Final Project

Chest X-Ray Images

Classifier

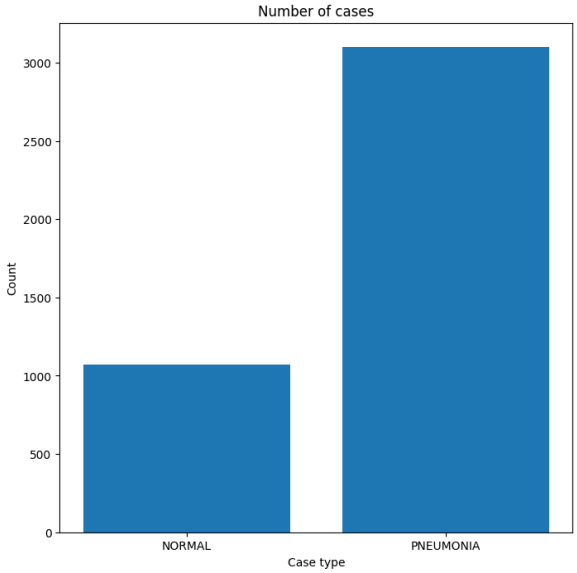
Yehuda Daniel
Tal Teri
Dr. Moshe Butman
26 January 2024

1. Introduction

In this project, we were given a dataset of chest X-Ray images selected from patients aged one to five, from Guangzhou Women and Children's Medical Center.

1.1. Data

Our dataset contains a total of 5,863 X-Ray images, in various resolutions, divided into 2 classes, normal results, and results displaying pneumonia.
At first glance at the data, an imbalance can be seen, as presented in the following graph.



1.2. Problem

Initial exploration of the data unveiled several key challenges. Notably, the limited size of the dataset might hinder robust training, further more significant variations in image resolutions may pose a challenge, as selecting a uniform resolution for the model introduces trade-offs and potential information loss. Moreover, the data exhibits a significant class imbalance, as illustrated above, can lead to biased models that prioritize the majority class, and perform poorly on the minority class.

A. Binary Classification

For this section, we need to use the given dataset and construct a binary model to distinguish between Normal and Pneumonia images.

A.1. General Approach

For optimal model performance, image preprocessing was important. Recognizing the use of ResNet architecture for medical image analysis, including problems similar to ours, we adopted a 224x224 pixel resolution for image resizing. This decision aligns with the standard input size for pre-trained ResNet model, facilitating potential future utilization of transfer learning techniques. Furthermore, to explore the impact of color information on model performance, we evaluated both RGB and grayscale representations of the images.

A.2 Image Augmentation

To mitigate class imbalance, we leveraged the **ImageDataGenerator** from keras. This framework enabled us to apply diverse transformations exclusively to the minority class within our dataset. These transformations included rotations, zooms, width and height shifts, and horizontal flips. By strategically augmenting the underrepresented class, we effectively increased the dataset size and introduced crucial variations, using a split ratio of 80-20.

A.3 First Experiment

A.3.1 Base Model

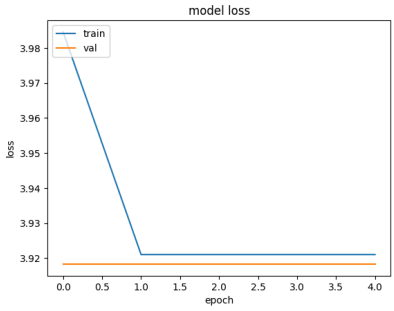
Our model was implemented within the keras framework, leveraging the aforementioned data augmentation techniques exclusively on the training set to enhance diversity and mitigate overfitting. We utilized grayscale image representations to accentuate relevant features for the classification task.

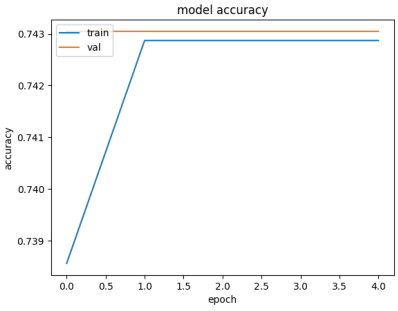
The architecture comprises four convolutional blocks sequentially followed by two fully connected layers, culminating in a single output neuron.
All convolutional layers employ a (3, 3) filter with padding to preserve image size during feature extraction. The ReLU activation function, known for its effectiveness in non-linear learning, is consistently used throughout the model.

The initial convolutional block starts with 32 filters and incorporates batch normalization for optimized learning, followed by max-pooling of stride 2 with (2, 2) filter, reduces the dimensionality of feature maps while learning crucial spatial information.
Subsequent blocks follow a similar structure, incrementally increasing the filter count (64 in block 2, 128 in blocks 3 and 4) to capture progressively more complex features.

Prior to entering the dense layers, the extracted features are flattened into a one-dimensional vector. The first dense layer houses 1024 units, accompanied by a dropout layer to prevent overfitting and promote generalization. Another 1024-unit layer follows, leading to the final output layer with a single neuron for the classification task.
All dense layers consistently employe the ReLU activation function due to its ability to model non-linear relationships within the data.

To optimize the model's learning process, we employed the Adam optimizer with a learning rate of 0.0007. Binary cross-entropy, a suitable choice for binary classification tasks, served as the loss function, measuring the model's performance in predicting the correct class.





Despite implementing a carefully designed architecture with a suitable CNN blocks for medical feature extraction, the model's performance on the test set fell short of expectations, achieving an accuracy of 62.5%.

```
624/624 [==============================] - 3s 5ms/step - loss: 5.7185 - accuracy: 0.6250
Test Accuracy: 62.50%
```

This suggests that further investigation into alternative architectures or optimization techniques might be necessary.

A.4 Second Experiment

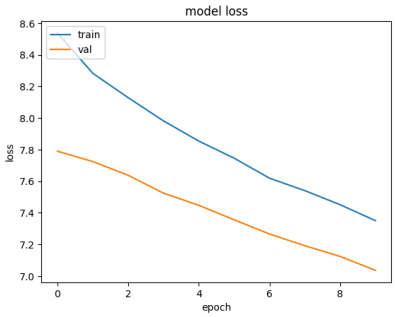
A.4.1 Base Model

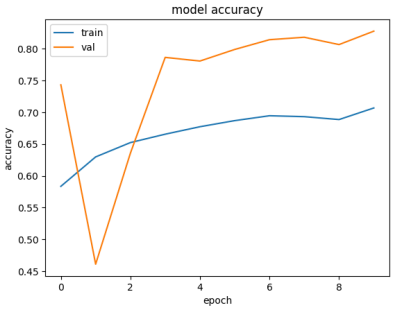
Following the insights gained from our initial experiment, we recognized the potential benefits of utilizing RGB image representations. Subsequently, we implemented this change and maintained the data augmentation techniques that proved effective in the previous iteration. Furthermore, we reduced the batch size from 32 to 16, anticipating a potential positive impact on the learning process.

This time, we embraced the well-established practice of transfer learning. We utilized Keras's pre-trained EfficientNetB3 model, originally trained on the ImageNet dataset, as our base architecture. This choice leverages the extensive learning inherent in the pre-trained weights, while requiring a 224x224 input shape to match the EfficientNetB3's design.

The adopted architecture incorporates batch normalization, followed by a fully connected layer with 256 units. To resist potential overfitting and encourage generalization, we employed L1 and L2 regularization on the kernel, activity, and bias parameters. These regularizers promote sparsity and reduce model complexity, respectively. To further mitigate overfitting, we implemented a 40% dropout layer, forcing the model to rely on a wider range of features during training. Finally, a fully connected layer with 2 units and a softmax activation function provides the final output layer for our binary classification task.

Recognizing the value of fine-tuning the pre-trained weights, all layers of the EfficientNetB3 model were frozen to preserve the valuable learned features. The AdaMax optimizer with a learning rate of 0.00001 was used in this initial phase. After this fine-tuning step, training continued with the categorical cross-entropy loss function.





Although transfer learning should have improved the performance, we can see a growth in accuracy of merely 14% setting the accuracy at 71.15% for the test.

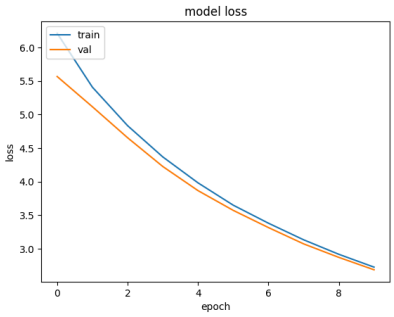
```
39/39 [==============================] - 5s 117ms/step - loss: 7.0801 - accuracy: 0.7115
Test Accuracy: 71.15%
```

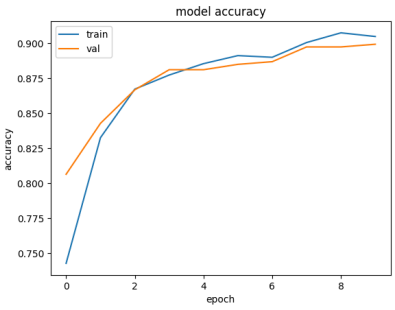
This may indicate that a different architecture might suit a better accuracy.

A.5 Third Experiment

A.5.1 Base Model

Seeking to optimize our model for this specific task, we investigated alternative architectures. Based on preliminary analysis and task characteristics, we transitioned from EfficientNetB3 to VGG16, hypothesizing its strengths in image classification would align better with our data and objectives. Furthermore, we cautiously adjusted the learning rate to 0.0001 to enhance convergence towards optimal solutions while maintaining frozen layers to prevent overfitting. We retained the Adamax optimizer and categorical cross-entropy loss function due to their suitability for our problem and consistent performance in the initial architecture.





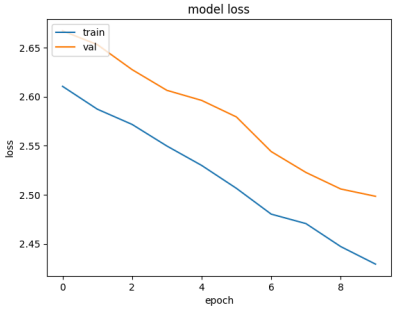
Gladly, this architecture brought better performance with an accuracy of 88.46%.

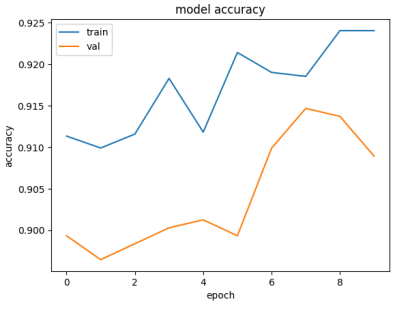
```
39/39 [==============================] - 5s 113ms/step - loss: 2.7562 - accuracy: 0.8846
Test Accuracy: 88.46%
```

A.6 Fourth Experiment

A.6.1 Fine-Tuning

We build upon our previous experiment that utilized pre-trained VGG16 weights with our dataset. In this iteration, we unfreeze the last three convolutional layers of VGG16 while maintaining the frozen state of the earlier layers. This approach aims to leverage the learned features from pre-training while allowing our model to adapt to our specific dataset through fine-tuning. We compare the approach with our previous experiment and analyze the impact of unfreezing layers on performance.





Showing the test results for this experiment, achieving 90.06% accuracy

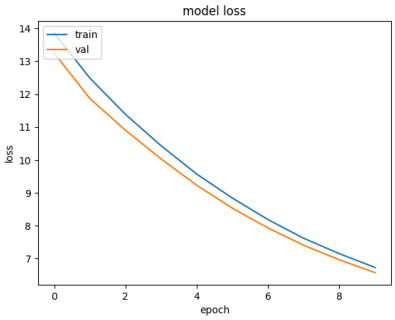
```
39/39 [==============================] - 5s 112ms/step - loss: 2.5472 - accuracy: 0.9006
Test Accuracy: 90.06%
```

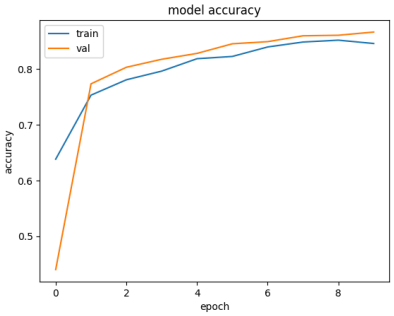
A.7 Fifth Experiment

A.7.1 Base model

For our next experiment we tried to switch to ResNet50 architecture while still using imagenet pre-trained weights which were proven to be useful in our previous experiments.

We freeze all the layers of the resnet architecture in order to keep to architecture results unchanged. Our chosen optimizer this time is Adamax with a learning rate of 0.00001, and the is still categorical cross entropy**.**





This model was proven to be less accurate from our fourth experiment's model which delivered an accuracy rate of 90.06% compared to 85.74% provided by the resnet based model.

```
39/39 [==============================] - 5s 114ms/step - loss: 6.5860 - accuracy: 0.8574
Test Accuracy: 85.74%
```

B. Multi-Class Classification

For this section, we need to use the given dataset and construct a multi-class model to distinguish between Normal, Virus and Bacteria images.

B.1. General Approach

Our data is divided into two classes, normal and pneumonia. In order to build a multi class classifier we had to split the pneumonia folder and change it into two new folders, virus and bacteria. As soon as we finished reorganizing our dataset we continued with the same approach with the binary classification.

For optimal model performance, image preprocessing was important here as well, Recognizing the use of VGG16 architecture for medical image analysis, including problems similar to ours, we adopted a 224x224 pixel resolution for image resizing. This decision aligns with the standard input size for pre-trained VGG16 model, facilitating potential future utilization of transfer learning techniques. This time we evaluated only RGB representations of the images.

B.2 Image Augmentation

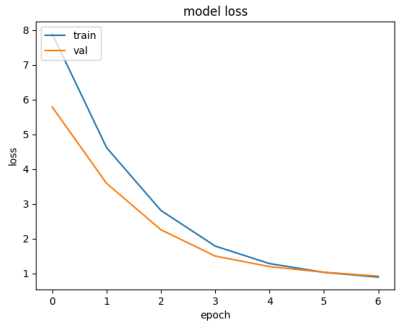
To mitigate class imbalance, we leveraged the ImageDataGenerator from keras. This framework enabled us to apply diverse transformations exclusively to the minority class within our dataset. These transformations included rotations, zooms, width and height shifts, and horizontal flips. By strategically augmenting the underrepresented class, we effectively increased the dataset size and introduced crucial variations, using a split ratio of 80-20.

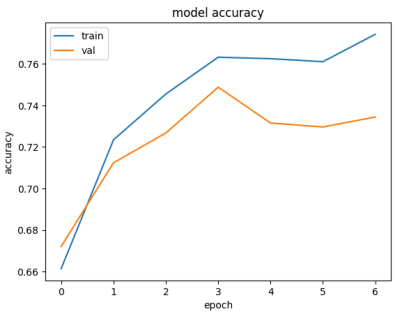
B.3 First Experiment

B.3.1 Base Model

Following the insights gained from our binary classifier experiment, we recognized the potential benefits of utilizing RGB image representations. Furthermore, we acknowledged the effectiveness of using VGG16 network with ImageNet pre-trained weights for our transfer learning. We set the batch size to 16, anticipating a potential positive impact on the learning process.

We picked VGG16, which proved its strengths in image classification on our binary model. Furthermore, we picked the learning rate to be 0.001 to enhance convergence towards optimal solutions while maintaining frozen layers to prevent overfitting. We retained the Adamax optimizer and categorical cross-entropy loss function due to their suitability for our problem and consistent performance in the initial architecture.





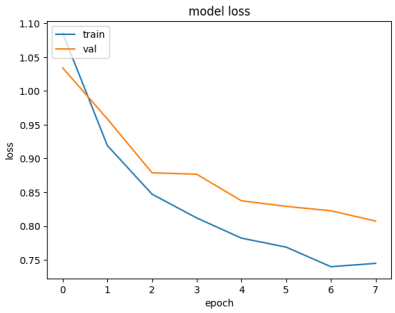
Gladly, this architecture brought better performance with an accuracy of 82.85%.

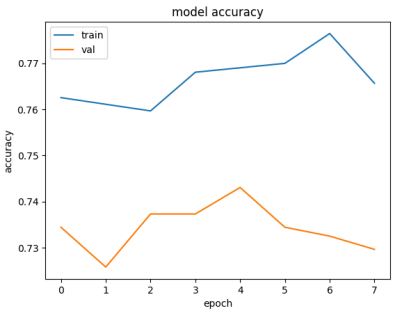
```
39/39 [==============================] - 5s 122ms/step - loss: 1.4273 - accuracy: 0.8285
Test Accuracy: 82.85%
```

B.4 Second Experiment

A.4.1 Fine-Tuning

We build upon our previous experiment that utilized pre-trained VGG16 weights with our dataset. In this iteration, we unfreeze the last three convolutional layers of VGG16 while maintaining the frozen state of the earlier layers. This approach aims to leverage the learned features from pre-training while allowing our model to adapt to our specific dataset through fine-tuning. We compare the approach with our previous experiment and analyze the impact of unfreezing layers on performance.



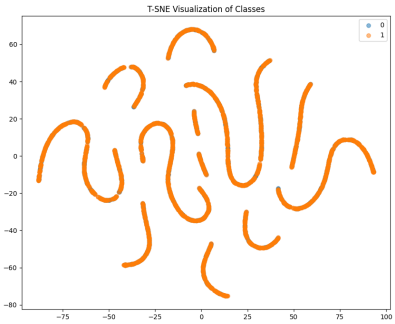


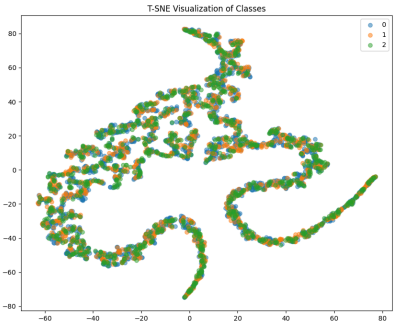
Showing the test results for this experiment, achieving 82.21% accuracy

```
39/39 [==============================] - 5s 126ms/step - loss: 0.7272 - accuracy: 0.8221
Test Accuracy: 82.21%
```

C.1 KNN clssification

This study proposes an approach for pneumonia detection in chest X-rays. We combine latent embeddings from both binary and multi-class deep learning models to train a KNN classifier. This two-stage approach offers improved accuracy, works with limited labeled data, and provides interpretability. We compare our method to others, analyze KNN hyperparameters, and discuss limitations. This approach offers potential for robust pneumonia detection, especially with limited data, and warrants further research for clinical validation.





D.1. Anomaly Detection

D.1.1 Base Model

Chest X-rays play a crucial role in diagnosing various pulmonary pathologies, with pneumonia being a prominent example. However, the limited availability of radiologists and the growing volume of imaging data necessitate the development of automated detection tools. This study explores the efficacy of an autoencoder-based anomaly detection model for identifying pneumonia instances in chest X-rays, utilizing solely normal data for training.