ALGORITHM BOOTCAMP

DEEP LEARNING FOR CHEST X-RAYS

By: Angela Christabel

TABLE OF CONTENTS

Presentation Outline

1. Business Background
2. Data Understanding and Exploratory Data Analysis
3. Modelling and evaluation
4. Conclusion and Recommendation

CORONAHACK

BUSINESS BACKGROUND

Background

I am a data scientist working at a health care institution. During this pandemic, it is important, even more so, to be able to succesfully diagnose patients who have COVID-19, which is a pneumonia-like disease.

CORONAHACK

BUSINESS BACKGROUND

Business Objective

The objective of this project is to predict whether someone has a pneumonia-like disease or is completely healthy, given their chest x-rays. This will make healthcare workers' jobs easier.

CORONAHACK

Project Limitations

Limited time and resources

Due to this project's limited time and resources, I will only be able to tune my model as far as performing hyperparameter tuning and I won't be able to try various techniques to improve my model.

Analytic Approach

Deep Learning Technique

Supervised Learning (classification) to predict whether someone has pneumonia or not.

Performance Measures

Recall supported by AUC scores to minimize prediction error.

Data Understanding and Exploratory Data Analysis

Data Collection

KAGGLE

The dataset I used in this project was downloaded from Kaggle, titled "CoronaHack Chest X-Ray Dataset".

Data shape:
- 5286 rows
- 6 columns

Data Description

X_ray_image_name

File names of all X-ray images contained in this dataset.

Label

Label for each X-ray, whether that X-ray indicates healthy lungs or lungs infected by a pneumonia-like disease.

Label_1_Virus_category

Label for each X-ray, cause for pneumonia-like disease.

Label_2_Virus_category

Label for each X-ray, virus/bacteria that caused the pneumonia-like disease.

Data Description

Index

Unique index for all X-ray images contained in this dataset.

Dataset_type

Specifies whether the certain X-ray image is meant for the training dataset or the test dataset.

CORONAHACK

EMPTY VALUES

```
Unnamed: 0                    0
X_ray_image_name              0
Label                         0
Dataset_type                  0
Label_2_Virus_category     5217
Label_1_Virus_category     1342
dtype: int64
```

All columns

The null values in the Label_2_Virus_category and Label_1_Virus_category columns are probably caused by the fact that the corresponding x-ray images just don't have their causes for pneumonia recorded yet. For further EDA, I filled those empty values with "Unknown".

VALUES IN COLUMNS

Pnemonia	3944
Normal	1342

bacteria	2535
Virus	1407
Unknown	1342
Stress-Smoking	2

Unknown	5217
COVID-19	58
Streptococcus	5
SARS	4
ARDS	2

Label

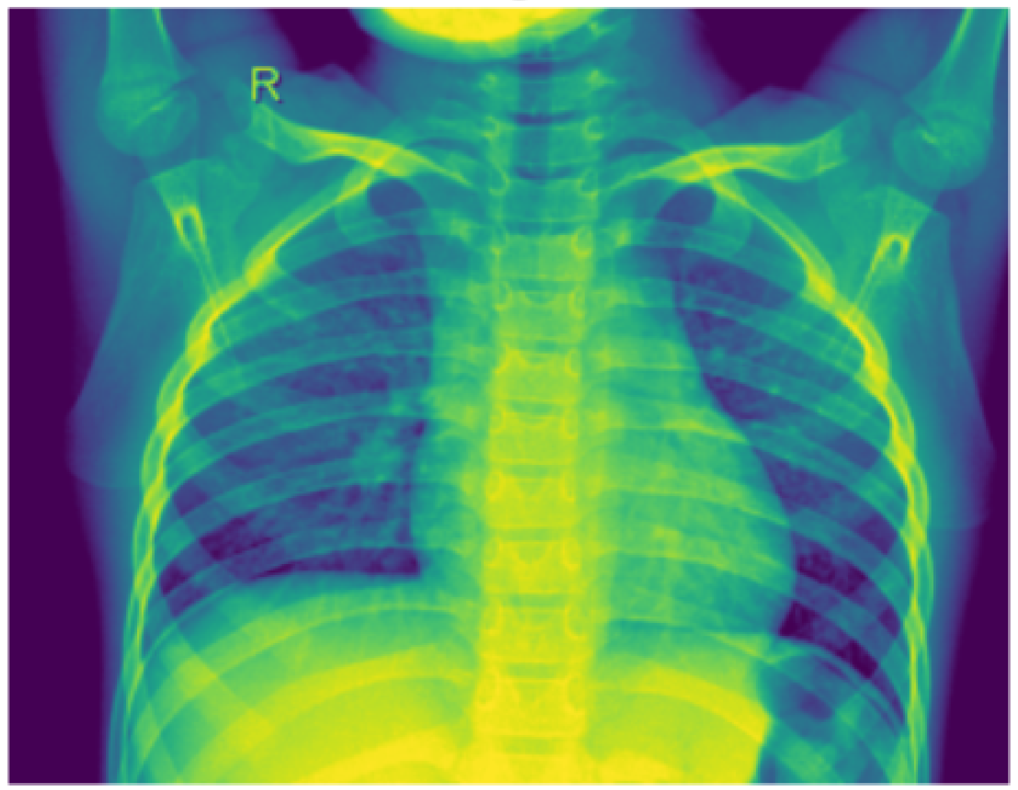
These x-rays are labeled either 'Pnemonia' or 'Normal'.

Label_1_Virus_category

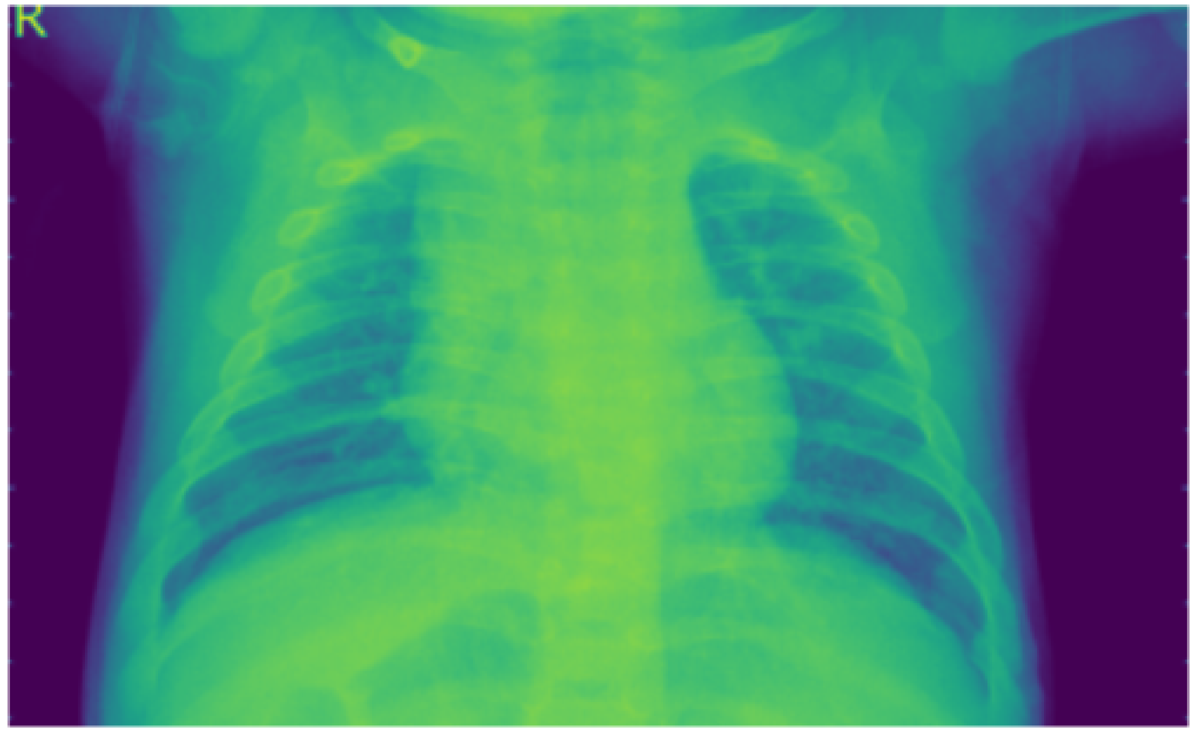
General causes for the pneumonia-like diseases found in these x-rays.

Label_2_Virus_category

Specific causes for the pneumonia-like disease found in these x-rays.



Normal lungs

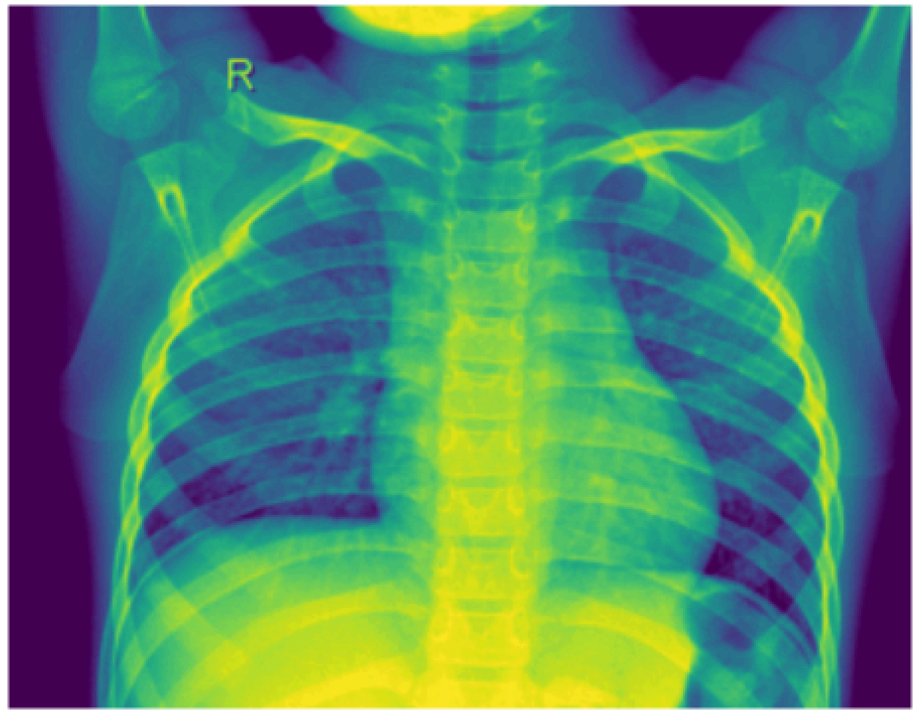


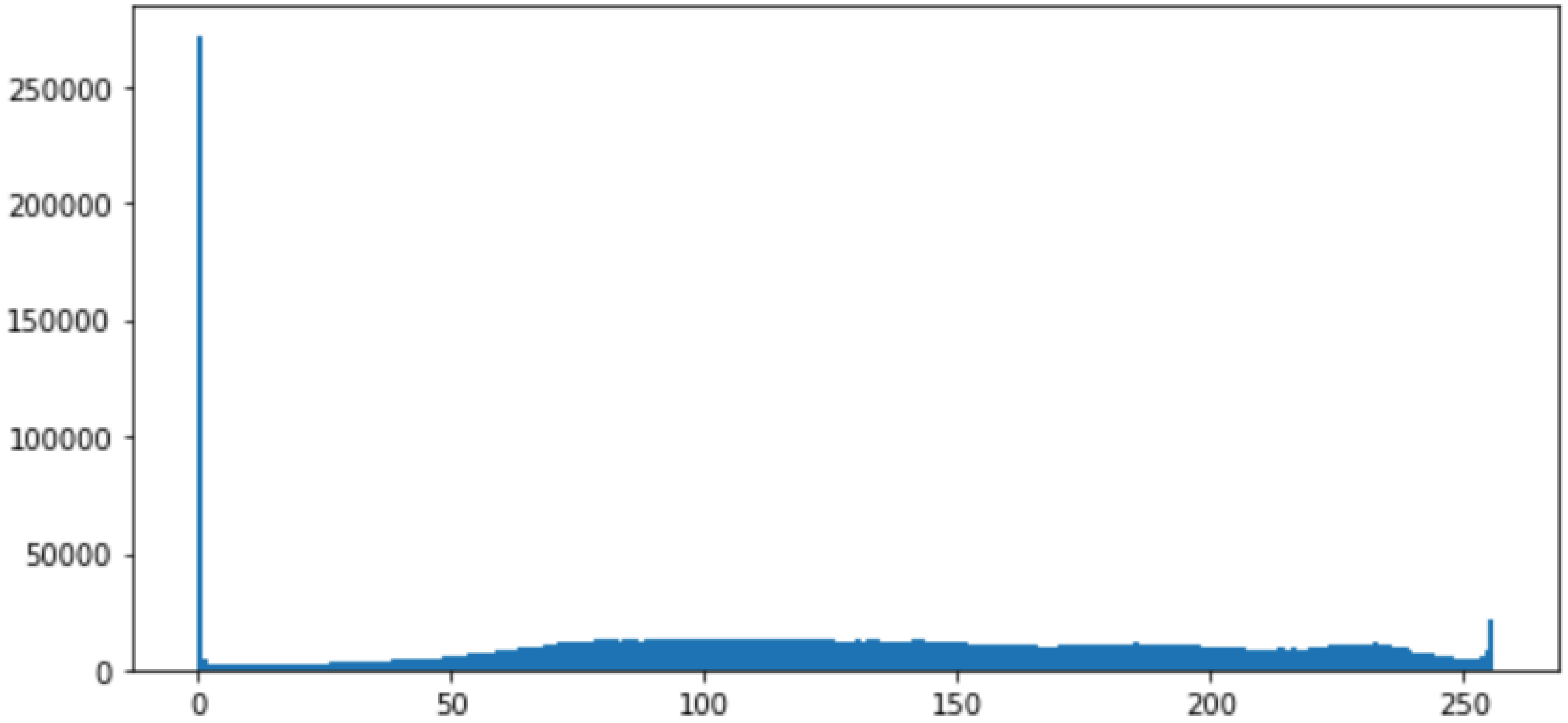
Lungs infected by Pneumonia

CORONAHACK

IMAGE HISTOGRAM

Normal lungs

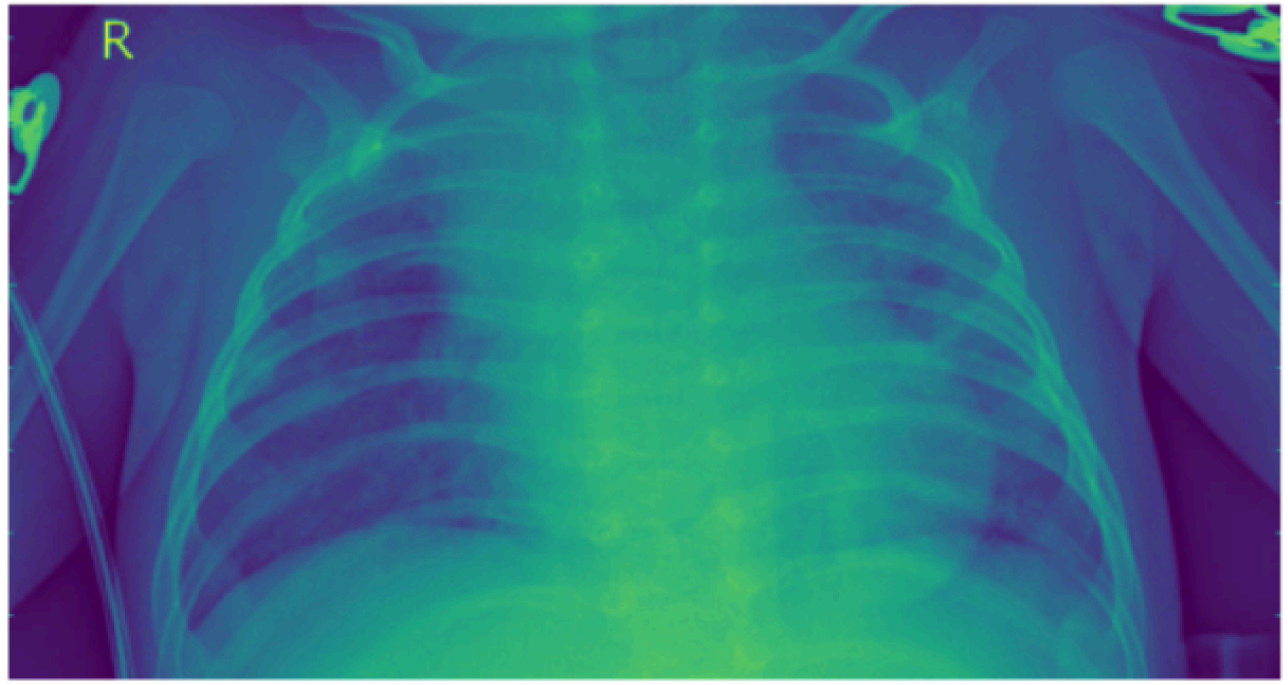


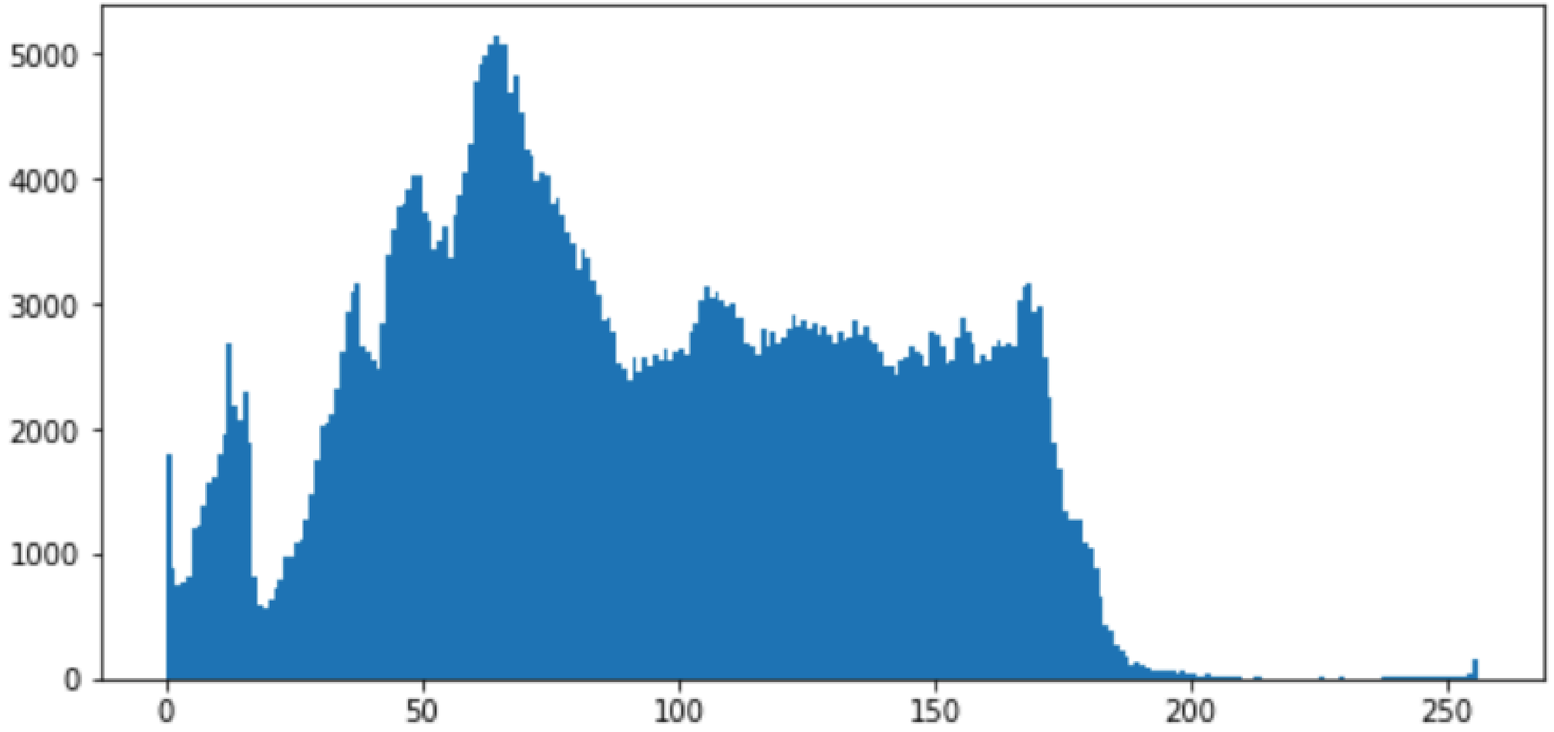


CORONAHACK

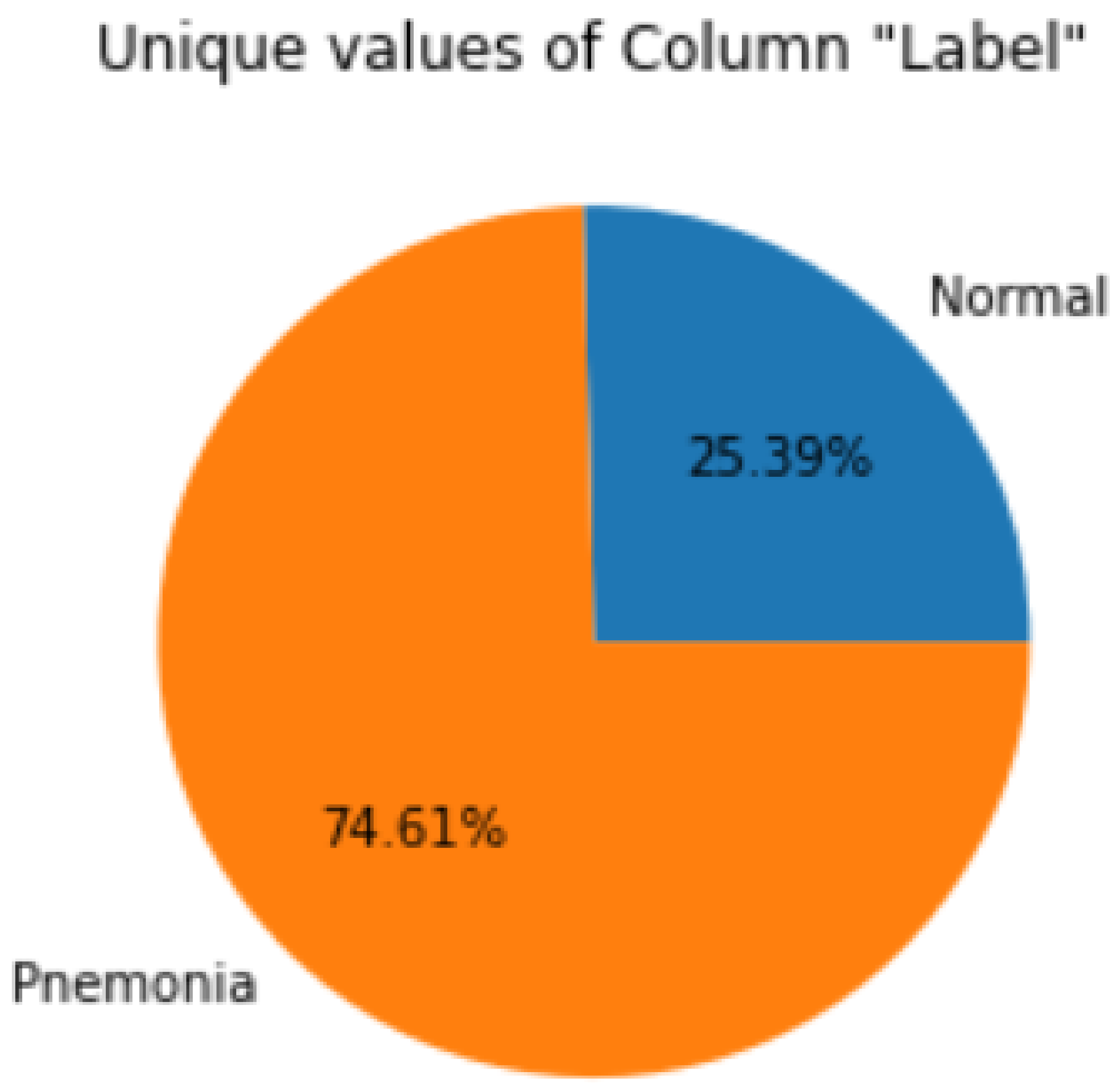
IMAGE HISTOGRAM

Lungs infected by pneumonia





CORONAHACK

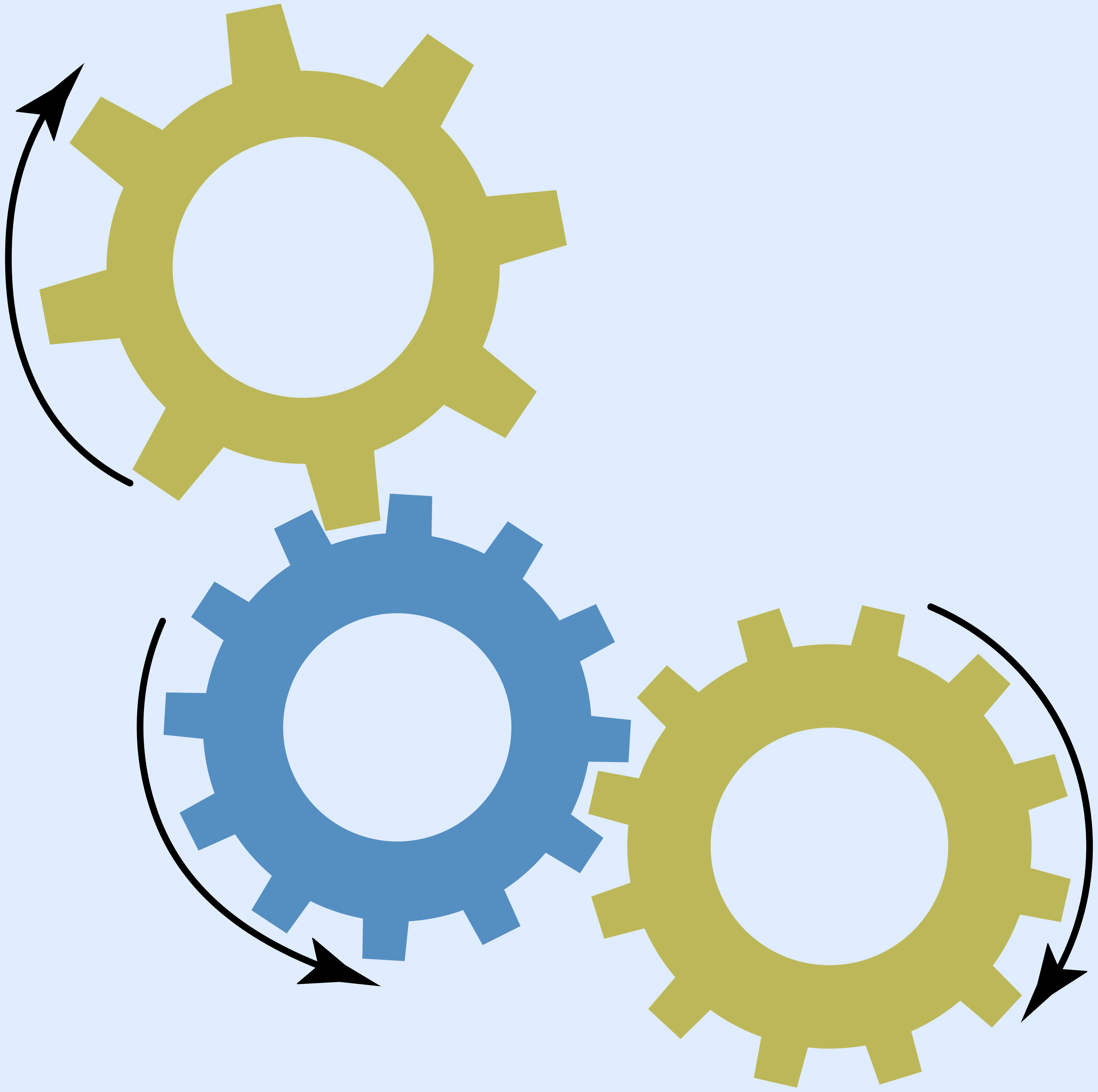


TARGET CLASS DISTRIBUTION

Imbalanced

As we can see from the chart on the left, 'Normal' lungs only make up 25% of the data. But thankfully, the class we care about, 'Pnemonia' is the majority class. Still, we will have to take some measures to handle this imbalance in the data.

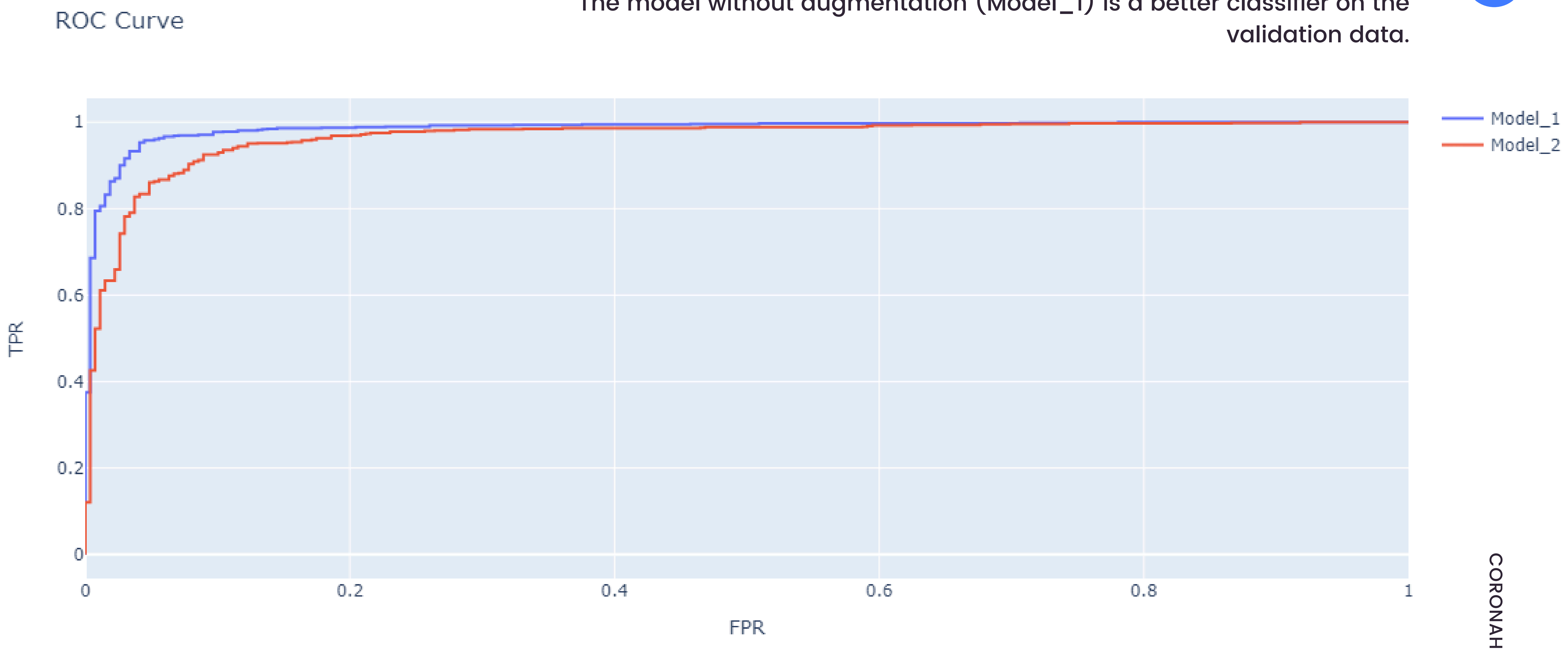
Data Preprocessing



Methods

After testing using augmentation methods, I concluded that using augmentation methods harmed the model's performance, so the only preprocessing I performed on the data was to resize it to 224 x 224 and rescale the features by dividing each pixel by 255.

The model without augmentation (Model_1) is a better classifier on the validation data.



CORONAHACK

MODELLING AND EVALUATION

CORONAHACK

MODELLING STEPS

Step-by-step approach

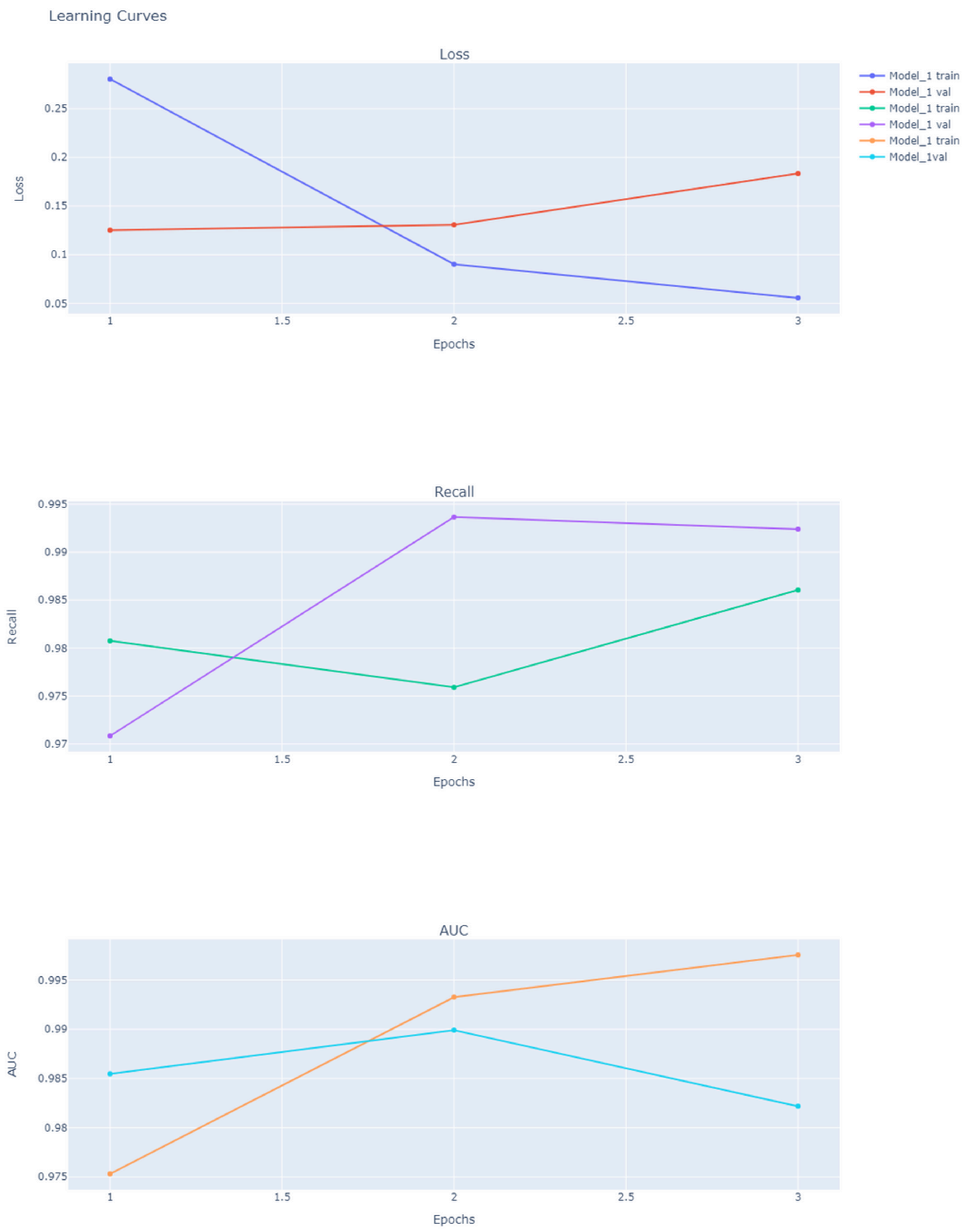
- Create a baseline model
- Compare various pretrained models
- Choose the best model and tune it
- Test and evaluate on unseen data

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 10)      280
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 10)      910
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 218, 218, 10)      910
_________________________________________________________________
flatten (Flatten)            (None, 475240)            0
_________________________________________________________________
dense (Dense)                (None, 1)                 475241
=================================================================
Total params: 477,341
Trainable params: 477,341
Non-trainable params: 0
_________________________________________________________________
```

BASELINE MODEL

Architecture

I used a 3 layer Convolutional Neural Network as my baseline model.

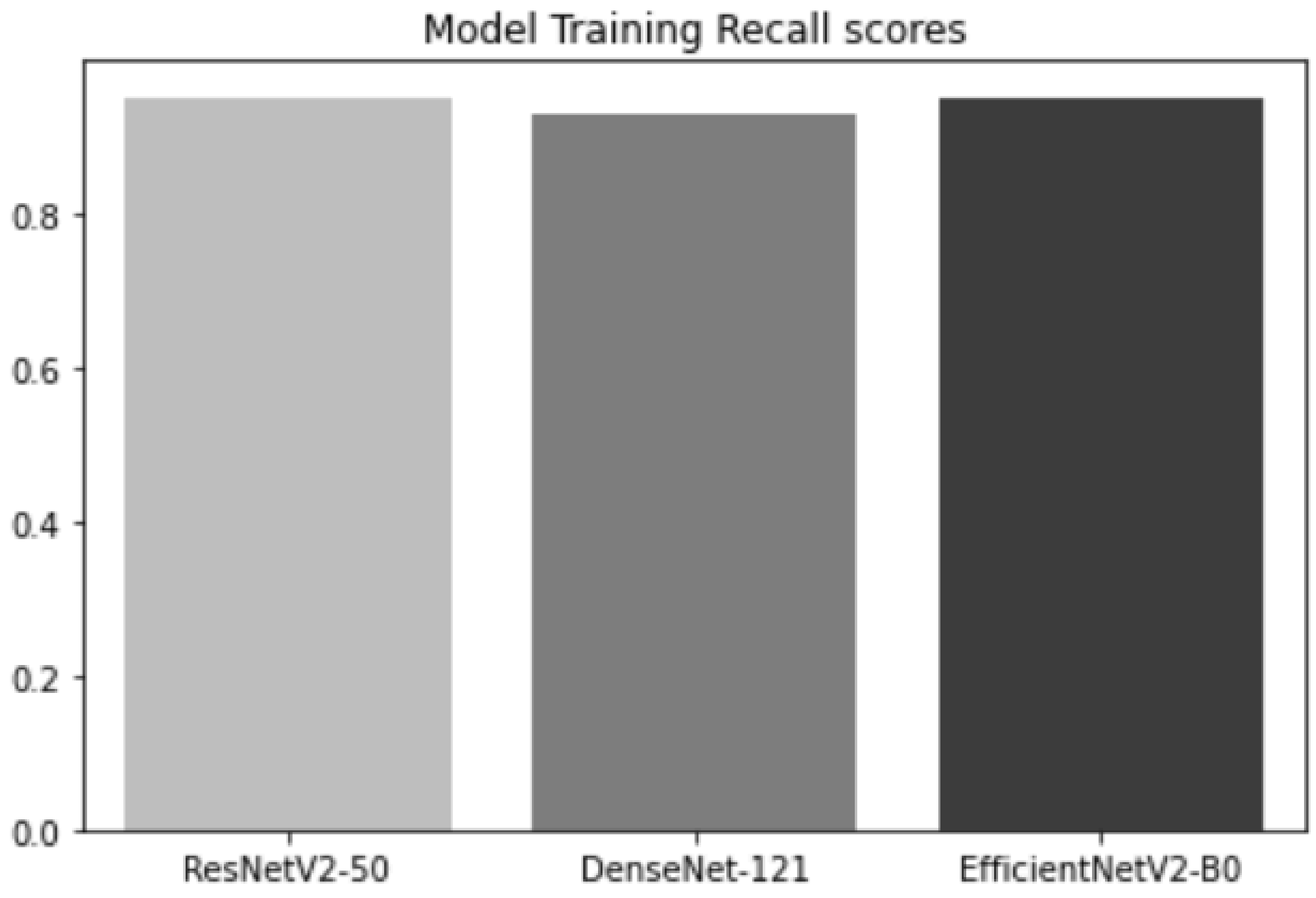


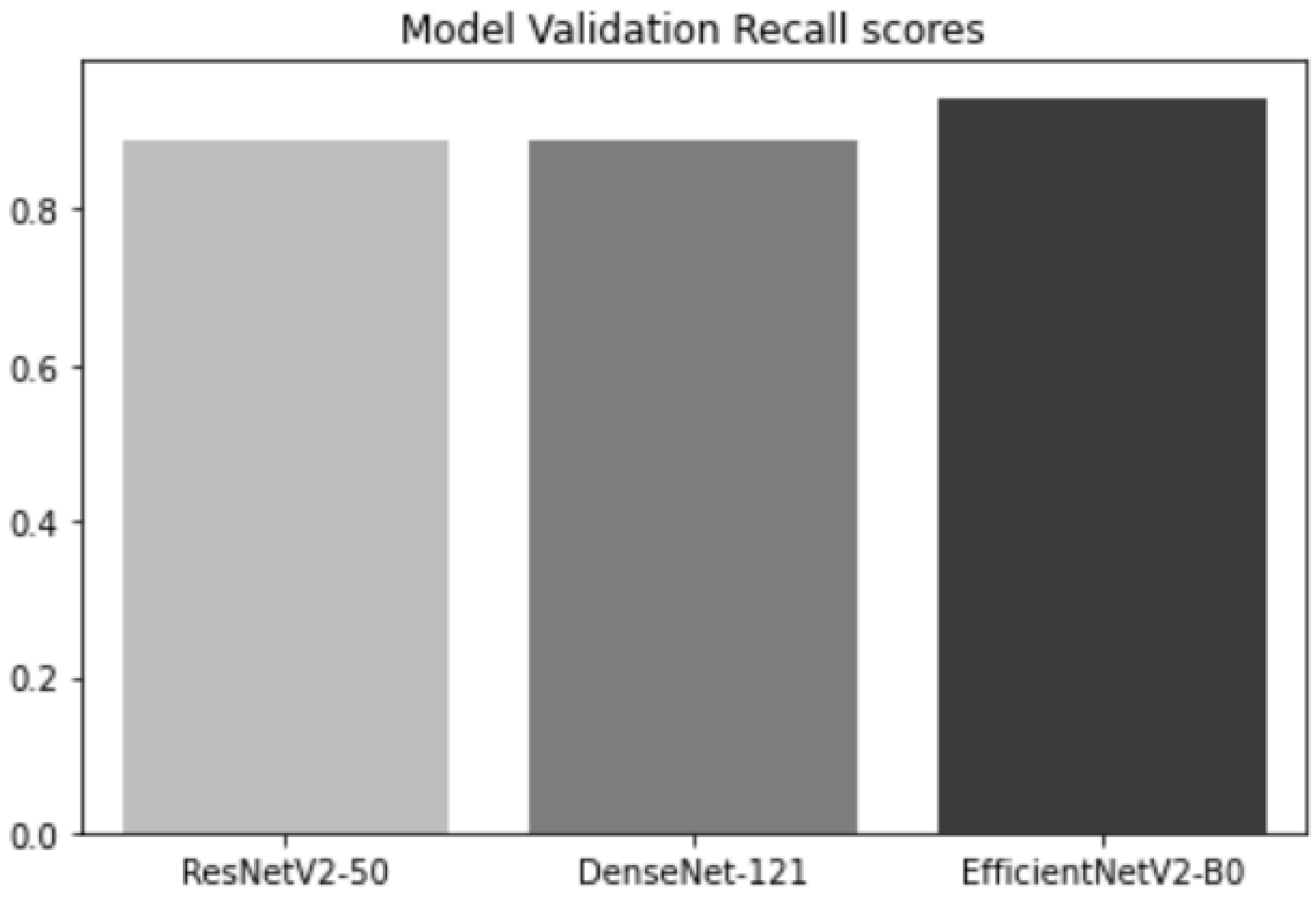
BASELINE MODEL

Learning curves

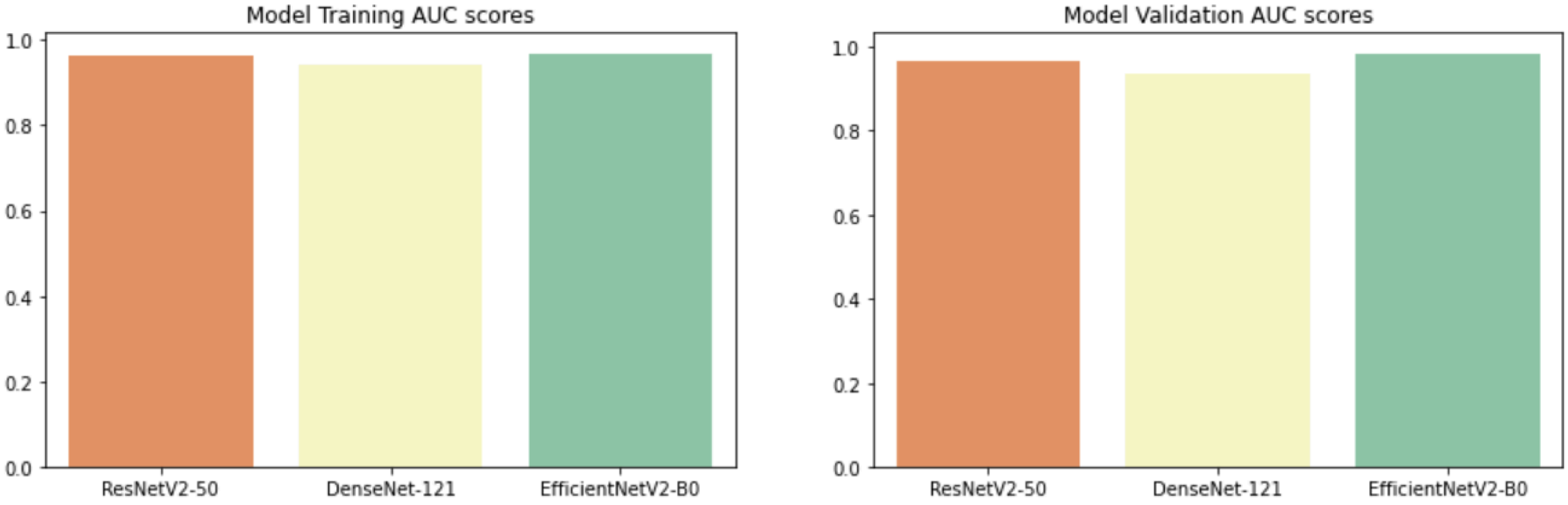
Pretrained Models' Results

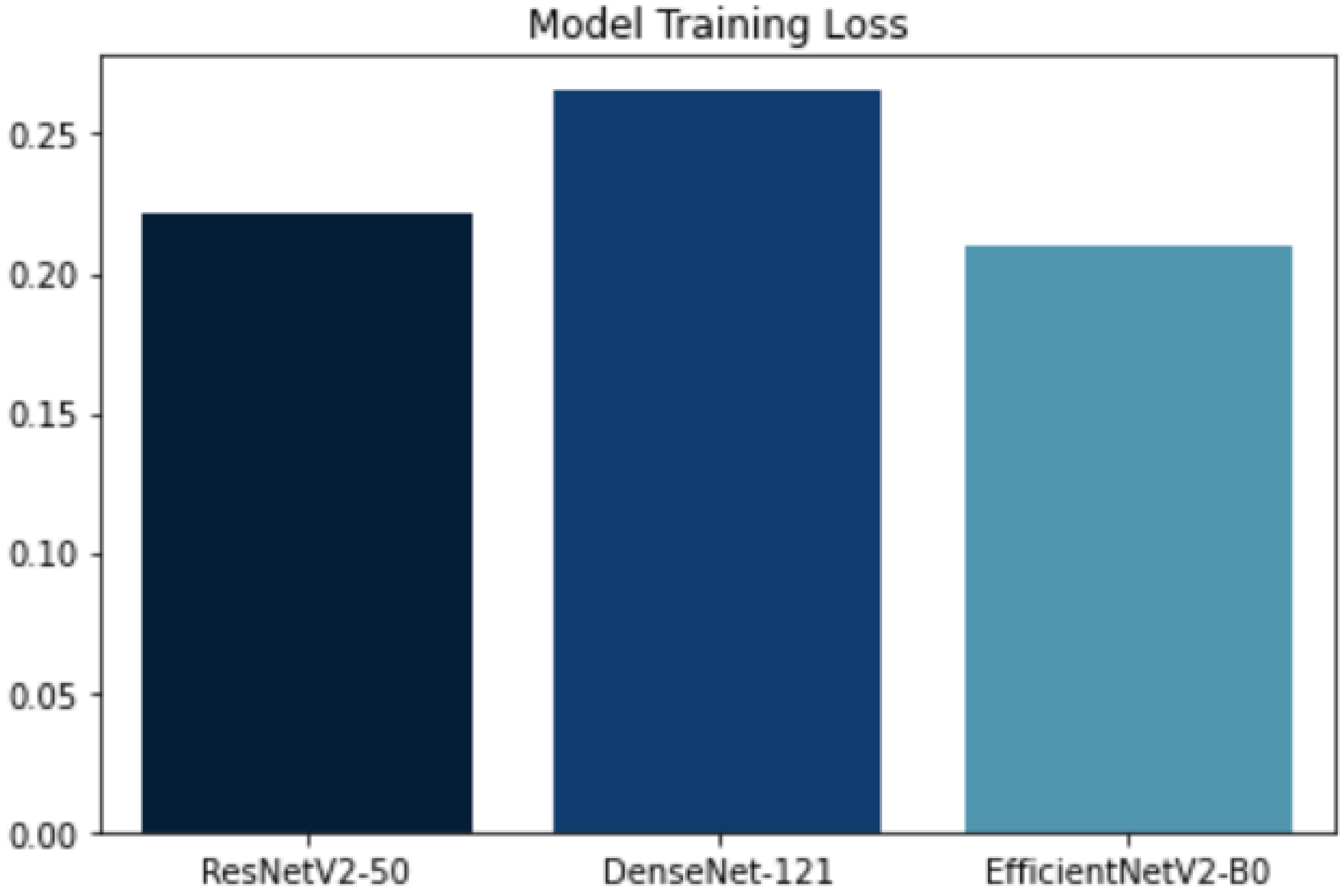
We will focus on the recall score, because it is more important for healthcare workers to be able to lessen the number of wrongly predicted, sick people.

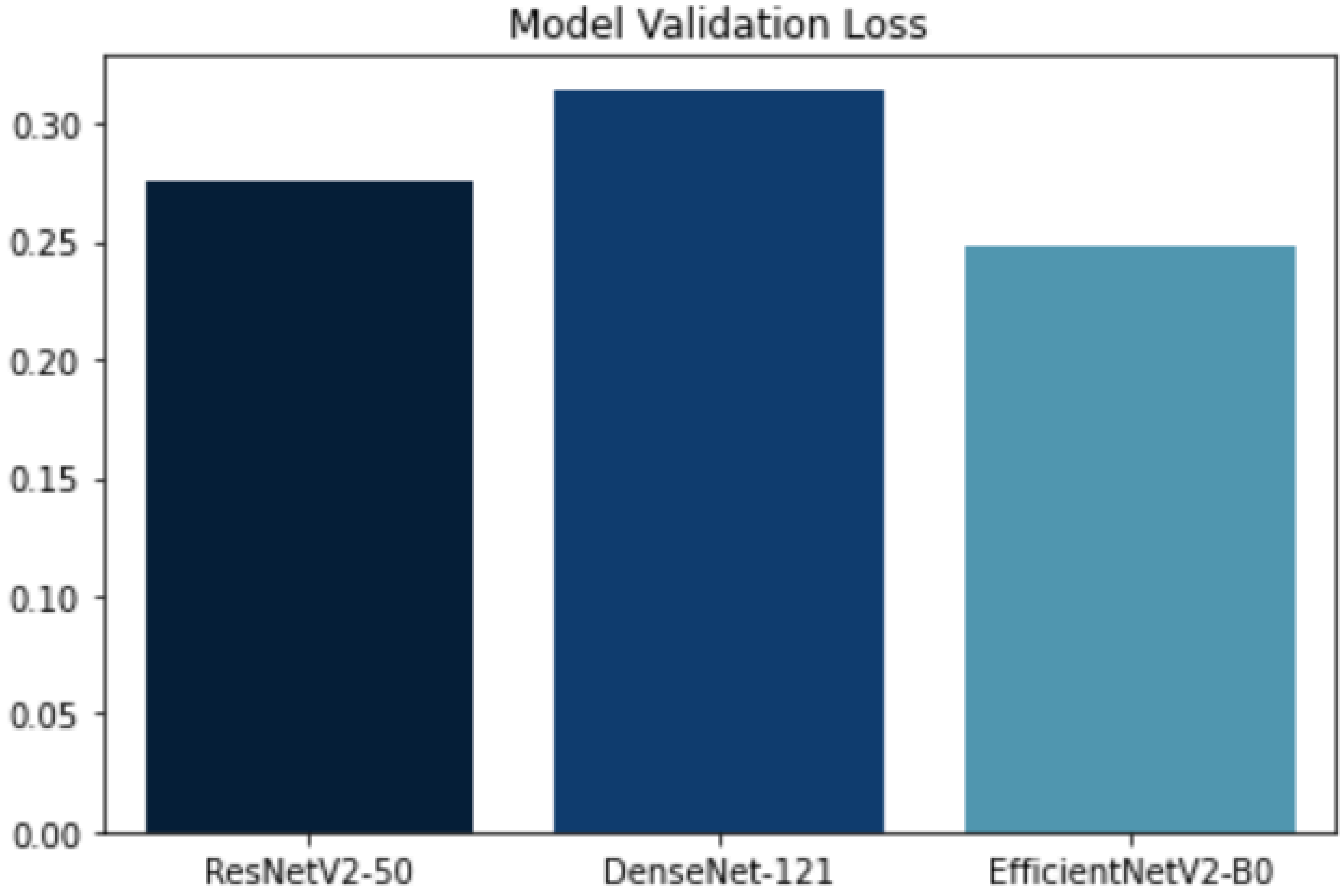




CORONAHACK


Model Training AUC scores
1.0
0.8
0.6
0.4
0.2
0.0
ResNetV2-50
DenseNet-121
EfficientNetV2-B0
Model Validation AUC scores
1.0
0.8
0.6
0.4
0.2
0.0
ResNetV2-50
DenseNet-121
EfficientNetV2-B0


Model Training Loss
0.25
0.20
0.15
0.10
0.05
0.00
ResNetV2-50
DenseNet-121
EfficientNetV2-B0


Model Validation Loss
0.30
0.25
0.20
0.15
0.10
0.05
0.00
ResNetV2-50
DenseNet-121
EfficientNetV2-B0

CHOOSING THE BEST MODEL

EfficientNetV2-B0

From the previous slides, we can see that EfficientNetV2-B0 scored the highest in terms of recall, AUC, and loss, so in the next steps, we are going to tune this pretrained model.

Google AI

IMPROVING OUR MODEL

Steps

Regularization

I added batch normalization and dropout to get a more generalized model.

Hyperparameter tuning

Keras

Flr this tuning process, I used the Hyperband tuner from Keras library, which acts to optimize the search for the right hyperparameters.

HYPERPARAMETER TUNING RESULTS

Trial summary
Hyperparameters:
units: 384
dense_activation: relu
learning_rate: 0.000382870470449823
tuner/epochs: 2
tuner/initial_epoch: 0
tuner/bracket: 3
tuner/round: 0
Score: 0.9448669254779816

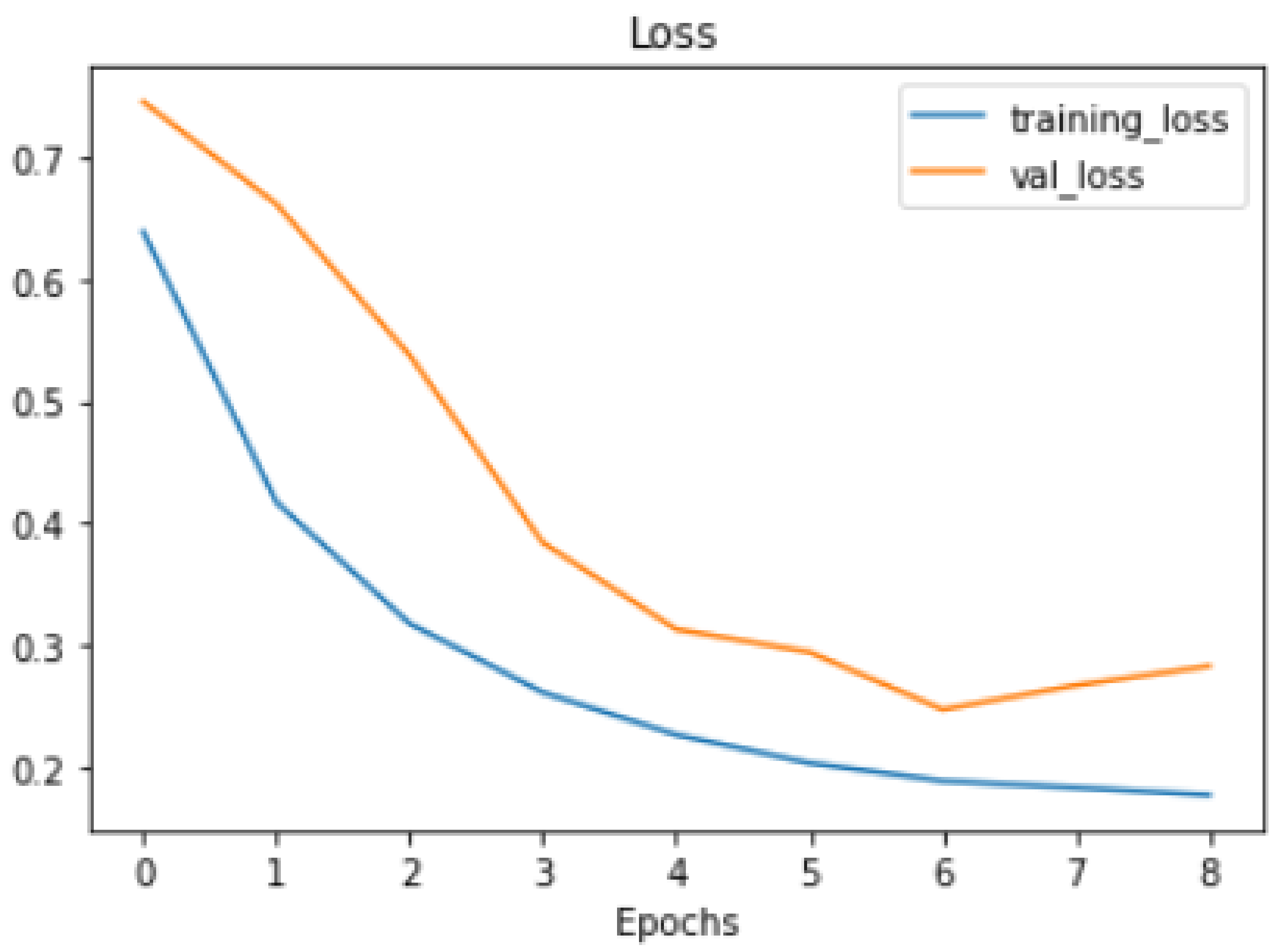
Top #1

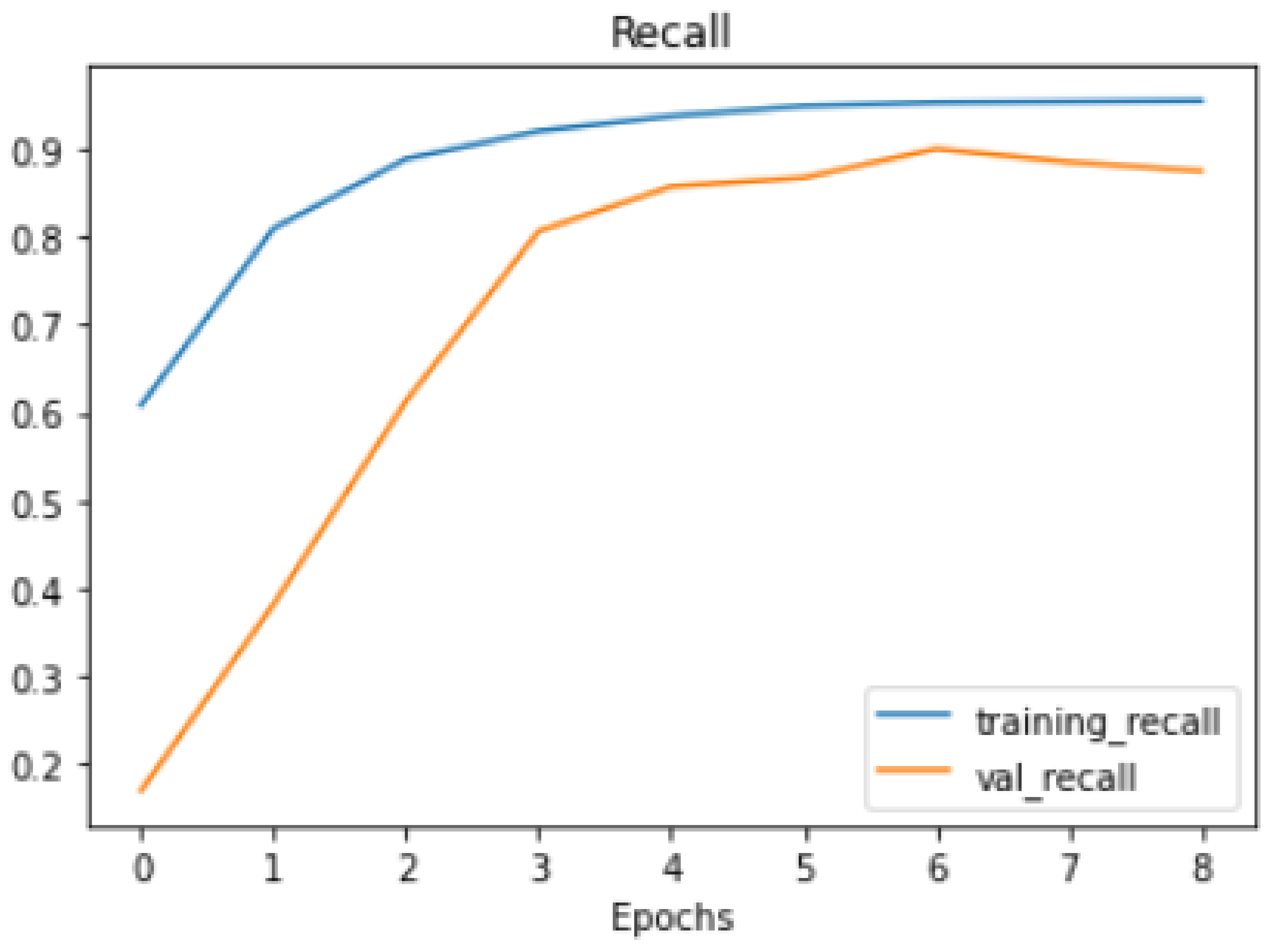
Trial summary
Hyperparameters:
units: 288
dense_activation: relu
learning_rate: 9.824542588158123e-05
tuner/epochs: 2
tuner/initial_epoch: 0
tuner/bracket: 3
tuner/round: 0
Score: 0.9442332088947296

Top #2

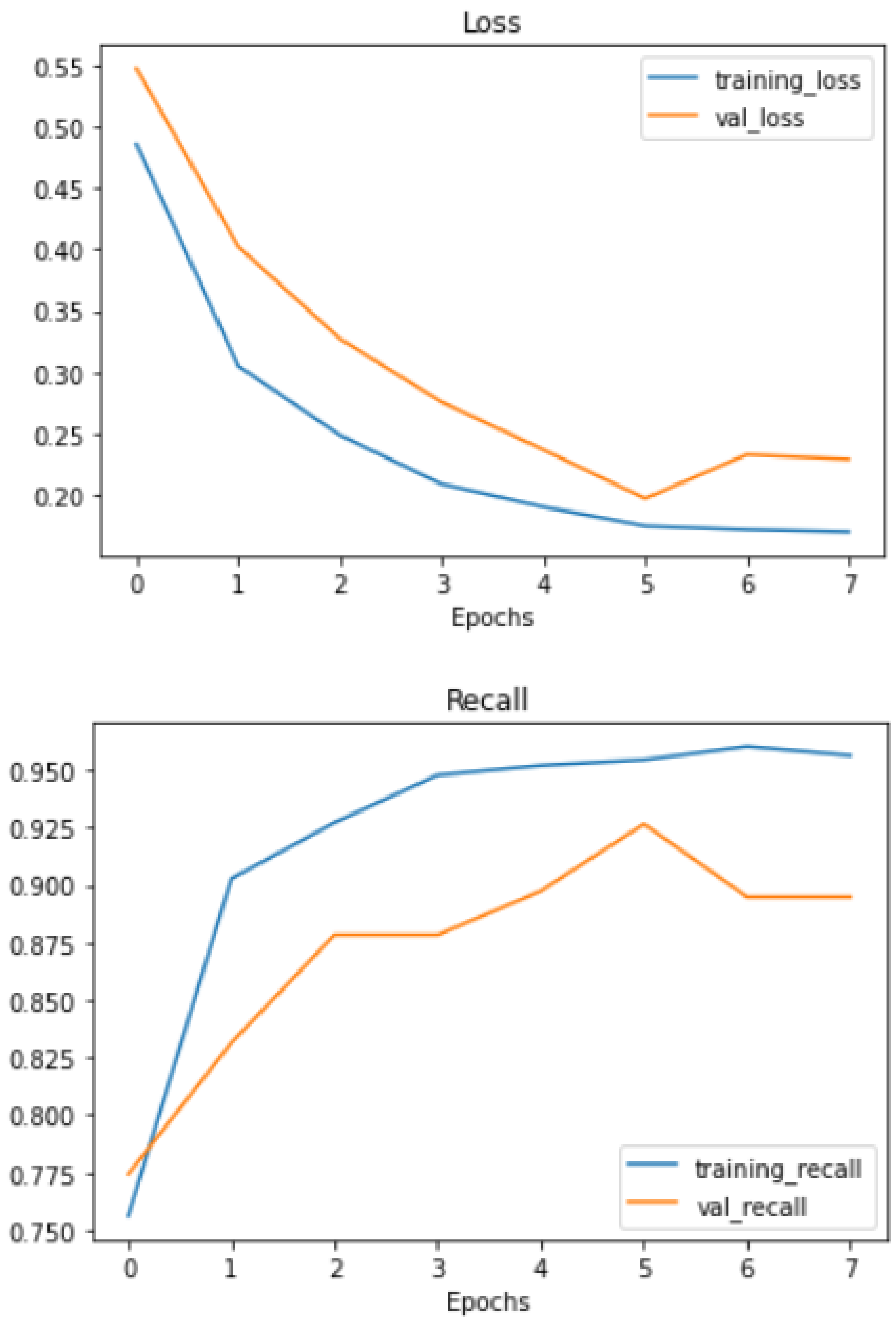
Trial summary
Hyperparameters:
units: 64
dense_activation: relu
learning_rate: 2.807830903779318e-05
tuner/epochs: 2
tuner/initial_epoch: 0
tuner/bracket: 3
tuner/round: 0
Score: 0.9391635060310364

Top #3





For my tuned model, I decided to take the average of parameters from my initial model and the hyperparameter tuning results. Here are the learning curves of this model.



For comparison, here are the learning curves for my model prior to performing hyperparameter tuning.

CORONAHACK

CONFUSION MATRIX

	0	1
0	176	58
1	38	352

	0	1
0	166	68
1	6	384

Baseline model

We got 352 True Positives and 38 False Negatives.

Tuned model

We got 384 True Positives and 6 False Negatives

CLASSIFICATION REPORT

Baseline model

	precision	recall	f1-score	support
0	0.82	0.75	0.79	234
1	0.86	0.90	0.88	390
accuracy			0.85	624
macro avg	0.84	0.83	0.83	624
weighted avg	0.84	0.85	0.84	624

CLASSIFICATION REPORT

Tuned model

	precision	recall	f1-score	support
0	0.97	0.71	0.82	234
1	0.85	0.98	0.91	390
accuracy			0.88	624
macro avg	0.91	0.85	0.86	624
weighted avg	0.89	0.88	0.88	624

FINAL REMARKS

Model of choice

Since my tuned model performed the best, I chose that model to be the final one.

CONCLUSION AND RECOMMEN-DATION

Conclusion and Recommendation

CONCLUSION

After testing several models, I concluded that EfficientNetV2-B0 performed the best and achieved a weighted recall score of ~88% and an AUC score of ~94.7% on unseen data, which is an improvement from the baseline model. Our AUC score improved by ~5%, with an improvement of weighted average recall score of ~3% There was also an improvement of about 30 less cases of false negatives and 30 more cases of true positives!

Conclusion and Recommendation

RECOMMEN-DATION

The work I have done is far from perfect; there are still many improvements that can be made in the future, such as:

- Spend more time on the hyperparameter tuning process
- Using the ensemble method to create one optimal model

THANK YOU