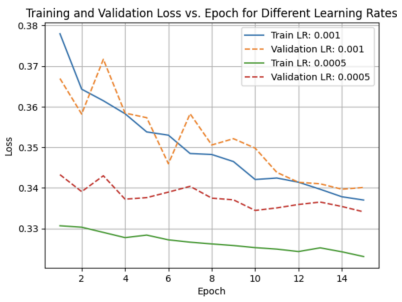


Figure 7: ResNet Loss

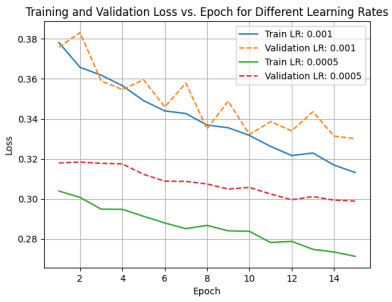


Figure 8: AlexNet Loss

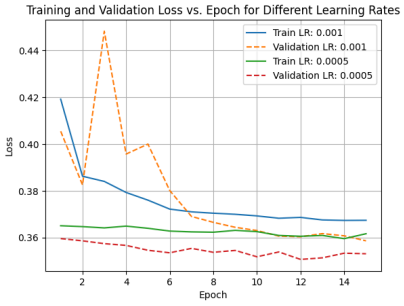


Fig 9: Custom AlexNet Loss

AlexNet (Figure 8) appears to have the lowest overall validation loss, followed by Custom AlexNet (Figure 9) and then ResNet (Figure 7). This suggests that the AlexNet architecture may have performed better on this task. As such, we decided to run the AlexNet on 30 epochs to see how the validation loss changes.

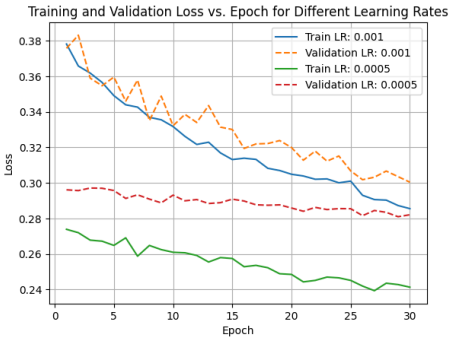


Figure 10: AlexNet Loss with 30 epochs

The validation loss for AlexNet across 30 epochs continues to drop steadily after the first 15 epochs (Figure 10), resulting in a lower validation loss of about 0.29 for the learning rate of 0.001.

We also decided to run the Custom AlexNet on 30 epochs to compare the Custom AlexNet with AlexNet model. Unsurprisingly, the AlexNet outperformed the custom AlexNet on both learning rates. However, despite being outperformed by the AlexNet model, our Custom AlexNet with a learning rate of 0.005 displayed significant learning progress. The loss value for this model decreased from an initial value of 0.39 to approximately 0.315 (Figure 11). This suggests that our Custom AlexNet architecture, when trained with a specific learning rate, exhibits a capacity for effective learning, even surpassing the performance of the widely recognized AlexNet architecture in terms of loss reduction. Further investigation into the underlying reasons for this observed behavior could yield valuable insights into the effectiveness of custom model architectures in specific learning scenarios.

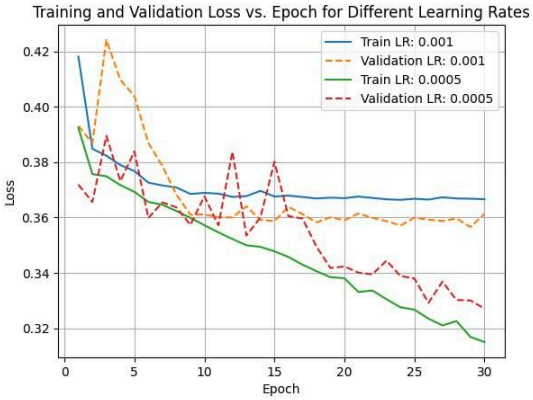


Figure 11: Custom AlexNet Loss with 30 epochs

Results

Metrics

The key evaluation metrics used in our study include accuracy, precision, recall, and F1 score.

Accuracy was chosen as a metric as it is a fundamental metric for assessing model performance, indicating how well the model predicts both positive and negative instances of respiratory diseases. As each image can belong to a different class, our output is an array of length 15 to represent different classes.

Precision is crucial in the context of respiratory disease prediction because it assesses the model's ability to minimize false positives. In healthcare applications, such as respiratory disease prediction, false positives can lead to unnecessary treatments or interventions, causing patient anxiety and additional healthcare costs.

Recall, also known as sensitivity, is particularly relevant in respiratory disease prediction as it measures the model's ability to identify all actual positive cases correctly. Missing positive cases (false negatives) can have severe consequences in healthcare, as it may lead to undiagnosed or untreated respiratory diseases, potentially compromising patient health outcomes.

The F1 score combines precision and recall into a single metric, providing a balanced assessment of the model's performance. In the context of respiratory disease prediction, where both false positives and false negatives have significant implications, the F1 score offers a comprehensive evaluation of the model's effectiveness. It is particularly useful when there is an imbalance between positive and negative instances or when the costs associated with false positives and false negatives are not equal. By considering both precision and recall, the F1 score provides a more nuanced understanding of the model's predictive capabilities, enabling better decision-making in clinical settings.

We calculated the metrics element-wise, and output the 4 metrics for each of the classes for each of the models. An example of the evaluation results from one of our models can be found in Table 1.

Model	Class	Metrics				
		Precision	**Accuracy**	**Recall**	**F1**	**Loss**
Alexnet Lr 0.001	Atelectasis	0.769231	0.827314	0.11976	0.207254	0.33811
	Cardiomegaly	0.733333	0.891648	0.106796	0.186441	
	Consolidation	1	0.88149	0.027778	0.054054	
	Edema	0.769231	0.924379	0.421053	0.544218	
	Effusion	0.916667	0.774266	0.052381	0.099099	
	Emphysema	0.736842	0.927765	0.191781	0.304348	
	Fibrosis	0.583333	0.900677	0.077778	0.137255	
	Hernia	0.5	0.946953	0.021277	0.040816	
	Infiltration	0.724138	0.727991	0.082677	0.14841	
	Mass	1	0.892777	0.030612	0.059406	
	No Finding	0	0.93228	0	0	
	Nodule	1	0.882619	0.009524	0.018868	
	Pleural Thickening	1	0.882619	0.028037	0.054545	
	Pneumonia	1	0.897291	0.01087	0.021505	
	Pneumothorax	0.761905	0.901806	0.163265	0.268908	

Table 1: Evaluation Metrics for AlexNet (Epoch=15, LR=0.001)

Due to the nature of our problem, we have identified recall as the most important metric out of the four metrics and will be focusing on it in the Results section. The full set of metrics for each of the models we have explored can be found in Appendix C.

Discussion

Firstly, we observed that our models were unable to predict the “No Findings” class. Essentially, this means that a chest X-ray scan indicating a healthy condition might be misinterpreted as showing diseased lungs. This difficulty can be attributed to the nature of our model, which is designed to handle multilabel classification tasks. This results in the model learning to recognize specific features or patterns in the data that correspond to multiple labels simultaneously during training. However, when it comes to cases like the "No Findings" class, where the features are associated exclusively with a singular class, the model may struggle to differentiate them and might incorrectly predict the sample as multiple labels instead of accurately identifying the singular class. Since our models were unable to extract meaningful learnings from the “No Findings” class, we will exclude it from our discussion of the results.

From Table 2 below, we can see that ResNet18 generally does not perform well on our dataset. AlexNet has a higher average recall on both learning rates when compared to the respective ResNet model. Additionally, ResNet is unable to make any meaningful predictions on 5 out of 14 classes even on the better performing learning rate of 0.0005. AlexNet, on the other hand, was able to make meaningful predictions on all 14 classes.

Additionally, we can observe that the model with a lower learning rate of 0.0005 outperforms the same model with a learning rate of 0.001. For example, in Table 1, the AlexNet with a learning rate of 0.0005 outperforms the AlexNet with a learning rate of 0.001 on 12 out of the 14 classes.

Class	**ResNet (15 Epoch)**		**AlexNet (15 Epoch)**	
	LR = 0.0005	LR = 0.001	LR = 0.0005	LR = 0.001
Atelectasis	0.023952	0.24551	0.137725	0.11976
Cardiomegaly	0.019417	0	0.446602	0.106796
Consolidation	0	0	0.083333	0.027778
Edema	0.484211	0.06316	0.347368	0.421053
Effusion	0.080952	0.04286	0.27619	0.052381
Emphysema	0	0	0.315068	0.191781
Fibrosis	0.011111	0.01111	0.122222	0.077778
Hernia	0	0	0.021277	0.021277
Infiltration	0.15748	0.07087	0.114173	0.082677

Mass	0	0	0.061224	0.030612
Nodule	0.009524	0.00952	0.07619	0.009524
Pleural Thickening	0	0	0.074766	0.028037
Pneumonia	0.01087	0.01087	0.054348	0.01087
Pneumothorax	0.102041	0.030612	0.27551	0.163265
Average Recall	**0.064**	**0.035**	**0.172**	**0.096**

Table 2: ResNet vs AlexNet Recall Results on 15 epochs

Since AlexNet performed better on our dataset than ResNet, we evaluated our custom AlexNet model against the state-of-the-art AlexNet. We initially ran our custom AlexNet on 15 epochs, but it was unable to learn any meaningful patterns in the data. Running our custom AlexNet on 30 epochs revealed that the higher learning rate of 0.001 was unable to extract any meaningful patterns from our data as well. It is hence excluded from Table 2.

The results for AlexNet against our custom AlexNet are summarized in Table 2. The detailed results can be found in Appendix C.

When compared against the 30 epoch AlexNet models, our custom AlexNet model falls short in the average recall. Comparing our custom AlexNet to the AlexNet with the same epoch and learning rate, our custom AlexNet is unable to outperform the AlexNet on any of the classes.

Class	**AlexNet (30 Epoch)**		**Custom AlexNet (30 Epoch)**
	LR = 0.0005	LR = 0.001	LR = 0.0005
Atelectasis	0.305389	0.131737	0.101796
Cardiomegaly	0.417476	0.271845	0.126214
Consolidation	0.175926	0.074074	0.083333
Edema	0.463158	0.326316	0.2
Effusion	0.42381	0.114286	0.214286
Emphysema	0.369863	0.260274	0.09589

Fibrosis	0.111111	0.044444	0.022222
Hernia	0.06383	0.021277	0
Infiltration	0.220472	0.102362	0.086614
Mass	0.153061	0.061224	0.091837
Nodule	0.114286	0.057143	0.038095
Pleural Thickening	0.102804	0.037383	0.056075
Pneumonia	0.108696	0.054348	0.01087
Pneumothorax	0.295918	0.285714	0.234694
Average Recall	**0.238**	**0.132**	**0.097**

Table 3: AlexNet vs custom AlexNet (30 epochs)

However, our custom AlexNet was able to outperform some of the models. The comparison between the custom AlexNet and some models it outperformed is shown in Table 4.

Our custom AlexNet outperforms the ResNet tested, and has a similar performance to the AlexNet run on 15 epochs in terms of average recall. Our custom model outperforms ResNet on 10 out of 14 classes and 8 out of 14 classes for AlexNet.

Class	**Custom AlexNet (30 Epoch)**	**ResNet (15 Epoch)**	**AlexNet (15 Epoch)**
	LR = 0.0005	LR = 0.0005	LR = 0.001
Atelectasis	0.101796	0.023952	0.11976
Cardiomegaly	0.126214	0.019417	0.106796
Consolidation	0.083333	0	0.027778
Edema	0.2	0.484211	0.421053
Effusion	0.214286	0.080952	0.052381

Emphysema	0.09589	0	0.191781
Fibrosis	0.022222	0.011111	0.077778
Hernia	0	0	0.021277
Infiltration	0.086614	0.15748	0.082677
Mass	0.091837	0	0.030612
Nodule	0.038095	0.009524	0.009524
Pleural Thickening	0.056075	0	0.028037
Pneumonia	0.01087	0.01087	0.01087
Pneumothorax	0.234694	0.102041	0.163265
Average Recall	**0.097**	**0.064**	**0.096**

Table 4: Custom AlexNet vs Models it Outperformed

Conclusion

The aim of our project was to determine if the use of lightweight models, particularly those inspired by current state-of-the-art models, would be feasible for multi-label classification of disease detection for X-ray images. From our results above, we have come to the conclusion that it is possible for lightweight models to be used for this problem. However, they would need to be fine-tuned for the specific objective of the problem and would require extensive parameter optimisation for better performance. Additionally, from our findings, we recommended that they are trained on lower learning rates on a higher number of epochs.

Future Work

In this project, we were only able to explore ResNet and AlexNet effectively due to time constraints as well as a lack of computational power. However, studies in the field of chest X-ray analysis often include other state-of-the-art model architectures such as Very Deep Convolutional Networks (VGG) and DenseNet (Meedeniya et al., 2022). Hence, a future improvement for this project would be to test these other models to determine which model has the best performance on our multi-label classification problem.

From there, another future work would be to create other custom models based on these state-of-the-art models, and rigorously train them. From our findings, we would suggest training custom models on more epochs and a smaller learning rate, but experimentation will be needed to achieve the best model. A potential avenue for exploration for the custom models would be to build ensemble models based on the best custom models as well.

Lastly, our project deals with the metrics by calculating them element-wise. This was done due to the complexity of the problem exponentially increasing if we were to compare them output-wise. However, our work acts as a stepping stone for output-wise comparison for metrics, and further improvements could produce models that are able to deal with the complexity of the output-wise comparison.

References

Ait Nasser, A., & Akhloufi, M. A. (2023). A Review of Recent Advances in Deep Learning Models for Chest Disease Detection Using Radiography. *Diagnostics*, *13*(1), 159. https://doi.org/10.3390/diagnostics13010159

Almezhghwi, K., Serte, S., & Al-Turjman, F. (2021, June 17). Convolutional neural networks for the classification of chest X-rays in the IoT era. National Library of Medicine. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8210525/

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 (CVPR). Retrieved from https://doi.org/10.48550/arXiv.1608.06993

Iqbal, S. (2019, December). *Fig. 2 Original ResNet-18 Architecture*. ResearchGate. https://www.researchgate.net/figure/Original-ResNet-18-Architecture_fig1_336642248

Jin, Y., Lu, H., Zhu, W., & Huo, W. (2023). Deep learning based classification of multi-label chest X-ray images via dual-weighted metric loss. *Computers in Biology and Medicine*, *157*, 106683. https://doi.org/10.1016/j.compbiomed.2023.106683

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Malik, H., Tayyaba Anees, Ahmad Sami Al-Shamaylehs, Alharthi, S. Z., Khalil, W., & Adnan Akhunzada. (2023). Deep Learning-Based Classification of Chest Diseases Using X-rays, CT Scans, and Cough Sound Images. *Diagnostics (Basel)*, *13*(17), 2772–2772. https://doi.org/10.3390/diagnostics13172772

Meedeniya, D., Kumarasinghe, H., Kolonne, S., Fernando, C., Díez, I. D. la T., & Marques, G. (2022). Chest X-ray analysis empowered with deep learning: A systematic review. Applied Soft Computing, 126, 109319. https://doi.org/10.1016/j.asoc.2022.109319

Putzu, L., Piras, L., & Giacinto, G. (2020). Convolutional neural networks for relevance feedback in content based image retrieval. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-020-09292-9

Residual Neural Network - an overview | ScienceDirect Topics. (n.d.). Www.sciencedirect.com. Retrieved April 17, 2024, from https://www.sciencedirect.com/topics/computer-science/residual-neural-network#:~:text=ResNet18%20is%20a%2072%2Dlayer

Yuan, C., Wu, Y., Qin, X., Qiao, S., Pan, Y., Huang, P., Liu, D., & Han, N. (2019). An effective image classification method for shallow densely connected convolution networks through squeezing and splitting techniques. *Applied Intelligence*, *49*, doi:10.1007/s10489-019-01468-7.
Yu, Y. (2022). Deep Learning Approaches for Image Classification. *Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering (EITCE 2022)*. Xiamen, China. https://doi.org/10.1145/3573428.3573691

Harders, S. W., Madsen, H. H., Hjorthaug, K., Arveschoug, A. K., Rasmussen, T. R., Meldgaard, P., Andersen, J. B., Pilegaard, H. K., Hager, H., Rehling, M., & Rasmussen, F. (2012, October 16). Characterization of pulmonary lesions in patients with suspected lung cancer: Computed tomography versus [18F] fluorodeoxyglucose-positron emission tomography/computed tomography. Cancer imaging : the official publication of the International Cancer Imaging Society. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3478790/

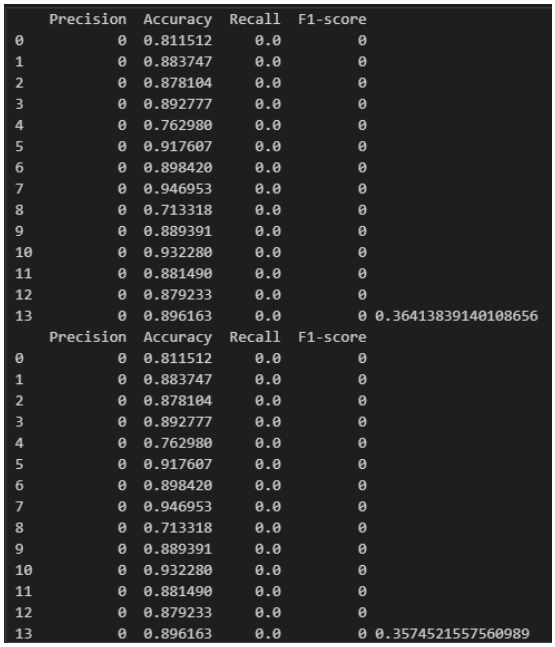
Appendix

Appendix A: Failed Models

Models with Evaluation

AlexNet variation with BCELossLogits

After experimenting with the AlexNet variation, we found out that it is not ideal to train with lightweight models as they might lack the capacity to capture complex patterns of the chest X-ray images. As shown in Figure A1, we got a score of ‘0’ for all the evaluation metrics excluding accuracy.



	Precision	Accuracy	Recall	F1-score	
0	0	0.811512	0.0	0	
1	0	0.883747	0.0	0	
2	0	0.878104	0.0	0	
3	0	0.892777	0.0	0	
4	0	0.762980	0.0	0	
5	0	0.917607	0.0	0	
6	0	0.898420	0.0	0	
7	0	0.946953	0.0	0	
8	0	0.713318	0.0	0	
9	0	0.889391	0.0	0	
10	0	0.932280	0.0	0	
11	0	0.881490	0.0	0	
12	0	0.879233	0.0	0	
13	0	0.896163	0.0	0	0.36413839140108656
	Precision	Accuracy	Recall	F1-score	
0	0	0.811512	0.0	0	
1	0	0.883747	0.0	0	
2	0	0.878104	0.0	0	
3	0	0.892777	0.0	0	
4	0	0.762980	0.0	0	
5	0	0.917607	0.0	0	
6	0	0.898420	0.0	0	
7	0	0.946953	0.0	0	
8	0	0.713318	0.0	0	
9	0	0.889391	0.0	0	
10	0	0.932280	0.0	0	
11	0	0.881490	0.0	0	
12	0	0.879233	0.0	0	
13	0	0.896163	0.0	0	0.3574521557560989

Figure A1: Evaluation Metrics of custom AlexNet

Lightweight ResNet variation

As the ResNet was relatively computationally heavy, we wanted to create a more lightweight model that could run on most computers. We thus created our own variation of the ResNet model, which had only 6 layers, but still made use of the skip connections that were introduced in ResNet. However, it was unable to extract the information and patterns of the chest. As shown in Figure A2, we got a score of '0' for all the evaluation metrics excluding accuracy.

	Precision	Accuracy	Recall	F1-score	
0	0	0.811512	0.0	0	
1	0	0.883747	0.0	0	
2	0	0.878104	0.0	0	
3	0	0.892777	0.0	0	
4	0	0.762980	0.0	0	
5	0	0.917607	0.0	0	
6	0	0.898420	0.0	0	
7	0	0.946953	0.0	0	
8	0	0.713318	0.0	0	
9	0	0.889391	0.0	0	
10	0	0.932280	0.0	0	
11	0	0.881490	0.0	0	
12	0	0.879233	0.0	0	
13	0	0.896163	0.0	0	0.3672283600483622

Figure A2: Evaluation Results for custom ResNet

Models we considered

VGG

Even though VGG is widely used, we were unable to run pre-trained VGG16 and VGG19 with our computers due to their computational complexity, making them computationally heavy and memory-intensive.

DenseNet

DenseNet utilizes DenseBlocks which creates dense connections between the different layers (Huang et al., 2017). However, due to a lack of computation resources, we could not run the DenseNet model on our data as shown in Figure A3.

```
49      bottleneck_output = self.conv1(self.relu1(self.norm1(concated_features)))  # noqa: T484
50      return bottleneck_output

OutOfMemoryError: CUDA out of memory. Tried to allocate 96.00 MiB (GPU 0; 23.99 GiB total capacity; 37.53 GiB already allocated; 0 bytes free; 3
```

Figure A3: Out of memory error

Lightweight DenseNet Variation

Since the state-of-the-art DenseNet was too computationally heavy, we decided to create a variation of DenseNet that also makes use of these Dense Blocks, but contains a smaller number of layers as shown in Figure A4. The Custom DenseNet model contains 9 layers, which is significantly less than the smallest DenseNet model which has 121 layers. However, due to a lack of computation resources, we could not run the lightweight DenseNet variation as well.

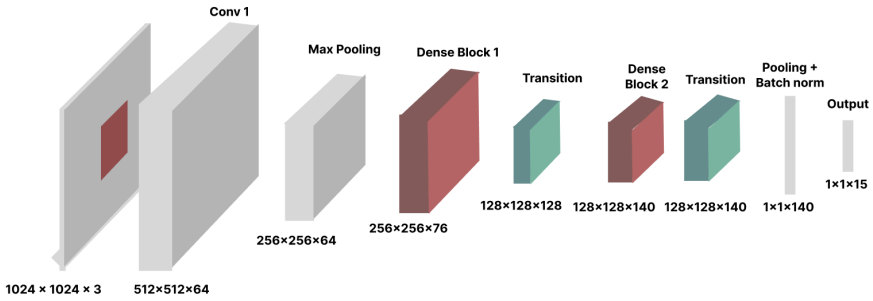


Figure A4: Architecture of the custom DenseNet

Appendix B: Test and Validation Loss Functions

ResNet18 (15 epochs)

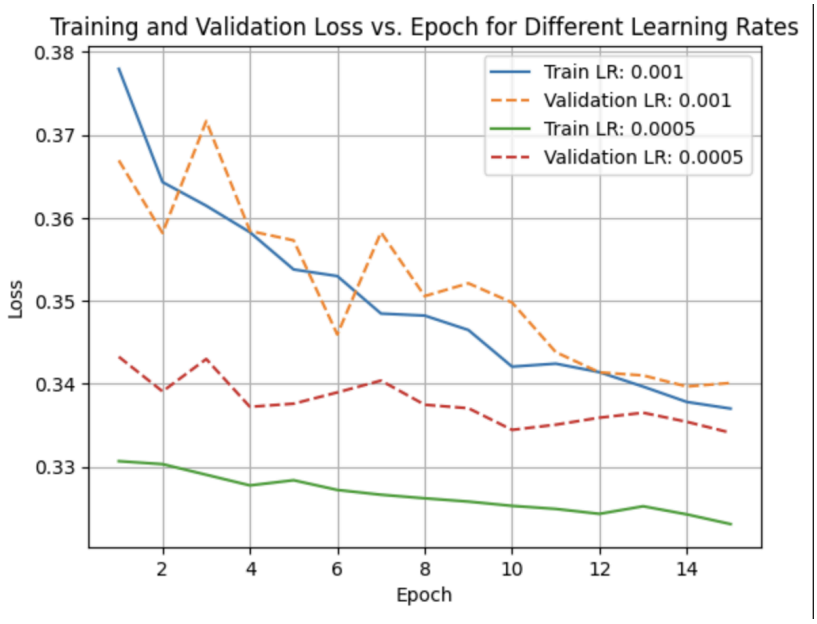


Figure B1: Train and Validation Loss for ResNet18 (15 epochs)

AlexNet (15 epoch)

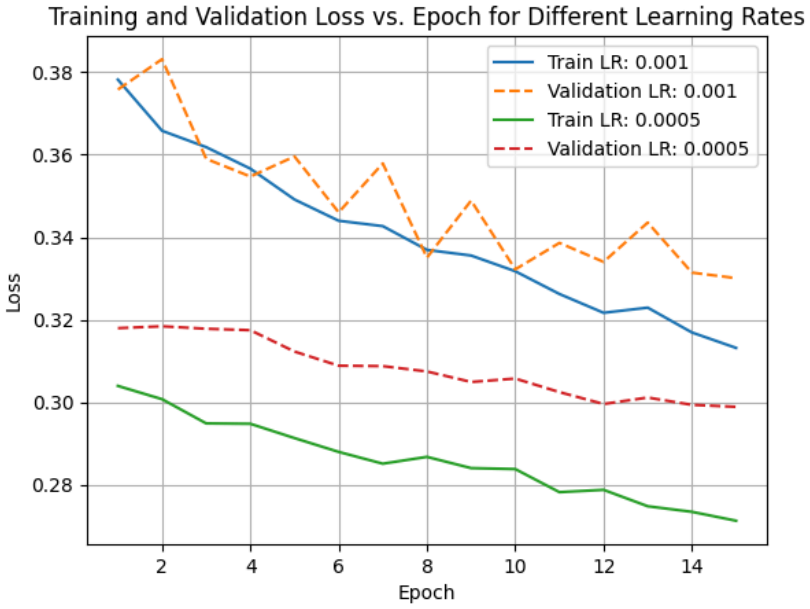


Figure B2: Train and Validation Loss for AlexNet (15 epochs)

AlexNet (30 epoch)

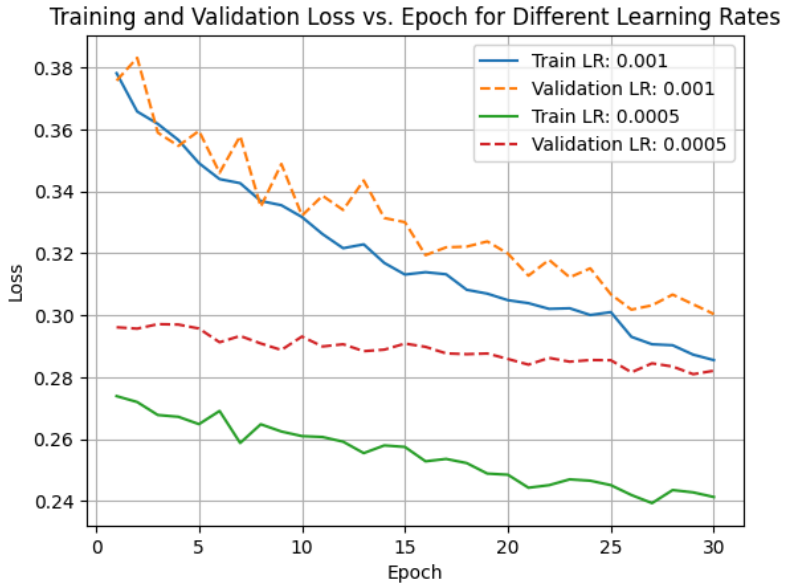


Figure B3: Train and Validation Loss for AlexNet (30 epochs)

Custom AlexNet (15 epochs)

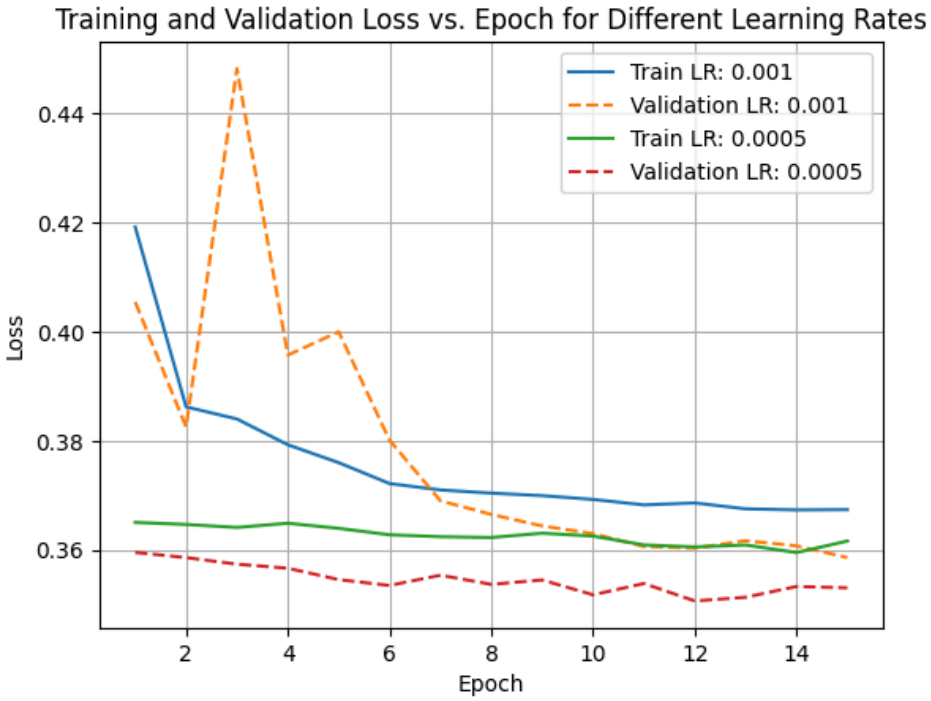


Figure B4: Train and Validation Loss for custom AlexNet (15 epochs)

Custom AlexNet (30 epochs)

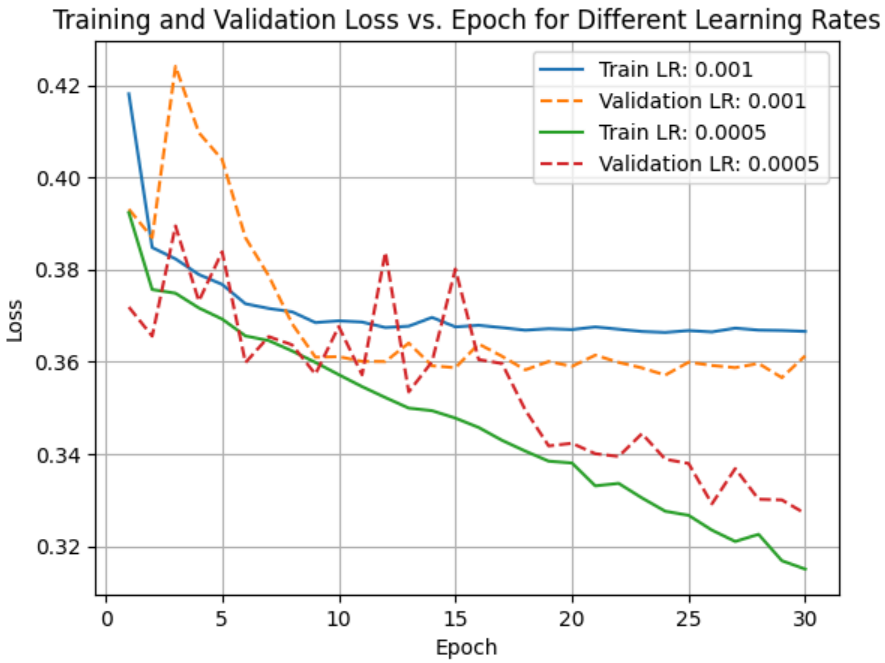


Figure B5: Train and Validation Loss for custom AlexNet (30 epochs)

Custom DenseNet (15 epochs)

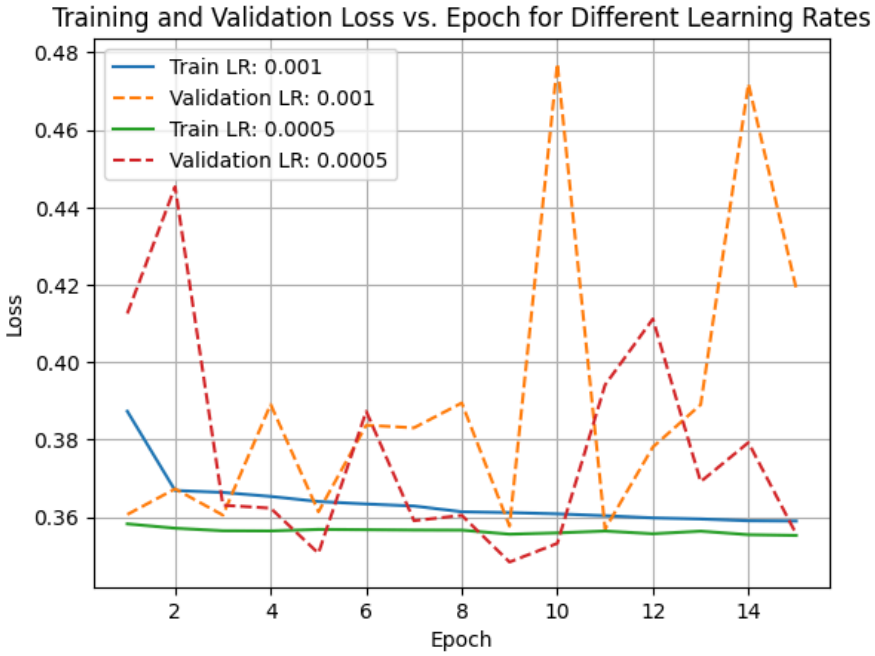


Figure B6: Train and Validation Loss for custom DenseNet (30 epochs)

Appendix C: Test Results for the 3 Successful Models

For ResNet18, AlexNet, and our custom AlexNet model, we explored various combinations of learning rates and epochs. The evaluation results for each of the models with the specified parameters can be found below. Learning rate is shortened to LR in the section headers.

AlexNet (Epoch=15, LR=0.001)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Alexnet Lr 0.001	Atelectasis	0.769231	0.827314	0.11976	0.207254	0.33811
	Cardiomegaly	0.733333	0.891648	0.106796	0.186441	
	Consolidation	1	0.88149	0.027778	0.054054	
	Edema	0.769231	0.924379	0.421053	0.544218	
	Effusion	0.916667	0.774266	0.052381	0.099099	
	Emphysema	0.736842	0.927765	0.191781	0.304348	
	Fibrosis	0.583333	0.900677	0.077778	0.137255	
	Hernia	0.5	0.946953	0.021277	0.040816	
	Infiltration	0.724138	0.727991	0.082677	0.14841	
	Mass	1	0.892777	0.030612	0.059406	
	No Finding	0	0.93228	0	0	
	Nodule	1	0.882619	0.009524	0.018868	
	Pleural Thickening	1	0.882619	0.028037	0.054545	
	Pneumonia	1	0.897291	0.01087	0.021505	
	Pneumothorax	0.761905	0.901806	0.163265	0.268908	

Table C1: Evaluation Metrics for AlexNet (Epoch=15, LR=0.001)

AlexNet (Epoch=15, LR=0.0005)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Alexnet Lr 0.0005	Atelectasis	1	0.837472	0.137725	0.242105	0.30319
	Cardiomegaly	0.730159	0.916479	0.446602	0.554217	
	Consolidation	1	0.888262	0.083333	0.153846	
	Edema	0.825	0.922122	0.347368	0.488889	
	Effusion	0.852941	0.817156	0.27619	0.417266	
	Emphysema	0.71875	0.933409	0.315068	0.438095	
	Fibrosis	0.6875	0.905192	0.122222	0.207547	
	Hernia	0.5	0.946953	0.021277	0.040816	
	Infiltration	0.90625	0.742664	0.114173	0.202797	
	Mass	0.857143	0.895034	0.061224	0.114286	
	No Finding	0	0.93228	0	0	
	Nodule	0.8	0.888262	0.07619	0.13913	
	Pleural Thickening	1	0.888262	0.074766	0.13913	
	Pneumonia	0.714286	0.899549	0.054348	0.10101	
	Pneumothorax	0.870968	0.91535	0.27551	0.418605	

Table C2: Evaluation Metrics for AlexNet (Epoch=15, LR=0.0005)

AlexNet (Epoch=30, LR=0.001)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Alexnet Lr 0.001	Atelectasis	0.88	0.832957	0.131737	0.229167	0.30767
	Cardiomegaly	0.823529	0.908578	0.271845	0.408759	
	Consolidation	1	0.887133	0.074074	0.137931	
	Edema	0.885714	0.923251	0.326316	0.476923	
	Effusion	0.888889	0.786682	0.114286	0.202532	
	Emphysema	0.730769	0.931151	0.260274	0.383838	
	Fibrosis	0.8	0.901806	0.044444	0.084211	
	Hernia	1	0.948081	0.021277	0.041667	
	Infiltration	0.83871	0.73702	0.102362	0.182456	
	Mass	1	0.896163	0.061224	0.115385	
	No Finding	0	0.931151	0	0	
	Nodule	0.857143	0.887133	0.057143	0.107143	
	Pleural Thickening	1	0.883747	0.037383	0.072072	
	Pneumonia	1	0.901806	0.054348	0.103093	
	Pneumothorax	0.823529	0.914221	0.285714	0.424242	

Table C3: Evaluation Metrics for AlexNet (Epoch=30, LR=0.001)

AlexNet (Epoch=30, LR=0.0005)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	
Alexnet Lr 0.005	Atelectasis	0.653846	0.8386	0.305389	0.416327	0.28273
	Cardiomegaly	0.86	0.924379	0.417476	0.562092	
	Consolidation	0.826087	0.895034	0.175926	0.290076	
	Edema	0.733333	0.924379	0.463158	0.567742	
	Effusion	0.741667	0.828442	0.42381	0.539394	
	Emphysema	0.870968	0.943567	0.369863	0.519231	
	Fibrosis	0.714286	0.905192	0.111111	0.192308	
	Hernia	0.25	0.940181	0.06383	0.101695	
	Infiltration	0.717949	0.751693	0.220472	0.337349	
	Mass	0.882353	0.904063	0.153061	0.26087	
	No Finding	0	0.93228	0	0	
	Nodule	0.923077	0.893905	0.114286	0.20339	
	Pleural Thickening	0.916667	0.890519	0.102804	0.184874	
	Pneumonia	0.833333	0.905192	0.108696	0.192308	
	Pneumothorax	0.878788	0.917607	0.295918	0.442748	

Table C4: Evaluation Metrics for AlexNet (Epoch=30, LR=0.0005)

ResNet (Epoch=15, LR=0.001)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Resnet Lr 0.001	Atelectasis	0.42268	0.79458	0.24551	0.31061	0.34433
	Cardiomegaly	0	0.88375	0	0	
	Consolidation	0	0.8781	0	0	
	Edema	0.4	0.88939	0.06316	0.10909	
	Effusion	0.69231	0.76862	0.04286	0.08072	
	Emphysema	0	0.91761	0	0	
	Fibrosis	0.2	0.89503	0.01111	0.02105	
	Hernia	0	0.94695	0	0	
	Infiltration	0.51429	0.71445	0.07087	0.12457	
	Mass	0	0.88939	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	1	0.88262	0.00952	0.01887	
	Pleural Thickening	0	0.87923	0	0	
	Pneumonia	1	0.89729	0.01087	0.0215	
	Pneumothorax	1	0.892777	0.030612	0.059406	

Table C5: Evaluation Metrics for ResNet (Epoch=15, LR=0.001)

ResNet (Epoch=15, LR=0.0005)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Resnet Lr 0.0005	Atelectasis	0.8	0.814898	0.023952	0.046512	0.33591
	Cardiomegaly	0.666667	0.884876	0.019417	0.037736	
	Consolidation	0	0.876975	0	0	
	Edema	0.429907	0.875847	0.484211	0.455446	
	Effusion	0.73913	0.775395	0.080952	0.145923	
	Emphysema	0	0.916479	0	0	
	Fibrosis	0.333333	0.897291	0.011111	0.021505	
	Hernia	0	0.946953	0	0	
	Infiltration	0.38835	0.687359	0.15748	0.22409	
	Mass	0	0.889391	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	0.5	0.88149	0.009524	0.018692	
	Pleural Thickening	0	0.879233	0	0	
	Pneumonia	1	0.897291	0.01087	0.021505	
	Pneumothorax	0.769231	0.897291	0.102041	0.180180	

Table C6: Evaluation Metrics for ResNet (Epoch=15, LR=0.0005)

Custom AlexNet (Epoch=15, LR=0.001)

Model	Class	Metrics				
		Accuracy	Precision	Recall	F1	Loss
Custom Alexnet Lr 0.001	Atelectasis	0.811512	0	0	0	0.36413
	Cardiomegaly	0.883747	0	0	0	
	Consolidation	0.878104	0	0	0	
	Edema	0.892777	0	0	0	
	Effusion	0.762980	0	0	0	
	Emphysema	0.917607	0	0	0	
	Fibrosis	0.898420	0	0	0	
	Hernia	0.946953	0	0	0	
	Infiltration	0.713318	0	0	0	
	Mass	0.889391	0	0	0	
	No Finding	0.932280	0	0	0	
	Nodule	0.881490	0	0	0	
	Pleural Thickening	0.879233	0	0	0	
	Pneumonia	0.896163	0	0	0	
	Pneumothorax	0.889391	0	0	0	

Table C7: Evaluation Metrics for Custom AlexNet (Epoch=15, LR=0.001)

Custom AlexNet (Epoch=15, LR=0.0005)

Model	Class	Metrics				
		Accuracy	Precision	Recall	F1	Loss
Custom Alexnet Lr 0.0005	Atelectasis	0.811512	0	0	0	0.35745
	Cardiomegaly	0.883747	0	0	0	
	Consolidation	0.878104	0	0	0	
	Edema	0.892777	0	0	0	
	Effusion	0.762980	0	0	0	
	Emphysema	0.917607	0	0	0	
	Fibrosis	0.898420	0	0	0	
	Hernia	0.946953	0	0	0	
	Infiltration	0.713318	0	0	0	
	Mass	0.889391	0	0	0	
	No Finding	0.932280	0	0	0	
	Nodule	0.881490	0	0	0	
	Pleural Thickening	0.879233	0	0	0	
	Pneumonia	0.896163	0	0	0	
	Pneumothorax	0.889391	0	0	0	

Table C8: Evaluation Metrics for Custom AlexNet (Epoch=15, LR=0.0005)

Custom AlexNet (Epoch=30, LR=0.001)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	
Custom Alexnet Lr 0.001	Atelectasis	0	0.811512	0	0	0.36390
	Cardiomegaly	0	0.883747	0	0	
	Consolidation	0	0.878104	0	0	
	Edema	0	0.892777	0	0	
	Effusion	0	0.76298	0	0	
	Emphysema	0	0.917607	0	0	
	Fibrosis	0	0.89842	0	0	
	Hernia	0	0.946953	0	0	
	Infiltration	0	0.713318	0	0	
	Mass	0	0.889391	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	0	0.88149	0	0	
	Pleural Thickening	0	0.879233	0	0	
	Pneumonia	0	0.896163	0	0	
	Pneumothorax	0	0.889391	0	0	

Table C9: Evaluation Metrics for Custom AlexNet (Epoch=30, LR=0.001)

Custom AlexNet (Epoch=30, LR=0.0005)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Custom AlexNet Lr 0.0005 Epoch 30	Atelectasis	0.809524	0.826185	0.101796	0.180851	0.33252
	Cardiomegaly	0.684211	0.891648	0.126214	0.213115	
	Consolidation	0.9	0.887133	0.083333	0.152542	
	Edema	0.76	0.907449	0.2	0.316667	
	Effusion	0.56962	0.775395	0.214286	0.311419	
	Emphysema	0.636364	0.920993	0.09589	0.166667	
	Fibrosis	1	0.900677	0.022222	0.043478	
	Hernia	0	0.945824	0	0	
	Infiltration	0.733333	0.72912	0.086614	0.15493	
	Mass	0.818182	0.897291	0.091837	0.165138	
	No Finding	0	0.93228	0	0	
	Nodule	1	0.886005	0.038095	0.073394	
	Pleural Thickening	0.857143	0.884876	0.056075	0.105263	
	Pneumonia	1	0.897291	0.01087	0.021505	
	Pneumothorax	0.718750	0.905192	0.234694	0.353846	

Table C10: Evaluation Metrics for Custom AlexNet (Epoch=30, LR=0.0005)

Custom ResNet (Epoch=15, LR=0.001)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	
Custom Resnet Lr 0.001	Atelectasis	0.42268	0.794582	0.245509	0.310606	0.34431
	Cardiomegaly	0	0.883747	0	0	
	Consolidation	0	0.878104	0	0	
	Edema	0.428571	0.890519	0.063158	0.110092	
	Effusion	0.692308	0.768623	0.042857	0.080717	
	Emphysema	0	0.917607	0	0	
	Fibrosis	0.2	0.895034	0.011111	0.021053	
	Hernia	0	0.946953	0	0	
	Infiltration	0.514286	0.714447	0.070866	0.124567	
	Mass	0	0.889391	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	1	0.882619	0.009524	0.018868	
	Pleural Thickening	0	0.879233	0	0	
	Pneumonia	1	0.897291	0.01087	0.021505	
	Pneumothorax	1	0.892777	0.030612	0.059406	

Table C11: Evaluation Metrics for Custom ResNet (Epoch=15, LR=0.001)

Custom ResNet (Epoch=15, LR=0.0005)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Custom Resnet Lr 0.0005	Atelectasis	0.8	0.814898	0.023952	0.046512	0.33591
	Cardiomegaly	0.666667	0.884876	0.019417	0.037736	
	Consolidation	0	0.876975	0	0	
	Edema	0.429907	0.875847	0.484211	0.455446	
	Effusion	0.727273	0.774266	0.07619	0.137931	
	Emphysema	0	0.916479	0	0	
	Fibrosis	0.333333	0.897291	0.011111	0.021505	
	Hernia	0	0.946953	0	0	
	Infiltration	0.38835	0.687359	0.15748	0.22409	
	Mass	0	0.889391	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	0.5	0.88149	0.009524	0.018692	
	Pleural Thickening	0	0.879233	0	0	
	Pneumonia	1	0.897291	0.01087	0.021505	
	Pneumothorax	0.769231	0.897291	0.102041	0.18018	

Table C12: Evaluation Metrics for Custom ResNet (Epoch=15, LR=0.0005)

Custom DenseNet (Epoch=15, LR=0.001)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	
Custom DenseNet Lr 0.001	Atelectasis	0	0.811512	0	0	0.36683
	Cardiomegaly	0	0.883747	0	0	
	Consolidation	0	0.878104	0	0	
	Edema	0	0.892777	0	0	
	Effusion	0	0.76298	0	0	
	Emphysema	0	0.917607	0	0	
	Fibrosis	0	0.89842	0	0	
	Hernia	0	0.945824	0	0	
	Infiltration	0	0.713318	0	0	
	Mass	0	0.889391	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	0	0.88149	0	0	
	Pleural Thickening	0	0.879233	0	0	
	Pneumonia	0	0.896163	0	0	
	Pneumothorax	0	0.889391	0	0	

Table C13: Evaluation Metrics for Custom DenseNet (Epoch=15, LR=0.001)

Custom DenseNet (Epoch=15, LR=0.0005)

Model	Class	Metrics				
		Precision	Accuracy	Recall	F1	Loss
Custom DenseNet Lr 0.0005 Epoch 15	Atelectasis	0	0.811512	0	0	0.35928
	Cardiomegaly	0	0.883747	0	0	
	Consolidation	0	0.878104	0	0	
	Edema	0	0.892777	0	0	
	Effusion	0	0.76298	0	0	
	Emphysema	0	0.917607	0	0	
	Fibrosis	0	0.89842	0	0	
	Hernia	0	0.945824	0	0	
	Infiltration	0	0.713318	0	0	
	Mass	0	0.889391	0	0	
	No Finding	0	0.93228	0	0	
	Nodule	0	0.88149	0	0	
	Pleural Thickening	0	0.879233	0	0	
	Pneumonia	0	0.896163	0	0	
	Pneumothorax	0	0.889391	0	0	

Table C14: Evaluation Metrics for Custom DenseNet (Epoch=15, LR=0.0005)