covid-audio

A Three-Fold Machine Learning Approach to Detecting COVID-19 from Audio Data

Nikhil Kumar | Vishal Mittal

28 November 2020

Agenda

- Recap
  - Background
  - Data Collection & Preprocessing
  - Approach
  - Feature Extraction
- Training
  - SVM
  - CNN
  - LSTM
- Challenges & Future Work

Recap

- Background
- Data Collection & Preprocessing
- Approach
- Feature Extraction

Background

Given audio samples, can we predict the presence of COVID-19?

Raw audio samples { Breath / Cough → COVID

Current approaches:
- X-ray images — Invasive
- Thermal images — Too general (detects fever — could be anything)

Audio data:
- Has generated interest – Cambridge COVID Sounds, IISc Coswara
- Using traditional ML with handcrafted features
- Using neural network black box approaches

How is Our Work Different?

Three-fold approach:

- Traditional ML: Using handcrafted features
- ConvNets: Using spectrograms
- Recurrent models: Using instantaneous features → Novel, has not been applied widely to this task

Data Collection & Preprocessing

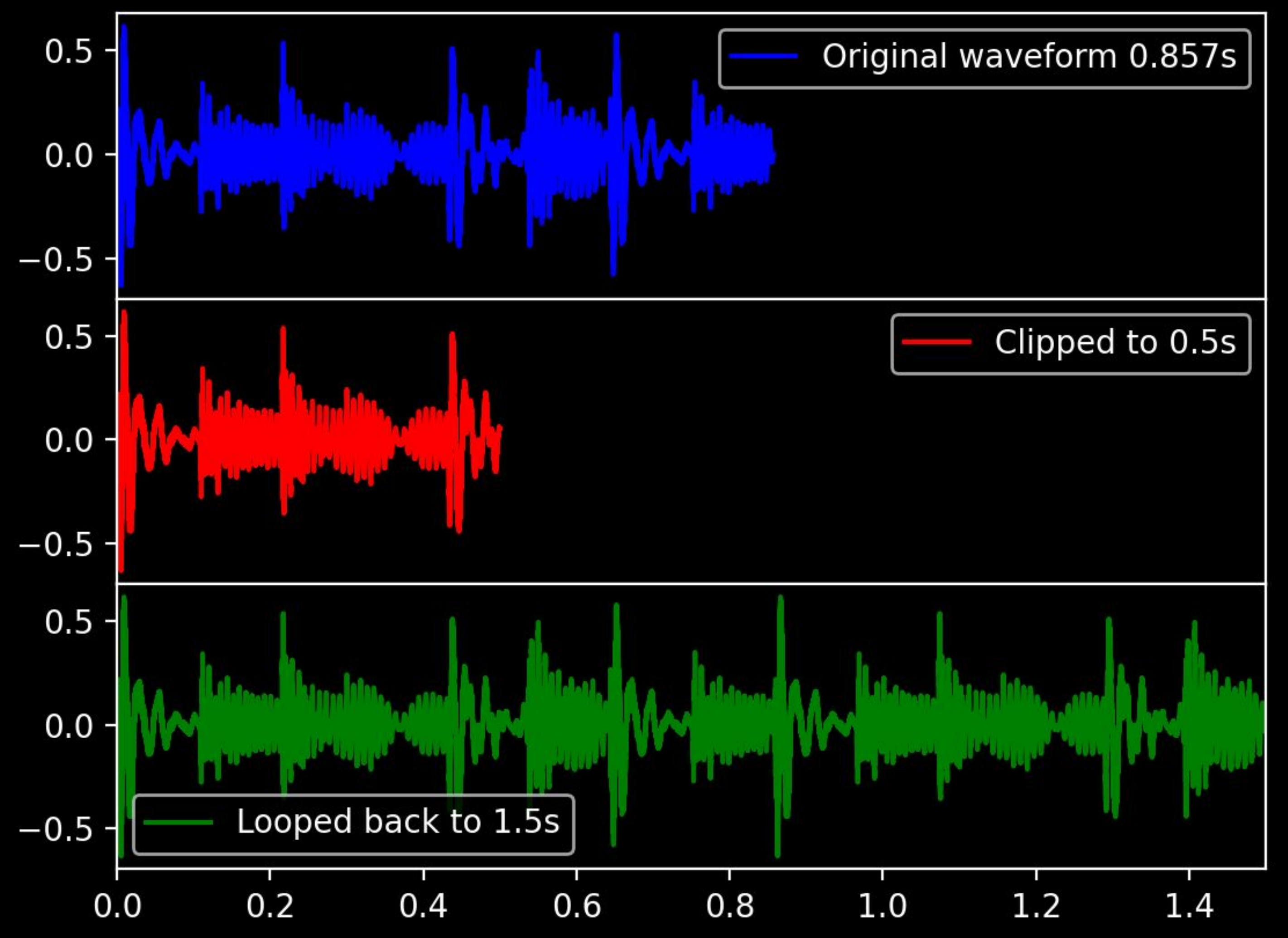
- Obtained data from University of Cambridge
  - Crowdsourced breath and cough samples
  - 1134 breath + 1135 cough
    - `asthma`
    - `covid`
    - `normal`

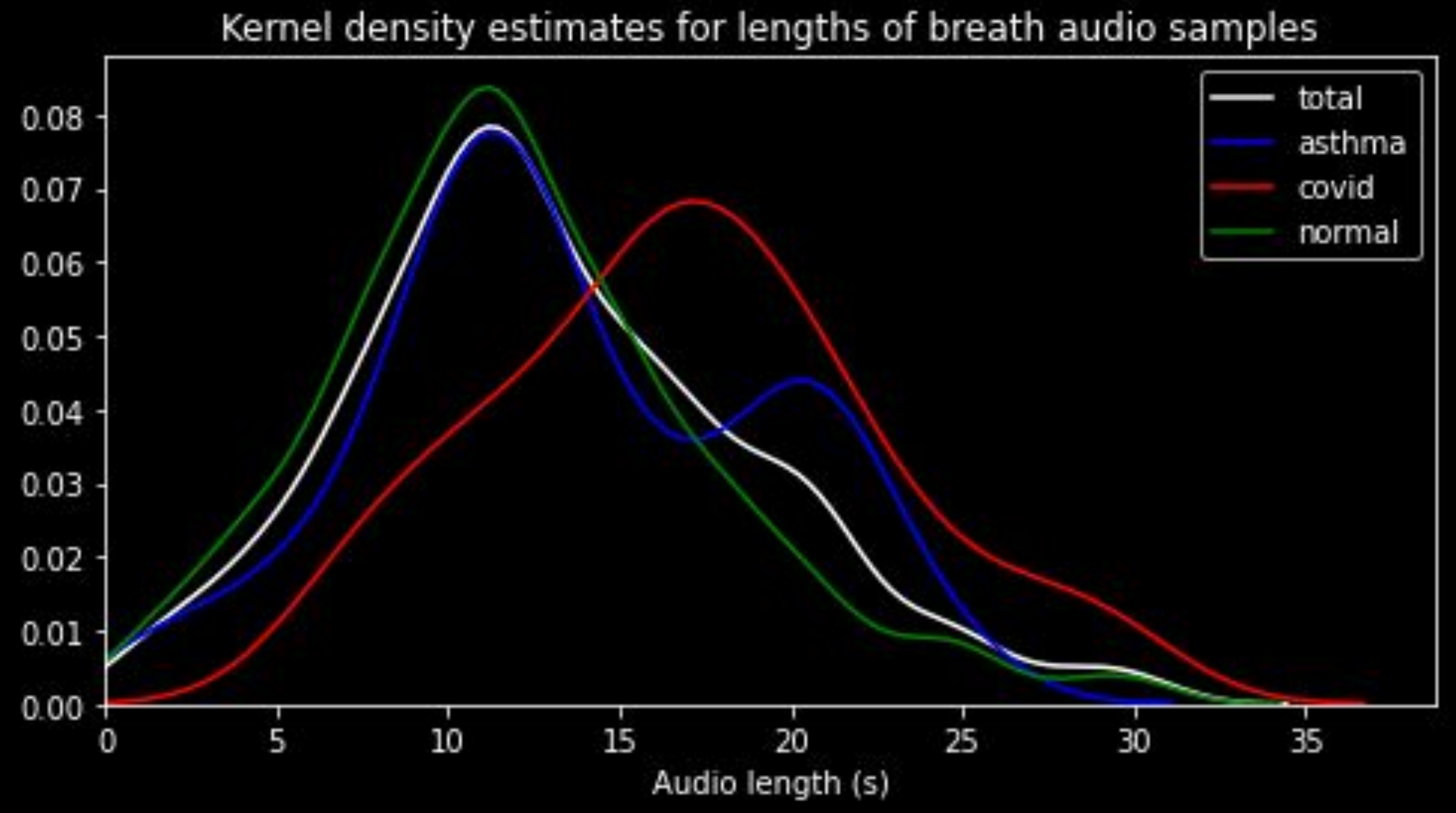
    15:15:70 ratio
  - Split data into train, validation and test
    - 80:10:10 split
    - Ensured class ratios / distribution maintained on performing the split

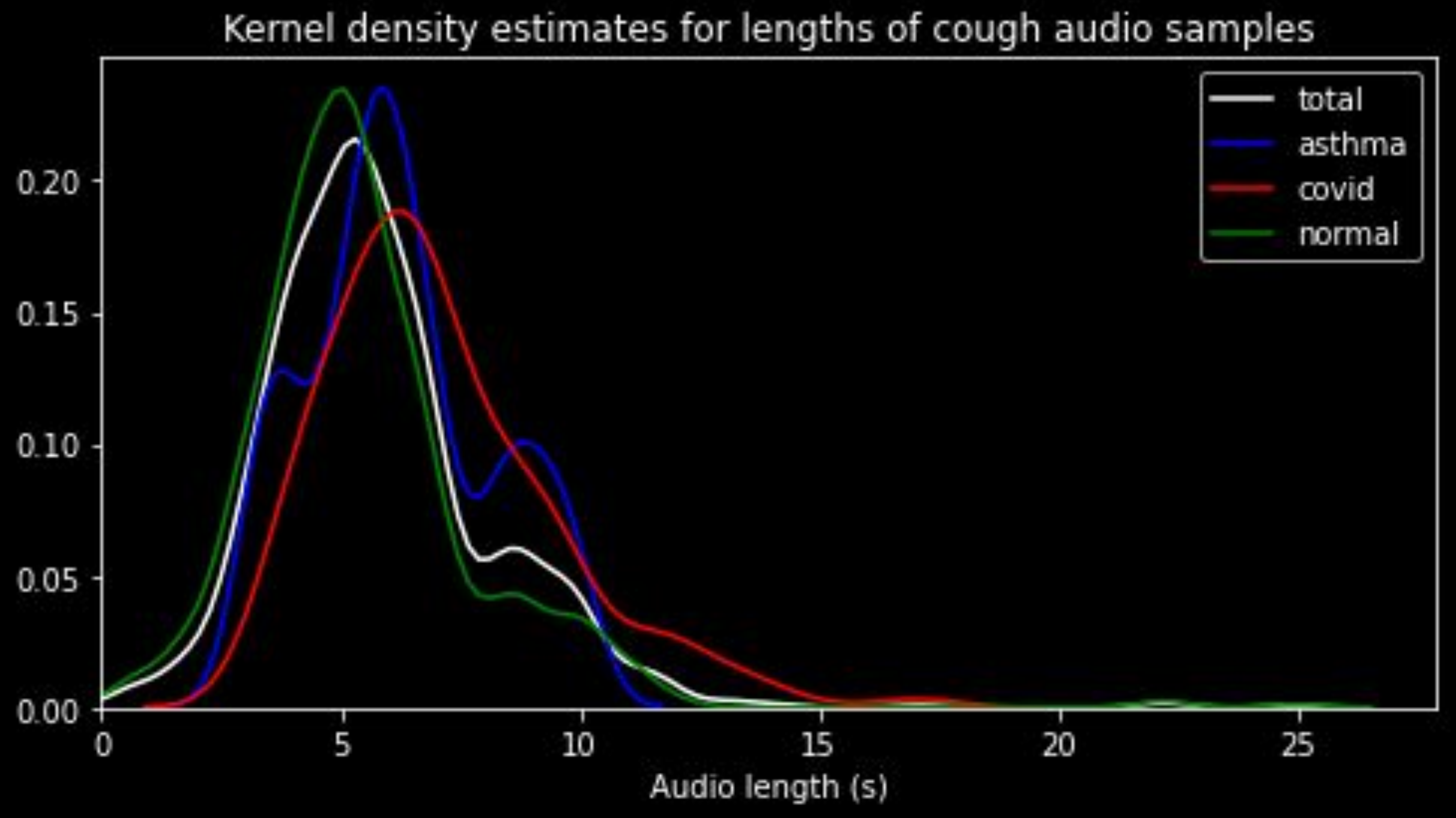
Data Collection & Preprocessing

- Loopback & clipping
  - Threshold as 95th percentile length
  - Tradeoff – Information loss vs processing time
  - Uniform length – loop is longer, clip if shorter

Audio obtained under Creative Commons license
from user casualsamples, freesound.org.

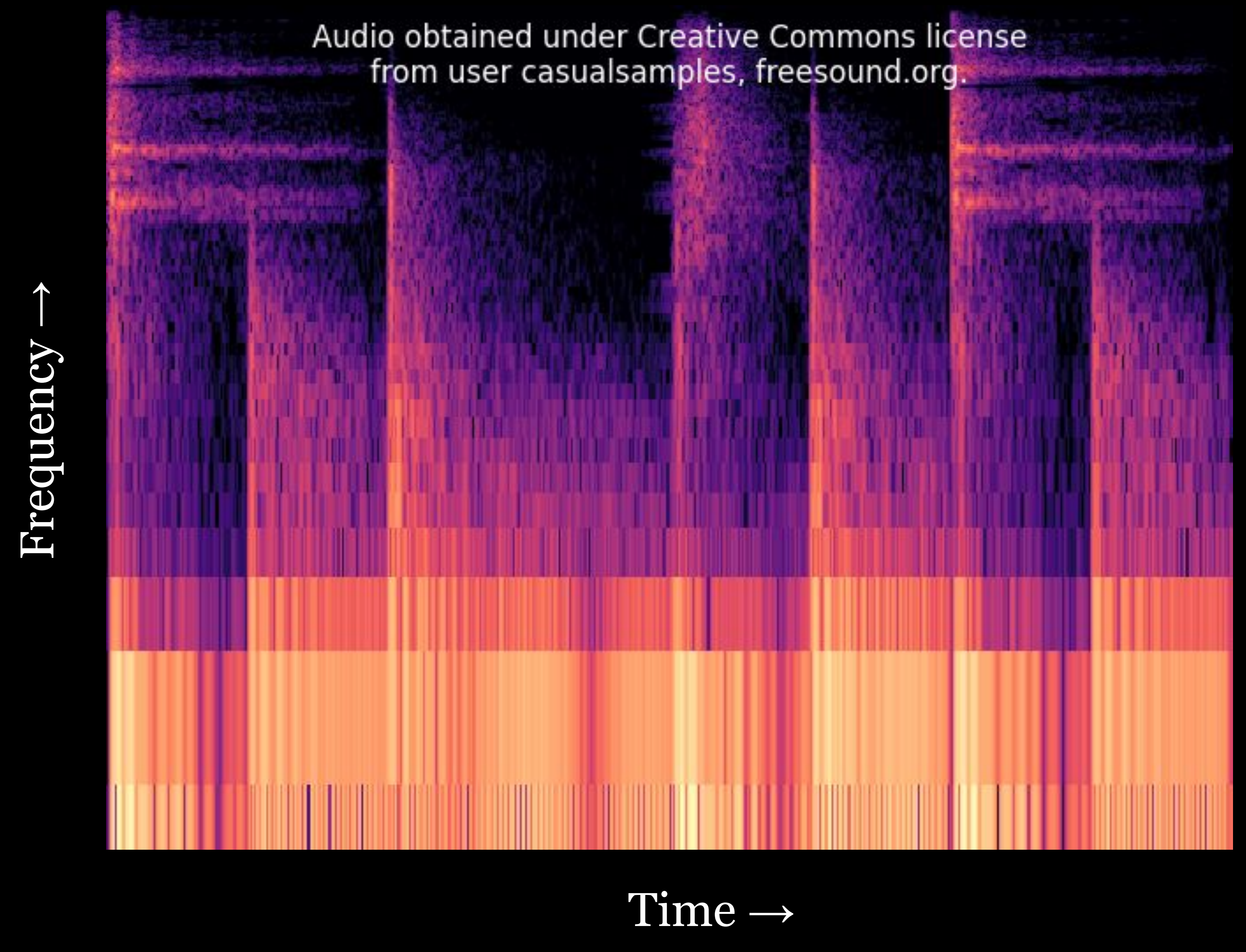


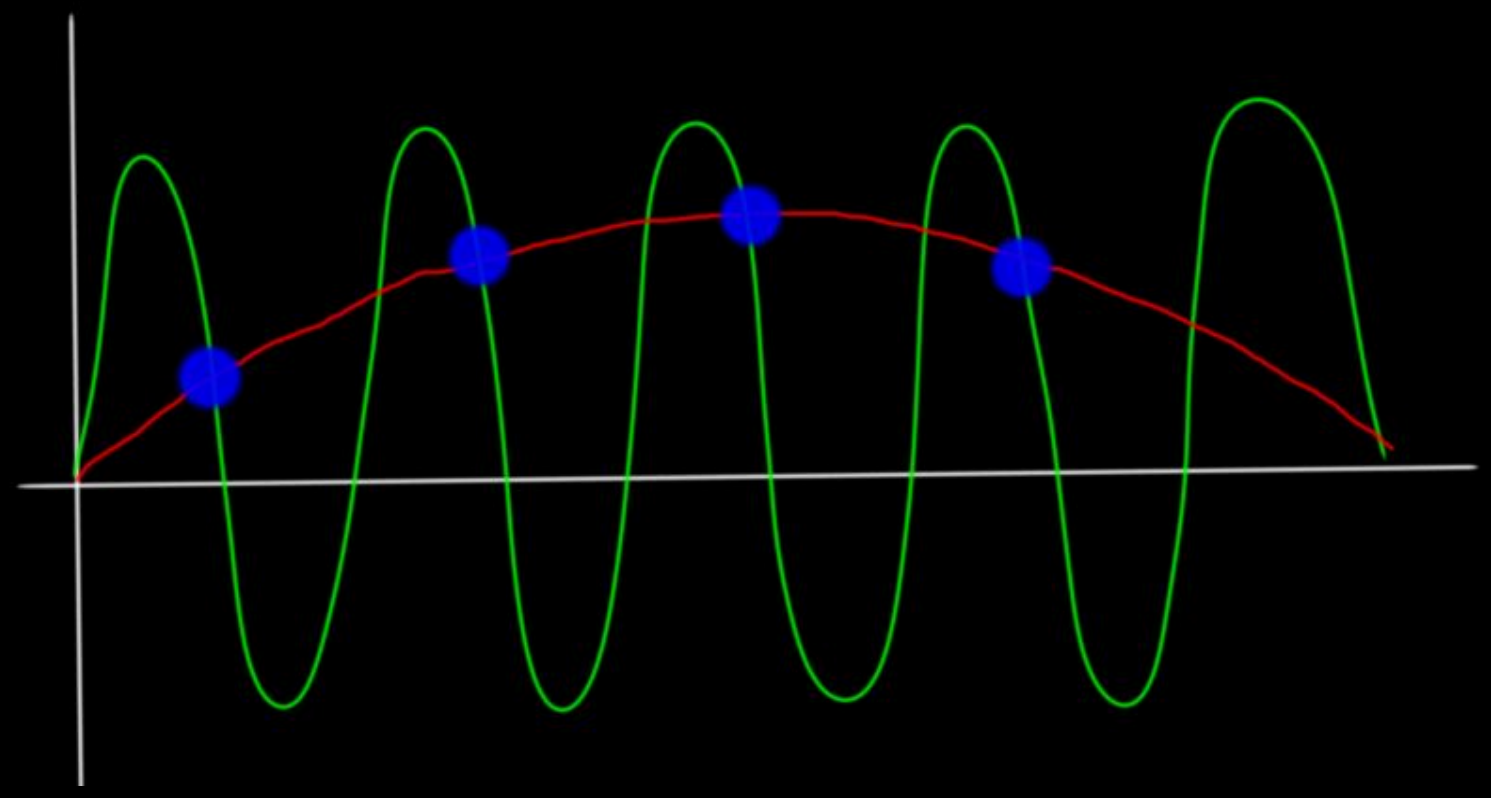




Data Collection & Preprocessing

- Spectrogram generation
  - Waveform → Windowing → STFT → Spectrogram
  - Sampling rate = 16 kHz
    - Nyquist-Shannon theorem
    - Sampling rate ≥ 2 × Max frequency
    - Same as KDD paper, 8 kHz max observed
  - Scales
    - Frequency: Mel scale (logarithmic) – make visible
    - Magnitude: dB (logarithmic)





Aliasing

Approach

- SVM
  - 108 handcrafted aggregate instantaneous features
  - Tabular data
- ConvNet model
  - Uses spectrograms
- LSTM
  - 18 time series
  - Instantaneous features

- Instantaneous features
  - Calculated over windows of single audio sample
  - Generates time series
  - `num_timesteps` values for an audio sample
- Aggregate instantaneous features
  - Statistic over instantaneous feature, summarizes
  - 1 value for an audio sample
- Global features
  - Calculated over audio sample as a whole

$$\text{num\_timesteps} = \left\lfloor \frac{\text{audio\_length} + 2 \times \text{pad\_length} - \text{frame\_length}}{\text{hop\_length}} \right\rfloor + 1$$

Feature Extraction

18 instantaneous features × 6 aggregation functions = 108 features

rmse		mean
zcr		median
sc	×	rms
sr		max
sb		min
mfcc1 – mfcc13		rewm

LSTM Models – Architecture

Breath

Layer (type)	Output Shape	Param #
lstm_17 (LSTM)	(None, 6073, 32)	6528
lstm_18 (LSTM)	(None, 32)	8320
dense_19 (Dense)	(None, 32)	1056
dense_20 (Dense)	(None, 1)	33

Total params: 15,937
Trainable params: 15,937
Non-trainable params: 0

input_shape = (6073, 18)
32 LSTM+ 32 LSTM, tanh activation
32 Dense, relu activation
1 Dense, sigmoid activation
Loss: binary_crossentropy
Optimizer: Adam

Cough

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 2481, 64)	21248
lstm_5 (LSTM)	(None, 64)	33024
dense_4 (Dense)	(None, 64)	4160
dense_5 (Dense)	(None, 1)	65

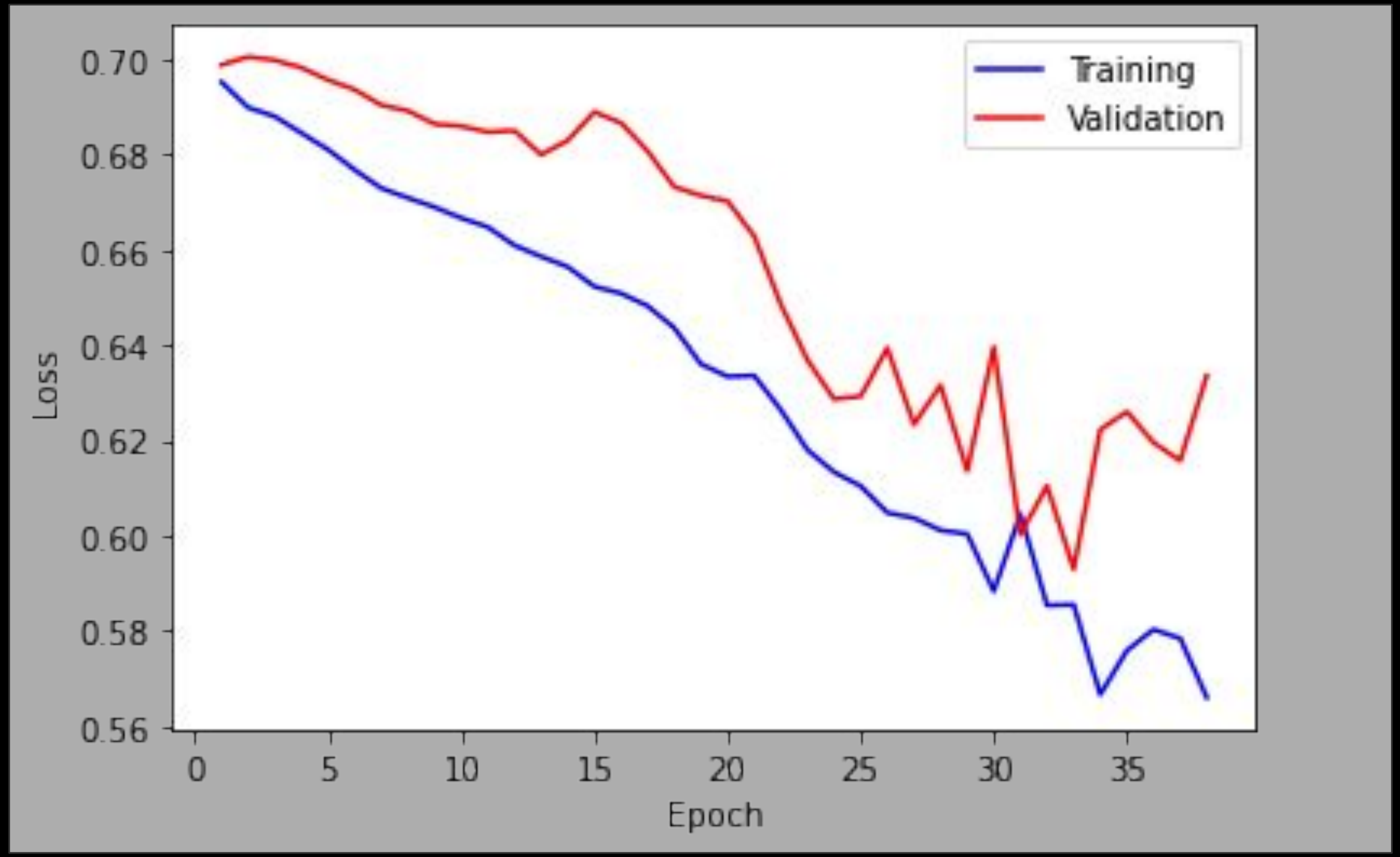
Total params: 58,497
Trainable params: 58,497
Non-trainable params: 0

input_shape = (2481, 18)
64 LSTM + 64 LSTM, tanh activation
64 Dense, relu activation
1 Dense, sigmoid activation
Loss: binary_crossentropy
Optimizer: Adam

LSTM Models – Results (after balancing)

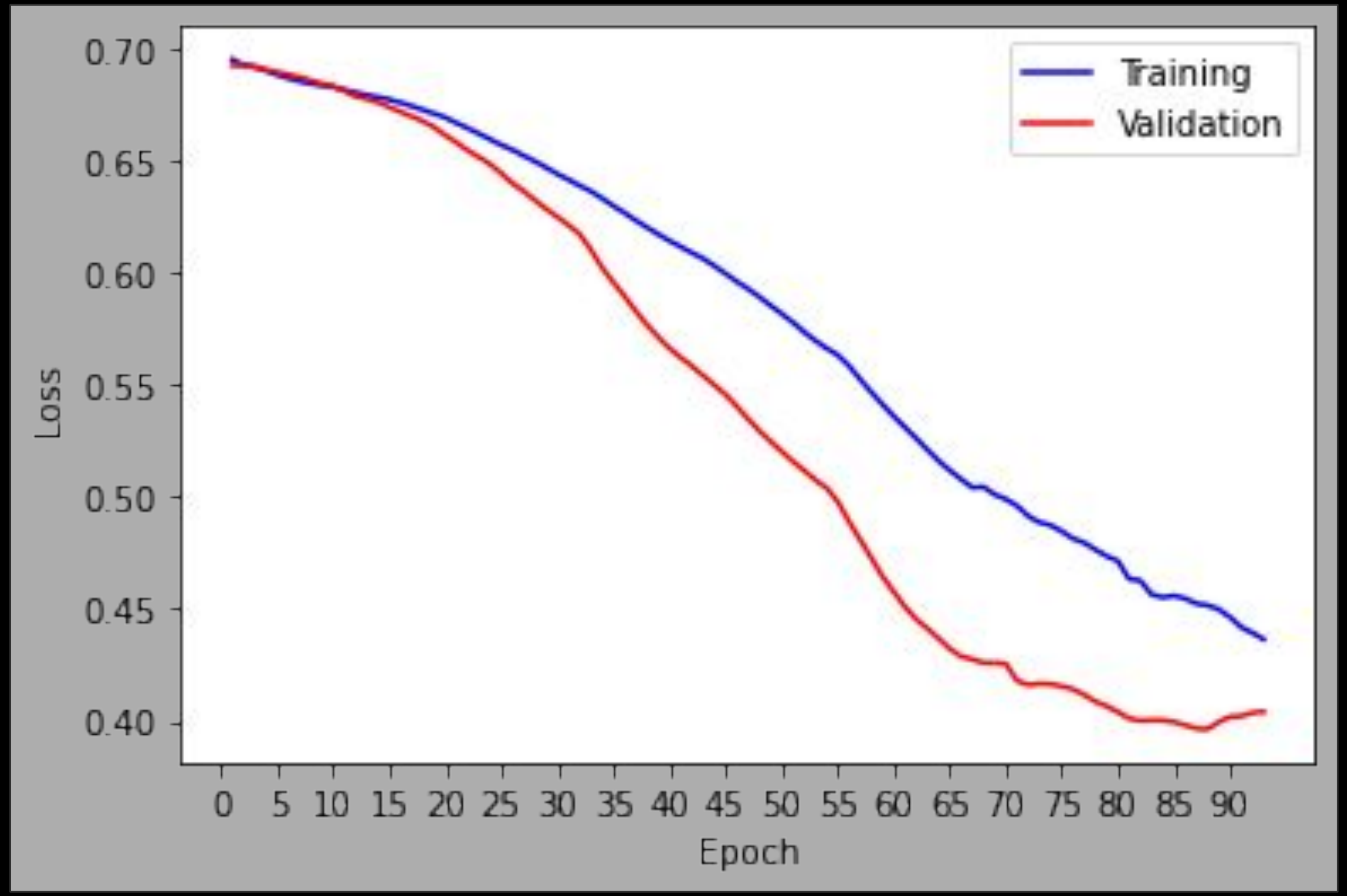
Breath

Metric	Training	Validation
Accuracy	0.7190	0.6765
Precision	0.7778	0.8000
Recall	0.6131	0.4706
AUC	0.7709	0.7232



Cough

Metric	Training	Validation
Accuracy	0.8169	0.7778
Precision	0.7711	0.7083
Recall	0.9014	0.9444
AUC	0.8781	0.9043



LSTM Models – Discussion

- Suffers heavily from class imbalance
  - Poor metrics on imbalanced data
  - Above metrics were generated after undersampling the majority class to 50:50
- Unless class imbalance is resolved, model is incentivized to predict all as `normal` to reduce the loss
- How to solve?
  - Augment
  - Techniques like SMOTE
- Same issue with CNN

Training

ML Models - SVM

- Binary Classification: Covid vs Normal

- Number of samples in training:
  - Covid: 137
  - Normal: 626
  - (Total: 763)

- Applied PCA on feature set keeping first 20 principal components (0.95 variance in data)

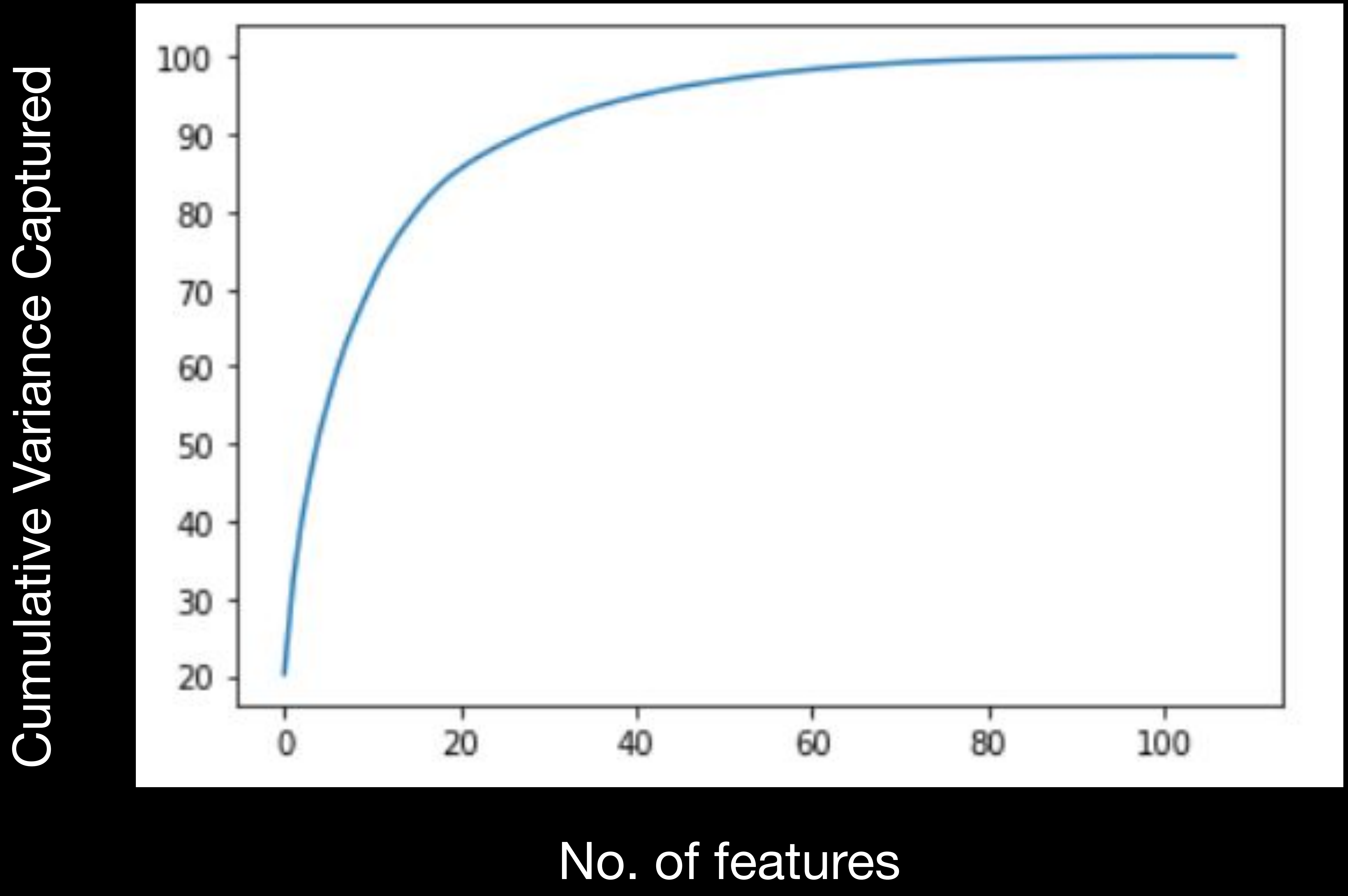
Model: SVM Classifier with 'rbf' kernel

Results (On testing data - 35 Covid samples, 157 normal samples):

Accuracy: 0.974
Precision: 0.89
Recall: 0.97
F1-score: 0.93

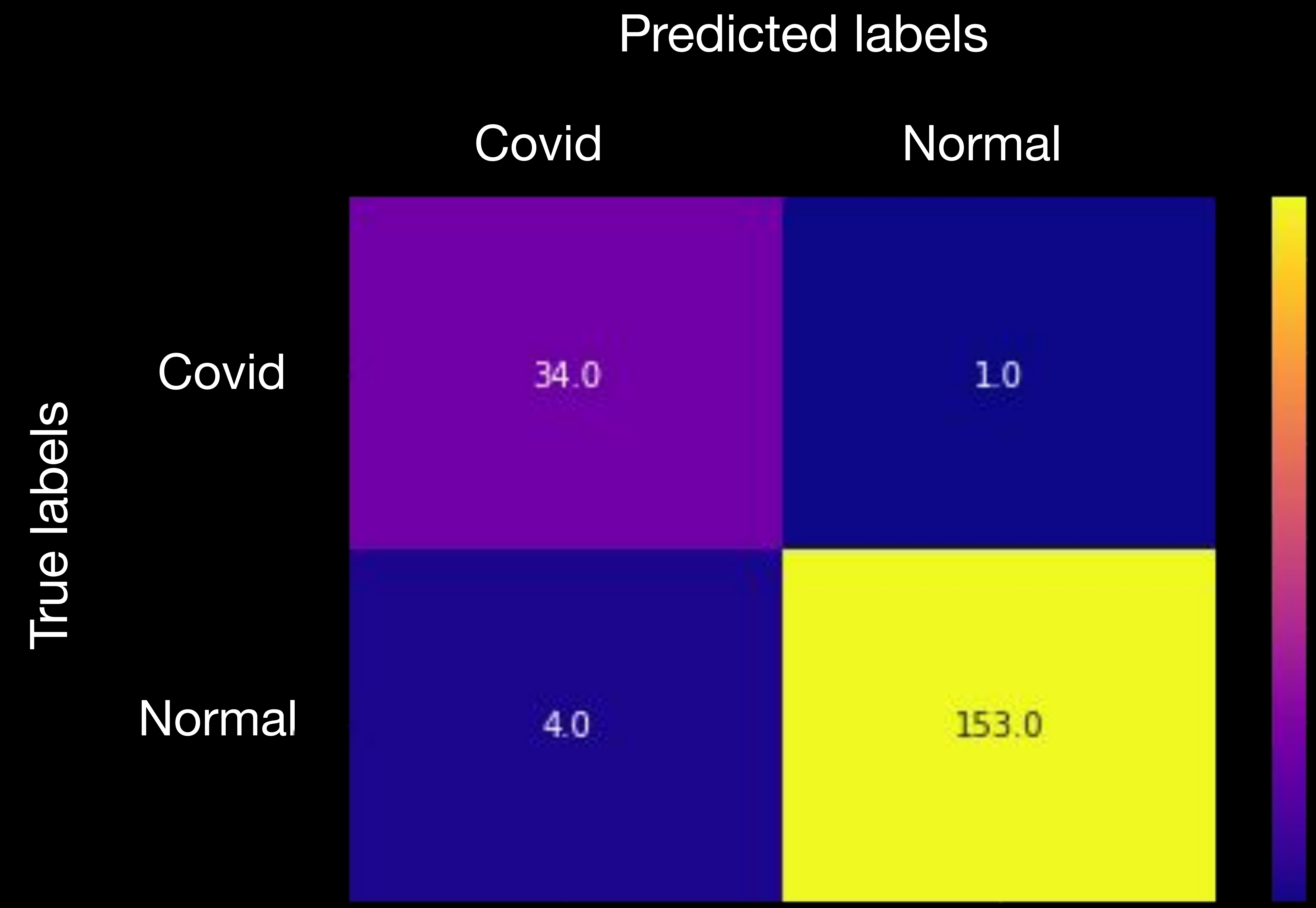
Training

Cumulative Variance Plot for PCA



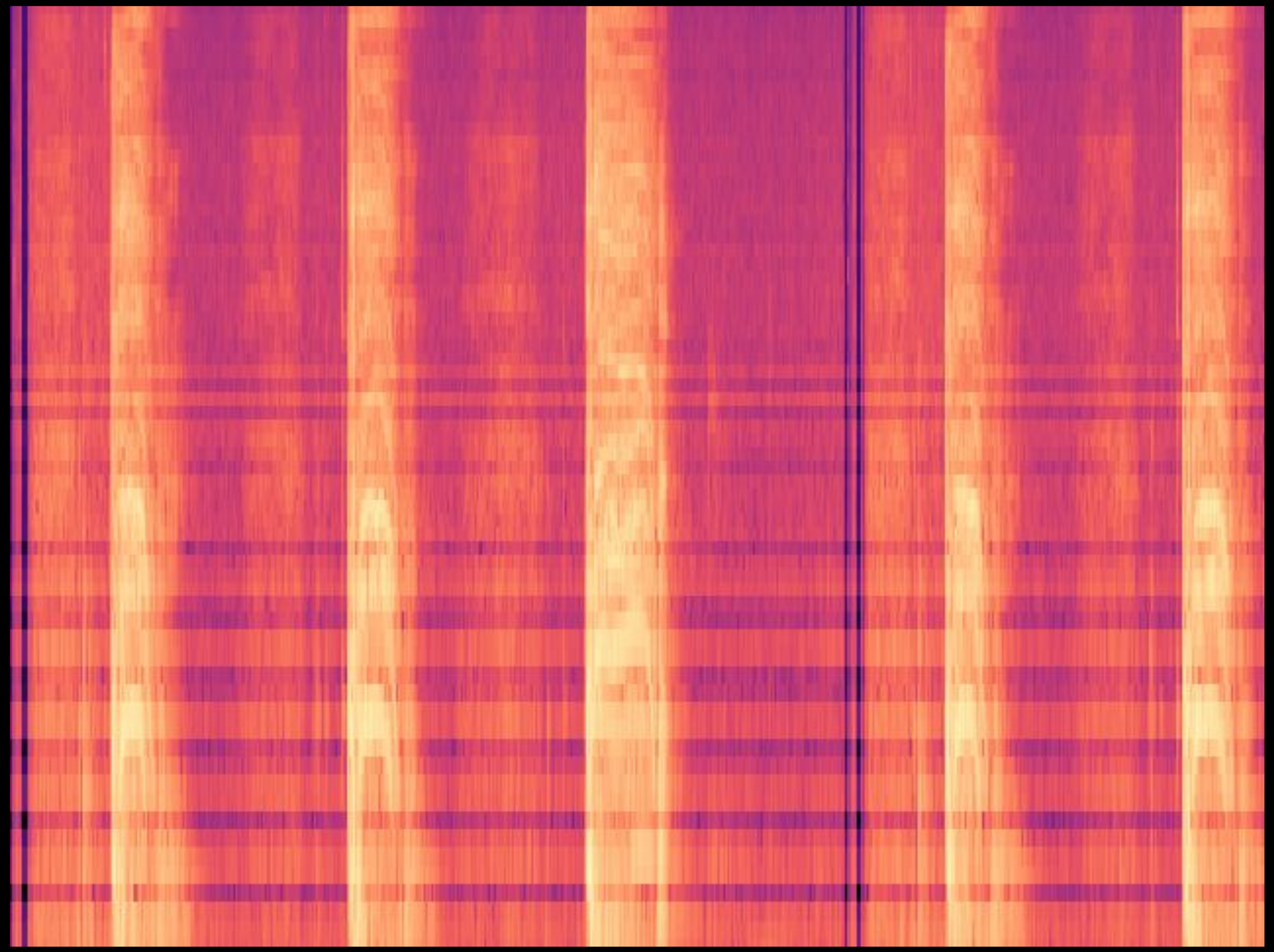
Testing

Confusion Matrix



CNN Model

CNN classifier on Mel Spectograms



Mel spectrogram of a Covid sample

CNN Model

Model Architecture

```
model = Sequential()
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(224,224,3)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))
```

CNN Model

Results

- Number of training samples:
  - Covid: 137
  - Normal: 631
  - (Total: 768)

Performed Data Augmentation using width_shift_range for proper time sampling

- Validation Accuracy: 0.82

Testing: Currently model is overfitting due to imbalanced classes and classifying all images as normal (Number of samples with class normal are around 5 times the number of samples with class covid)

Work completed

- RNN/LSTM on audio data to capture the sequential time-series relationship
- Deploying COVID detection through X-ray and audio work on a website where a user can upload the sample for testing

Future Work

- Overcoming the overfitting of CNN model through over-sampling and data augmentation and training on GPU

COVID Detection through X-ray Images

Nikhil Kumar | Vishal Mittal | 28 November 2020

Dataset Preparation

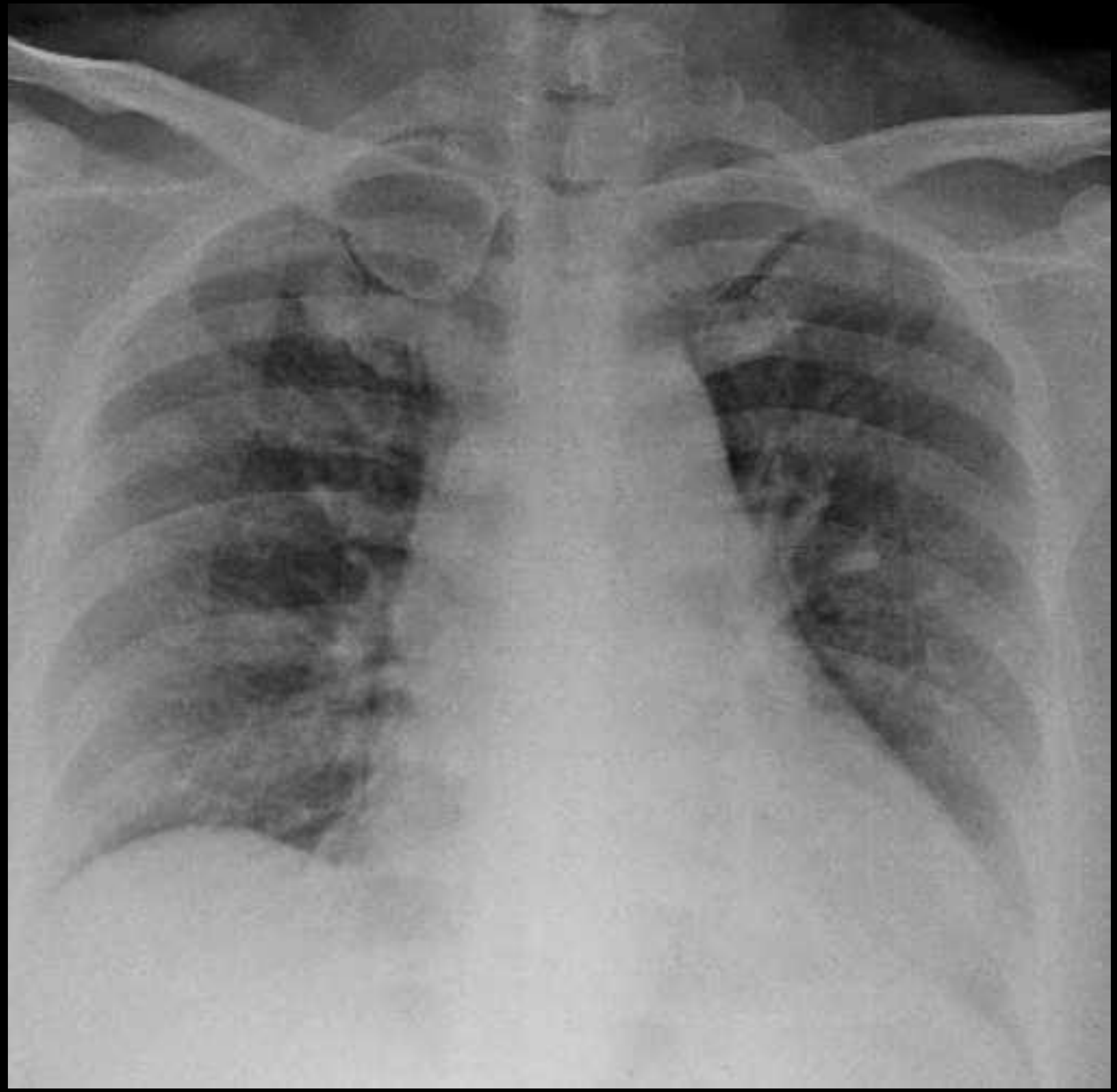
Sampling of images from through 2 sources:

COVID: https://github.com/ieee8023/covid-chestxray-dataset (180 samples)

Normal: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia (Sampled 180 out of 1341 images)

Data Augmentation

- horizontal_flip
- zooming
- shearing



Chest X-ray of a COVID positive person

Training

CNN Model - Results

- Binary Classification: Covid vs Normal

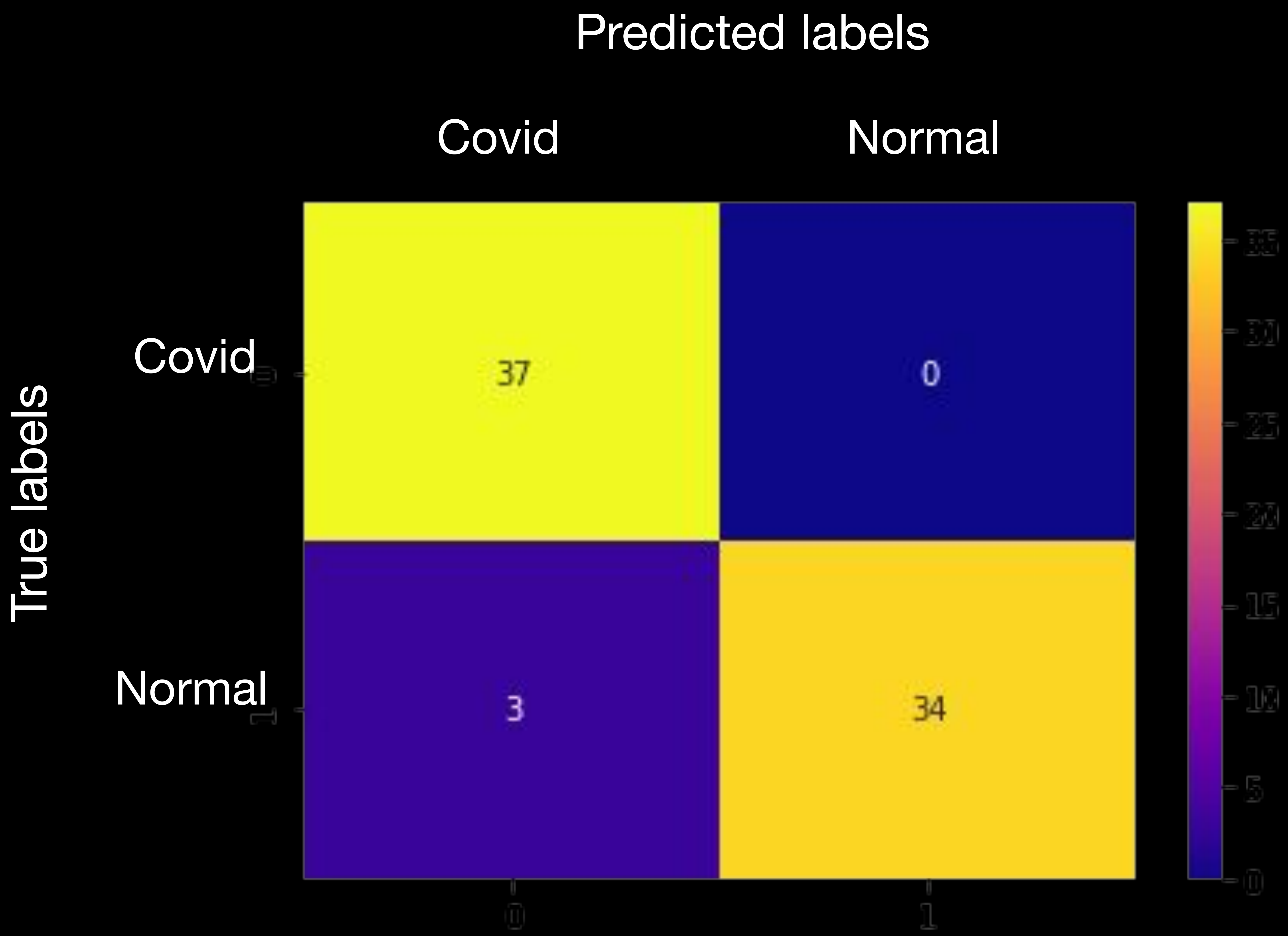
- Number of samples:
  - Training: Covid: 125, Normal: 125 (70%)
  - Validation: Covid: 18, Normal: 18 (10%)
  - Testing: Covid: 37, Normal: 37 (20%)

Results:

Accuracy: 0.959
Precision: 0.925
Recall: 1.00
F1-score: 0.961

Testing

Confusion Matrix



Thank You!