Investigating Relationship Between Covid-19 and Spectrograms of Coughing Acoustics Through the Use of Convolutional Neural Network

Research Question: To What Extent Can Convolutional Neural Network and Spectrograms be Used to Diagnose Covid-19 Patients?

Qianghao Wu

Word Count: 3946

CS EE World
https://cseeworld.wixsite.com/home
May 2022
26/34 (A)

Submitter Info:
Going to UWaterloo CS + WLU BBA, email: wooqianghao [at] gmail [dot] com

2021/10/23

CONTENTS

I. INTRODUCTION

According to MedicalNewsToday, a cough, also known as pertussis, is a voluntary or involuntary act that clears the throat and breathing passage of foreign particles, microbes, irritants, fluids, and mucus; it is a rapid expulsion of air from the lungs.[7] Research shows that respiratory disease including Asthma, AR, COPD, and rhinosinusitis present coughing as primary symptoms. In addition, it is shown that patients with COPD experience changes in voice.[16]

In the middle of a pandemic, this naturally leads to the question of whether if it is possible to diagnose patients with the respiratory disease by solely observing their coughing audio footages due to the insecurity of face to face diagnosis as well as a large quantity of COVID-19, a respiratory disease, patients which increases the difficulty for a human to identify the state of a patient correctly. With the help of artificial intelligence and machine learning, technology can provide helpful and instructive suggestions that can drastically lighten the burden of medical personnel. Many organizations and academia have already taken action in providing datasets and machine learning models for COVID-19 to bring this idea to reality, such as the DiCOVA Challenge. Many approaches and features were presented in this challenge and another paper by Madhurananda Pahar, a postdoctoral fellow at the University of Stellenbosch, used decision trees, logistic regression, k-nearest neighbor, support vector machine, multilayer perceptron, convolutional neural network, long short-term memory, VGG, and residual-based neural network architecture.[17] This essay's chosen approach is spectrograms and convolutional neural networks, but the most performative models concluded by these two papers were CNN using MFCC, Resnet50, and LSTM. [22] In this essay, I will attempt to explore how the combination of CNN and spectrograms compare to the above architectures by converting acoustic recordings into spectrograms and applying an image recognition neural network (CNN) to classify them. The dataset used includes Coswara Dataset, Coughvid dataset, and Virufy dataset due to their accessibility and scale.

II. THEORETICAL BACKGROUND

A. Neural Networks

1. *Neurons*

Fundamental knowledge of neural networks is required to understand the aim and execution of this essay. Neural Network (NN) is a category of machine learning that contains a network of neurons that aims to accomplish work generally done by a human brain. The weight represents the impact a neuron has on other neurons. The closer this value is towards one, the higher the impact since all the neurons are interconnected from different network layers. Bias is an adjustable value added to the neurons from previous neurons. Computer scientists used an activation function to squeeze the values into the range from zero to one to reduce complexity and chances of error. Some of the most popular activation functions include the Sigmoid function and ReLU shown in diagram 1.[20]

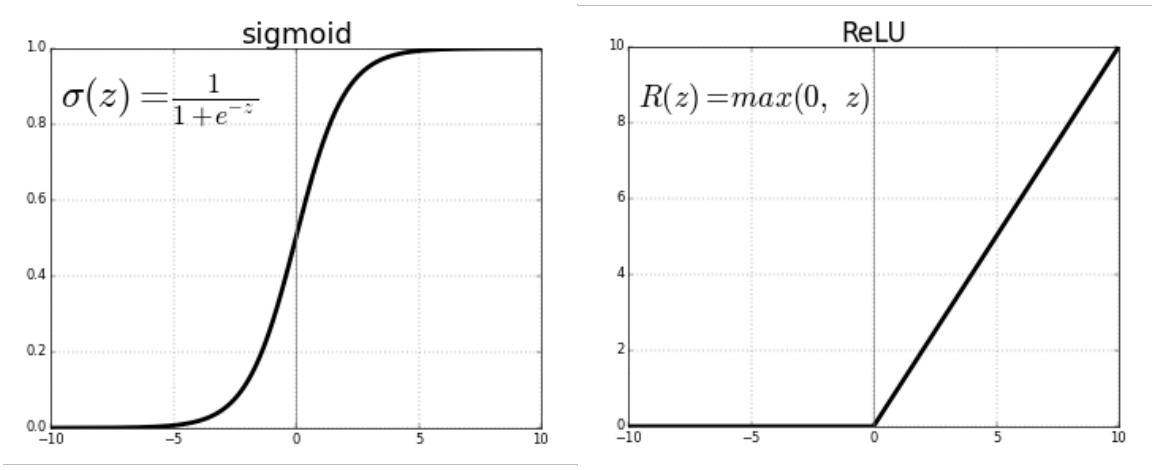
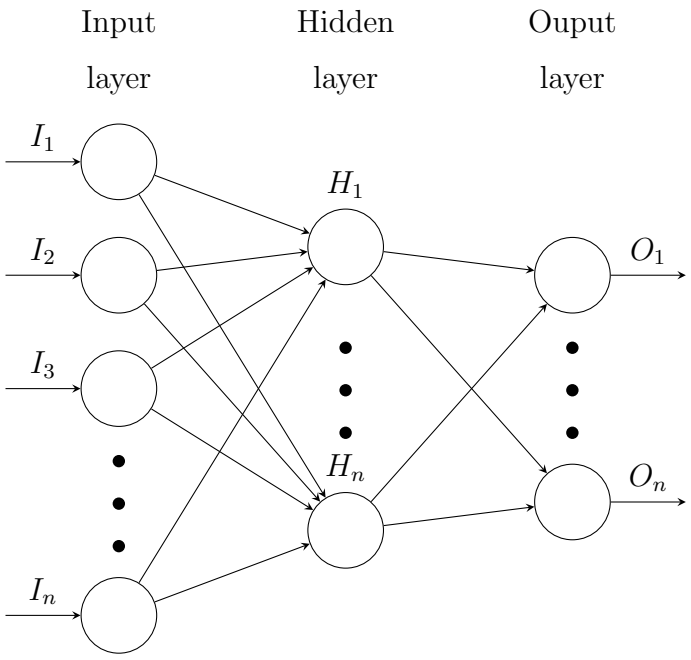


FIG. 1. Sigmoid and ReLU Function[24]

2. *Layers*

Looking at the diagram below, we can see that the neurons are arranged in a web-like structure in which every neuron in one column is connected to the next. These columns are called layers of the NN, and they are divided into three main categories, including an input layer, hidden layers, and output layer. The input layer is the first layer of the NN in

which the input data is fed into the network to study and train. The hidden layer is where the calculations happen. As the values inside each neuron are passed to the next layer with different weights and biases, the network using the input values ends up with values in the output layer, which is the final result a NN gave for a particular input.[20]



3. Matrix

By combining the above concepts, a NN can be condensed into the form of a matrix shown below in which w is the weights assigned to each neuron that is represented by x. b is the bias, and a is the output of one layer that will be sent to the next layer.[20]

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix}_{\text{activation}} \rightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{1}$$

4. Optimizer & Loss Function

An optimizer and backpropagation are used to improve the model's performance to more than just guessing randomly. They are responsible for two main components of improving a NN, including cost function and gradient descent.[13]

a. Cost Function/Loss Function To measure the performance of a cost function or a loss function is used. The higher the cost/loss function, the more inaccurate the model is.[13] The loss function chosen for this experiment is categorical cross-entropy. It is a multiclass classification problem in which we need to distinguish between two classes, healthy or sick, from the subject's coughing audio spectrogram.

One might question that since there are only two classes involved in this problem, it is more suitable to use binary cross-entropy. However, since binary cross-entropy is a particular case of categorical cross-entropy, they can be used interchangeably in this problem. We can prove this by substituting variable m with two into the equation 2, which will produce the equation 3, which is binary cross-entropy. Therefore, this decision will not differ from using binary cross-entropy.[11]

$$Categorical = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log\left(p_{ij}\right) \tag{2}$$

$$Binary = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\log\left(p_i\right) + \left(1-y_i\right)\log\left(1-p_i\right)\right] \tag{3}$$

Because the modifications on weights and biases are often minuscule, which will not result in any changes in the number of images predicted correctly, so the model cannot adjust weights and biases accordingly, instead, by using a smooth function like categorical cross-entropy, the model can constantly observe if the model is improving despite making minimal changes, so changes can be made constantly to minimize the loss function even on a small scale. [13]

b. (Stochastic) Gradient Descent With the cost function, we can observe the performance of the network, and in order to maximize its performance, gradient descent is used to find the cost function's minimum. This is achieved by calculating the first-order derivative of the cost function to determine which direction the weights and biases are modified into and gradually reach a minimum. However, gradient descent is heavily limited by a large dataset

since it only changes weight after training of one dataset and its inability to escape local minimums since the derivative is zero at both the local and global minimum. Therefore, I decided to use a variant of gradient descent, stochastic gradient descent. The graph below shows that stochastic gradient descent has far more fluctuations, meaning it has a higher chance of finding the global minimum instead of falling into one local minimum. This action of adjusting weights and biases according to the loss is called backpropagation.[9]

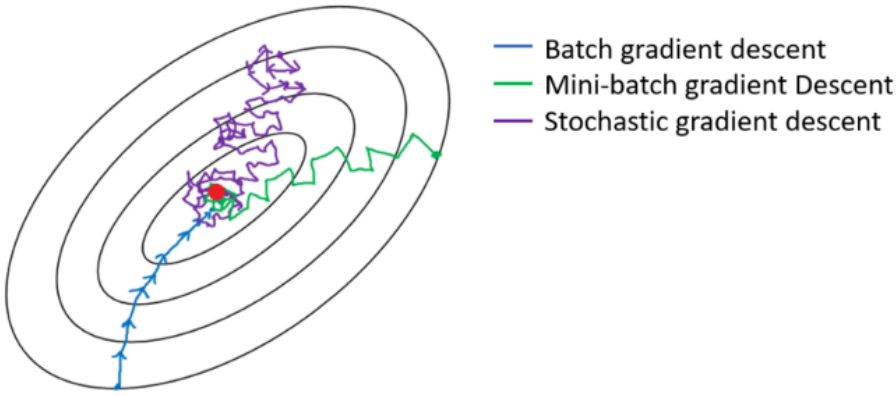


FIG. 2. Stochastic Gradient Descent vs Batch Gradient Descent vs Mini-Batch Gradient Descent[9]

B. Convolutional Neural Network & Spectrograms

1. *Convolutional Neural Network*

In order to see how convolutional neural networks (CNN) work with spectrograms to achieve the objective of this essay, a fundamental understanding of CNN is required. What distinguishes CNN from traditional feedforward neural networks is its unique ability to assign features in an image with weights and biases. The CNN was inspired by connectivity patterns in the human brain and the organizations of the visual cortex. Before applying CNN to an image, the image is converted to a table of values that could be fed into the CNN using color spaces including RGB, HSV, CMYK, or greyscales.[14]

a. Kernel How CNN captures vision features is by using kernel or also known as the filter and the convolutional layer. A kernel is a grid that performs the convolution operation to every pixel within an image through matrix multiplication between values inside the

kernel and the image portion it is hovering over. Once it is done with one portion of the data, it moves one stride length to the right and repeats the process until it reaches the end, then it moves back to the leftmost pixel and repeats until it traverses through the entire image. All the values produced by the matrix multiplication are then combined into another matrix used in future calculations.[10]

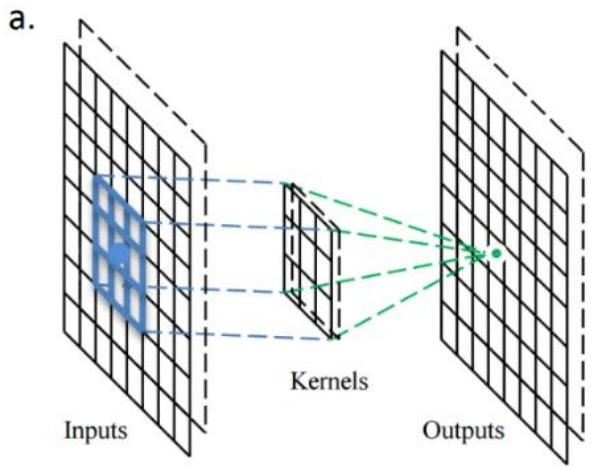


FIG. 3. Kernel[14]

b. Padding The convolutional layer produces two types of results. One is when the dimensions of the convolved features are reduced compared to the input image. The other is when dimensions remain the same or increase compared to the input. These two results are produced by using valid padding and the same padding.[14]

When we add an extra layer of empty surrounding to an image, a matrix with the image's exact dimension is produced, hence the same padding.[14]

On the other hand, if we apply the convolutional layer to the image directly without padding, a matrix with the same dimension as the kernel is produced, which is valid padding.[14]

c. Pooling Like the convolutional layer, the pooling layer also aims to reduce the spatial complexity of the input image. Generally, two types of pooling are used, max pooling and average pooling. Average pooling returns the average of all the values from the image portion that the kernel is hovering over. Max pooling only returns the maximum value from the image portion that the kernel is hovering over.[27]

In general, max pooling is preferred compared to average pooling due to its noise-suppressing ability. It discards the noisy activations altogether and performs de-noising

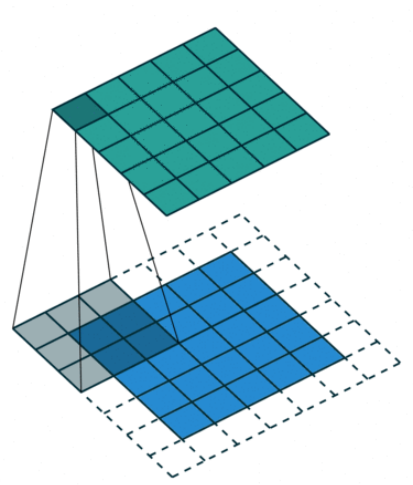


FIG. 4. Same Padding[4]

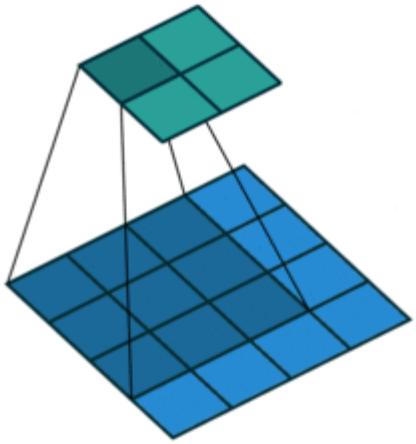


FIG. 5. Valid Padding[5]

with dimensionality reduction. On the other hand, Average Pooling performs dimensionality reduction as a noise suppressing mechanism.[27]

Figure 7 is an example of how pooling and kernel work together to drastically reduce the spatial size and extract features from an image.

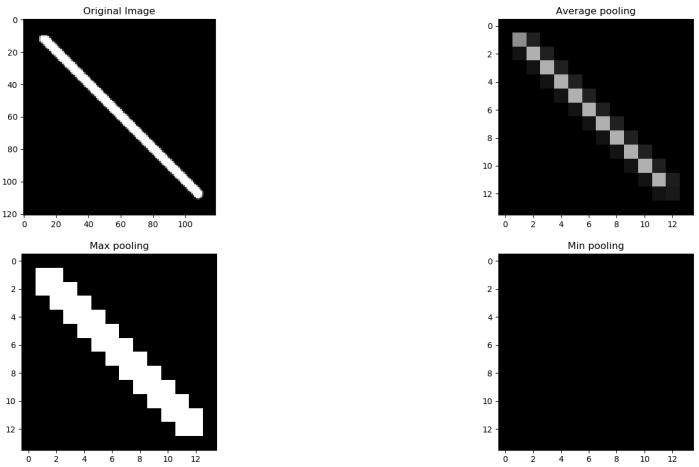


FIG. 6. Average Pooling vs Max Pooling[2]

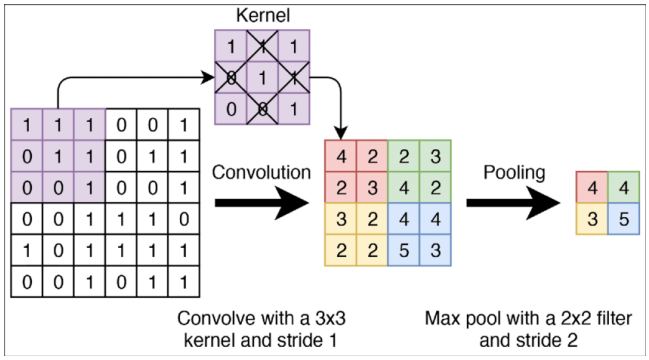


FIG. 7. Summary of a CNN[12]

d. Fully Connected Layers Neurons in a fully connected layer ultimately connect to all the previous neurons like a traditional feedforward neural network. One notable use of FC layers is that they can be connected to the flattened convolutional layer or pooling layer to execute the learning of the feature in the image and perform the classification. We can see that this is very frequently used in the Inception V3 architecture chosen for this essay.[8]

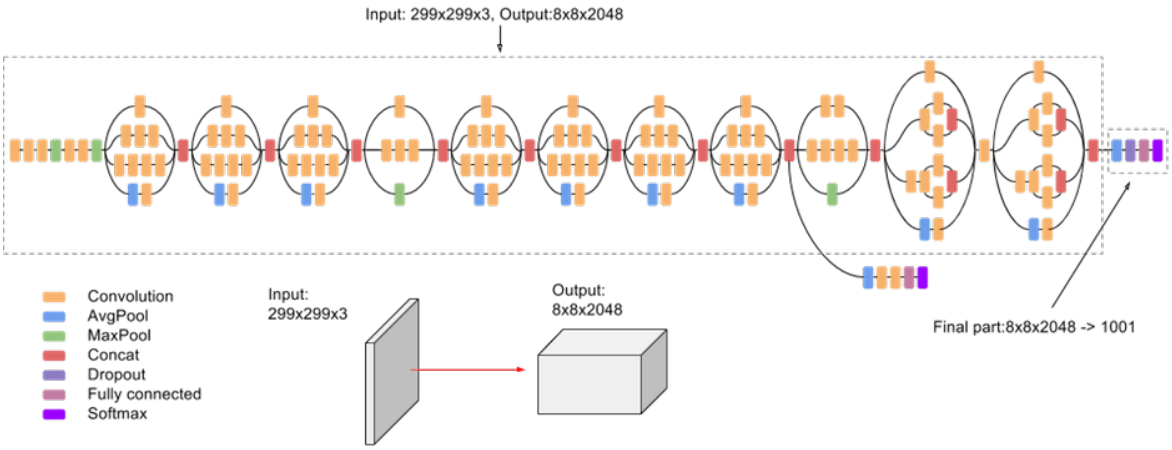


FIG. 8. Fully Connected Layers in InceptionV3 Arhitecture[**inceptionv3**]

2. *Spectrograms*

A spectrogram is a figure which represents the spectrum of frequencies of recorded audio over time. This means that the brighter it is in one part of the figure, the more sound is concentrated on this part of the frequencies. Similarly, the darker spots on the graph reflect low or empty sounds. This tool is handy since we can learn many essential audio features without listening to it, which is essential for CNNs to treat it like an image rather than an audio file.[15]

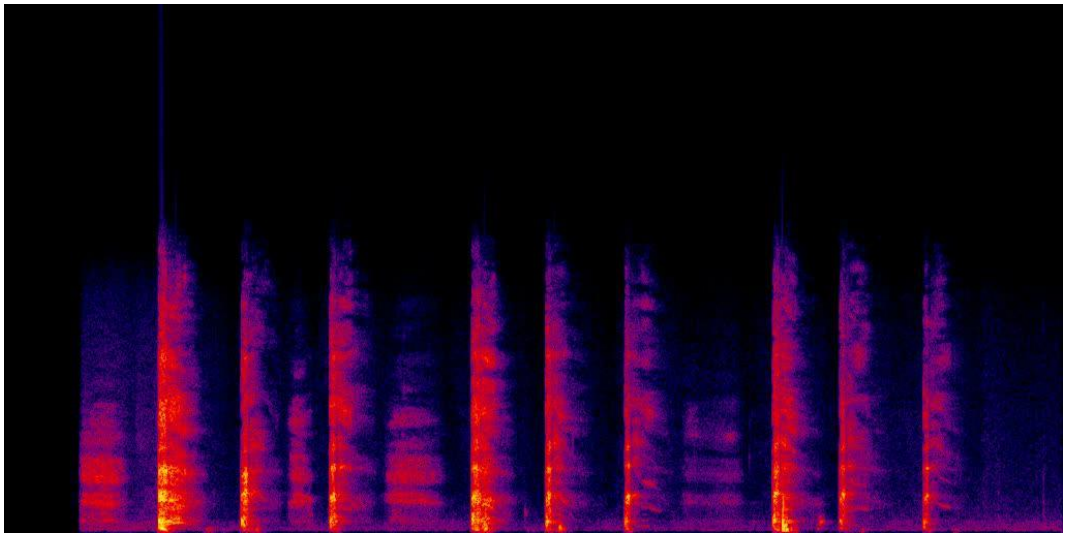


FIG. 9. Sample Spectrogram from the Dataset

We can produce a spectrogram very quickly by first dividing the audio into multiple overlapping small trunks of audio and applying the short-time Fourier transformation, equqation 4 on each trunk, producing a vertical line for each trunk. Using each vertical line to convert them to decibels and combine them back together produces the complete spectrogram of an audio file.[15]

$$\mathrm{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega m} \tag{4}$$

III. METHODOLOGY & EXPERIMENT

A. Experimental Procedure

1. Use Excel and Python to generate a CSV file containing the health status of each audio file.
2. Convert the .wav and .mp3 files into spectrograms using Bash and FFMPEG
3. Repeat steps 1 - 2 for all three dataset
4. Combine the CSV files using Excel and put all spectrograms in one folder
5. Use Python to generate a test folder randomly
6. Upload all files to Google Drive
7. Setup the model in TensorFlow and Keras on Google Collaboratory
8. Train the model on the train folder
9. Save the trained model as an h5 file
10. Produce the predictions on the test data using the h5 file
11. Generate graphs using the predictions and confirmed labels

B. Input Data & Data Processing

I read the CSV classification files provided by the dataset providers using a python script and produced folders classified into sick and healthy divided for training, validation, and testing purposes.

```
train/
 healthy/
   heavy_0Ha52POVIxTKEPqI1eGpIoMHUd52.wav.jpg
   heavy_01OCEf1yB4czsq8ygRoT51s96Ba2.wav.jpg
   heavy_01OCEf1yB4czsq8ygRoT51s96Ba2.wav.jpg
   ...
 sick/
   00039425-7f3a-42aa-ac13-834aaa2b6b92.wav.jpg
   0009eb28-d8be-4dc1-92bb-907e53bc5c7a.wav.jpg
   001328dc-ea5d-4847-9ccf-c5aa2a3f2d0f.wav.jpg
   ...
test/
 healthy/
   heavy_0EAAFsDWfTcrhktHy78LS6nf19G3.wav.jpg
   heavy_0Nuh8uDalHe47HGM31i2Ew6BPcl1.wav.jpg
   heavy_0bcMNFt3dlPlUPTyCO8DVAlbTUC3.wav.jpg
   ...
 sick/
   00291cce-36a0-4a29-9e2d-c1d96ca17242.wav.jpg
   004c24d8-e8cd-4755-86f6-5a1d8c7920c7.wav.jpg
   0066b126-104a-45a6-a88e-0697c6baa0aa.wav.jpg
   ...
```

Finally, I used FFMPEG, open-source software libraries which handle multimedia files such as audio recordings and a bash script to convert all the audio files in the formats of FLAC, M4A, WAV, and MP3 into a uniform size of 512 x 1024 pixels JPEG spectrograms.

C. Model Architecture

The architecture I chose to use is InceptionV3, a CNN for image analysis and object detection developed by researchers at Google known for excellent object detection and classification, which could be helpful in this experiment. As the diagram below shows, the architecture comprises symmetric and asymmetric building blocks consisting of convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used throughout the model and applied to activation inputs. Loss is computed via Softmax. Figure 10 is a graphical illustration of the model.[1]

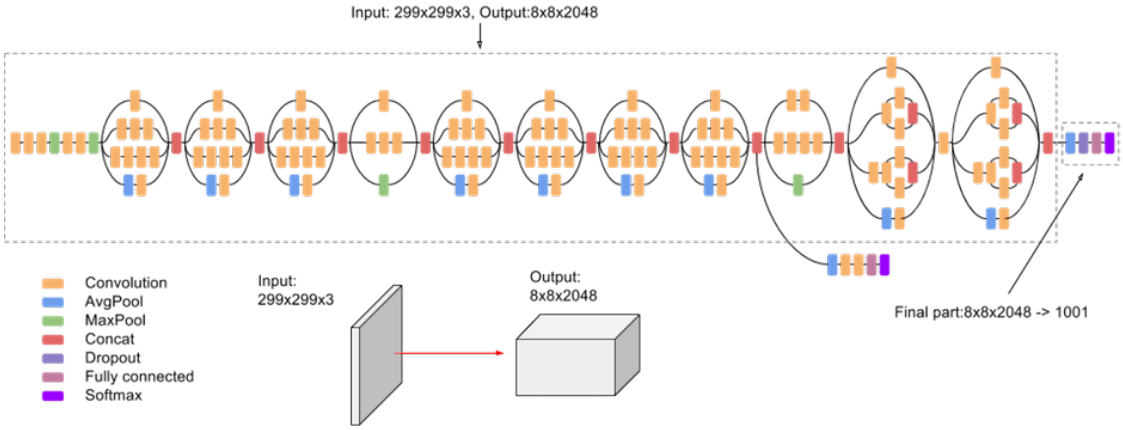


FIG. 10. InceptionV3 Architecture Summary[1]

The optimizer chosen is ADAM. Like previously mentioned, ADAM uses stochastic gradient descent to increase the chances of finding the global minima. In addition, it is widely known to be the best optimizer in machine learning since it combines the merit of AdaGrad and RMSProp algorithms, meaning that it is exceptionally good with both noisy data and sparse gradients. Also, ADAM is relatively easier to configure for performance. Looking at figure 11, we can see the significant performance gap.[3]

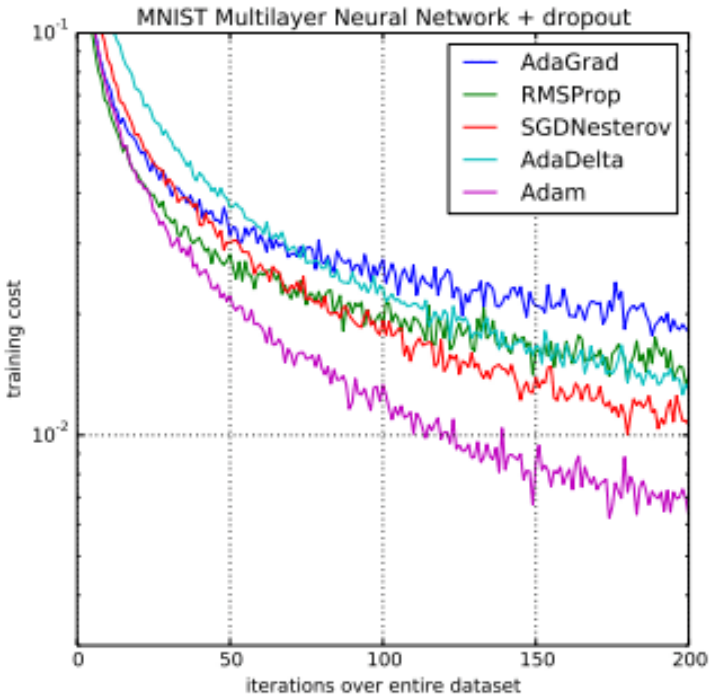


FIG. 11. Adam Optimizer vs Other Optimizers[3]

D. Training & Predicting

The entire model, training, data importing, predicting, processing results, and plotting graphs are written in Python using TensorFlow, Keras, NumPy, and matplotlib.

Listing 1. Model Building

```
normalization_layer = tf.keras.layers.experimental.preprocessing.Rescaling(1.

train_set = train_set.map(lambda x, y: (normalization_layer(x), y))

val_set = val_set.map(lambda x, y: (normalization_layer(x), y))

inception = InceptionV3(input_shape=image_size + [3], weights='imagenet', inc

for layer in inception.layers:
    layer.trainable = False
```

```
x = Flatten()(inception.output)

prediction = Dense(len(folders), activation='softmax')(x)

model = Model(inputs=inception.input, outputs=prediction)

opt = keras.optimizers.Adam(learning_rate=0.0002)

model.compile(optimizer=opt,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=
              metrics=['accuracy'])
```

Since I have limited data, the model is trained on the same dataset for 15 epochs. The model can use stochastic gradient descent to further enhance the model's performance even on the same dataset.

IV. ANALYSIS & CONCLUSION

A. Evaluation Metrics & Explanation

In order to understand the result of the experiment, a wide range of evaluation metrics were chosen to measure the performance of this including accuracy and loss, precision and recall, ROC(Receiver Operating Characteristics) and AUC(Area Under the Curve), as well as confusion matrix and MCC (Matthew's Correlation Coefficient). Some might argue that the F1 score should be included since it is one of the most commonly used metrics in machine learning classification models. However, research shows that it is not as accurate as of the MCC and will not be evaluated in this essay.[6] (See Appendix) The following is an array of outputs produced by the model.

[2.0739768e-05 2.1324334e-01 1.1255335e-05 ... 1.0000000e+00 1.8497093e-07 1.0000000e+00]

To begin with, the confusion matrix, figure 12 provides us with the most direct illustration of the model's performance by indicating the number of true and false predictions in each class. In this model, positive is defined as a subject with Covid positive and vice versa.

For each box, true negative corresponds to the number of correctly labeled negative images, false-negative corresponds to the number of wrongly labeled negative images, the same logic applies to the positive boxes. [19]

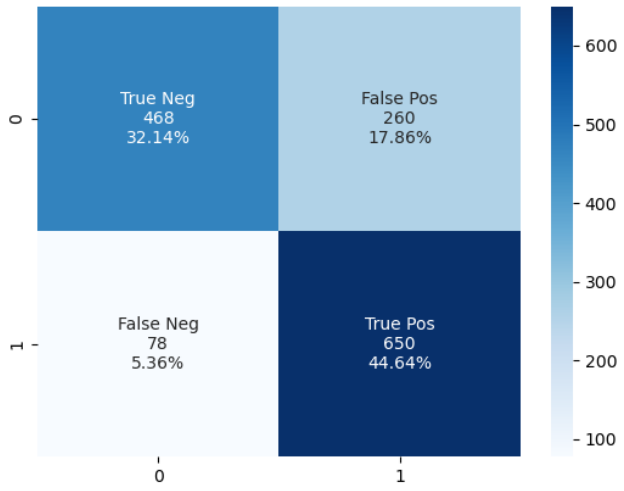


FIG. 12. Confusion Matrix

The loss is defined by the difference between the ground truth value, values known to be true, and the predicted value produced by the loss function, in this case, categorical cross-entropy. On the other hand, accuracy is simply the number of correct predictions over the total number of predictions. However, this is a highly inaccurate depiction of a model's performance. For example, a model that detects dangerous objects for flight transportation can easily score up to 99% accuracy by always predicting negative, since most of the time, the articles transported by flight are safe. However, if one is not and the model implemented does not recognize it, the model could cause severe damage.[25]

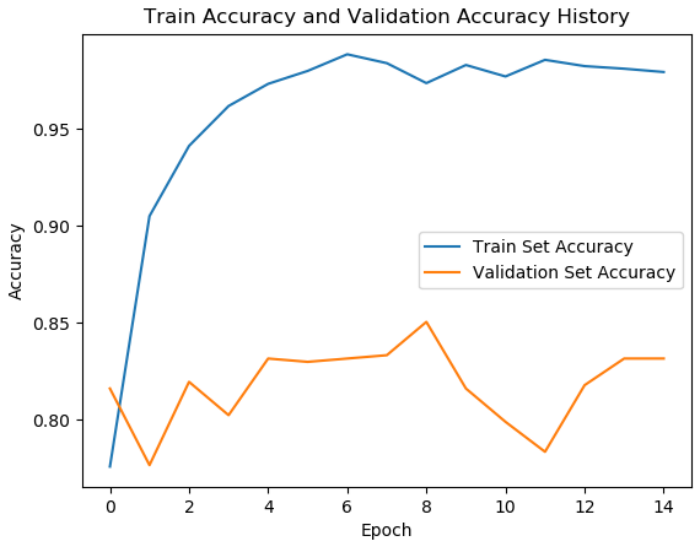


FIG. 13. Accuracy vs Epoch for Training and Validation Dataset

FIG. 14. Loss vs Epoch for Training and Validation Dataset

Therefore, we also need a clear understanding of how the model performs on all classes depending on the importance and the cost of specific predictions. This is where precision and recall help. By referring to the confusion matrix, precision is defined as true positive over total predicted positives (true positive + false positives). [26]

$$\text{Precision} = \frac{tp}{tp + fp} \tag{5}$$

Using precision, we can see what percentage of predicted positives are positive. Therefore, applicable when the cost of false positives is high. The recall is defined as the true positives over actual positives (true positives + false negatives); using the same logic as precision, we can see that recall is valid when the cost of false negatives is high. [26]

$$\text{Recall } = \frac{tp}{tp + fn} \tag{6}$$

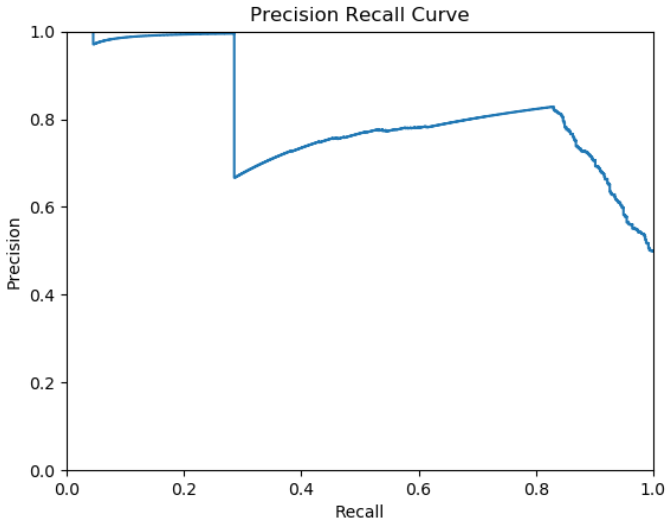


FIG. 15. Recall vs Precision on Test Dataset

ROC is a graphical representation of a binary classifier's diagnostic ability at different classification thresholds. A classification threshold needs to be chosen between 0 and 1 (usually 0.5). The ROC curve provides insight into how to optimize this threshold by illustrating true positive rates and false-positive rates on the x and y-axis at different classification thresholds, so a more suitable classification threshold can be chosen for different cases and scenarios to optimize the performance of the model. FPR and TPR can be calculated using the below formula. [18]

$$FPR = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{7}$$

$$TPR = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

Area Under a Curve (AUC) is the area under the ROC curve that conveniently summarizes the model's overall performance at all classification thresholds. [18]

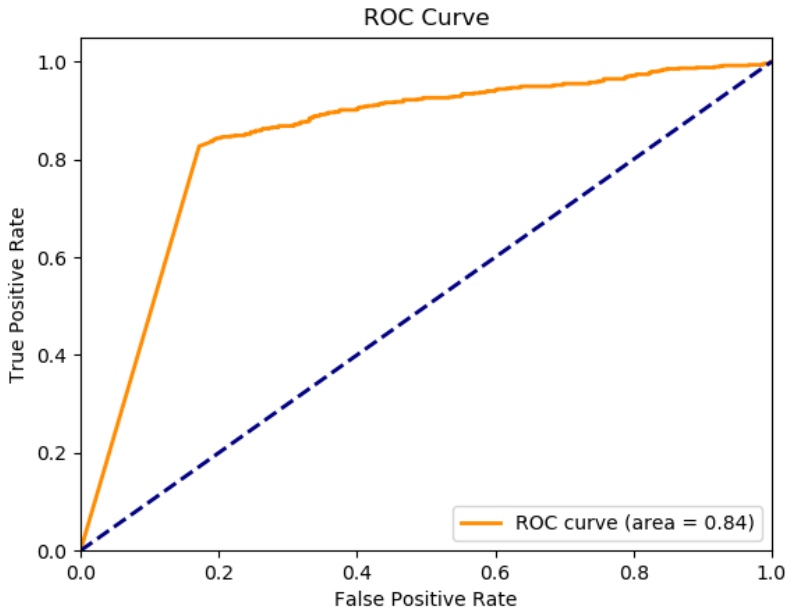


FIG. 16. ROC Curve on Test Dataset

Finally, I chose to compute Matthew's Correlation Coefficient (MCC). What makes MCC unique and different from all other metrics is how it treats true and predicted values as two different variables and calculates their correlation coefficient using the formula below. This coefficient is known as the phi coefficient. Therefore, the higher the correlation coefficient between the two variables, the better the model's performance.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{9}$$

By looking at the formula of MCC, we can see a balanced consideration of all four boxes of the confusion matrix, unlike precision, or recall which only considers two or three boxes. If you swap positive and negative, the MCC remains the same because it is completely symmetric and considers every box equally important. In contrast, if we were to do the same to precision or recall, the values will be reversed, giving a biased presentation of the model if only one is looked at, which further demonstrates MCC's ability to summarize a model's classification ability through an elegant formula. The achieved MCC of this experiment is 0.5532833351724882.

B. Result Analysis

Looking at a variety of metrics, the model's performance has all been generally positive and shows some promising potential in further development. However, some metrics show the significant weakness of this model. Looking at the confusion matrix, we can easily spot that the false positive rate is much higher than the false-negative rate(260 vs. 78). The model very frequently diagnoses healthy subjects to be sick, which is better than the opposite but can also be very problematic. This is reflected through the precision, too, scoring only 64 percent compared to 89, 86, and 71 percent for other boxes.

Similarly, the MCC also presents a relatively mediocre score taking account of both FPR and FNR. We know that this is a crucial issue because this problem requires both a low false positives rate and a false-negative rate since if we cannot diagnose subjects with the virus, then the spread could cause considerable damage. On the other hand, if we wrongly diagnose healthy subjects to be infected, it can create inconveniences that could impede the progress of many important events, such as flights or hospitalization priorities.

To further understand the model's performance, the results are also comparable if we compare this model to models developed by professionals and academics. Looking at the DiCOVA challenge hosted by the Indian Institute of Science with the same objectives as this essay, using acoustics to diagnose Covid-19, we can see that this model outperformed the majority of the models listed with an AUC score of 0.84, only slightly lower than the T1 by a margin of 3% and T2 by 1%.[17] Therefore, proving the hypothesis that the combination of CNN and spectrogram is usable and even has a competitive edge against most other architecture and methods. However, this doesn't mean that my model is designed just as well as the professionals, since only a relatively small dataset is provided to the participants, which requires very technical hyper tuning and data manipulations to maximize performance. On the other hand, I have compiled a much larger dataset of 8000 samples, so a crude method of directly applying CNN to spectrograms still works just about as well as the professional models. The result shows that this model is not correctly hyper-tuned and modified to the degree in the DiCOVA challenge. The primary reason why this model performed just about as excellent on the ROC & AUC metrics is that I had a massive data advantage. If this model is tuned and trained using more professional knowledge and more sophisticated strategies, the results should be better. If these participants had a larger dataset such as a

mixed dataset I used, combined with their more professional and knowledgeable operations in creating a model, the results would be much better.

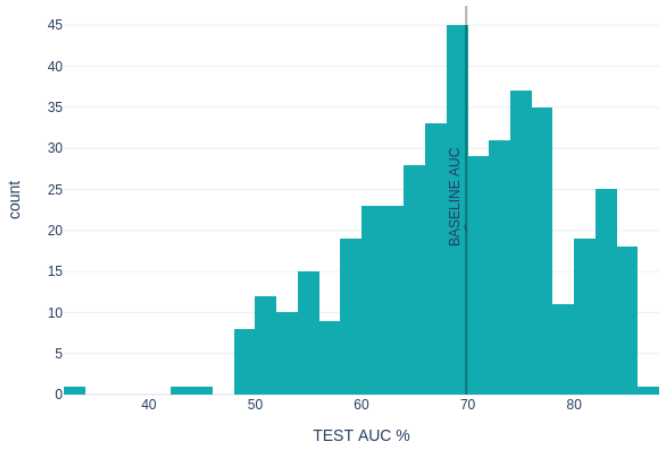


FIG. 17. DiCOVA Challenge Models' Performanc[17]

C. Limitations

In this experiment, though the metrics reflect somewhat positive results and the possibility of extending this model to more significant and realistic scenarios, there are still several limitations of the model and logistics of the combination of CNN and spectrograms. Furthermore, there is also a big room for improvement regarding data preparation and model selection.

1. Limited & Unbalanced Dataset

In computer science, there is a saying that "garbage in, garbage out," meaning that data is an essential part of creating a neural network or machine learning model. Even though several datasets created by credible institutes and organizations in this experiment are used for the process and training, there is still a large room for improvement.

First of all, even though all the data are collected and made public by credible institutes and organizations, including the Indian Institute of Science, EPFL, and Virufy, only Virufy contains audio files that are artificially modified to enhance the quality of the audio record-

ings for machine learning. Coswara and Coughvid datasets are all crowd-sourced, recorded using the subject's recording device, which creates uncertainty with frequency, amount of energy for every subject's recording that would, in turn, reflect on the spectrogram.

Secondly, though four professional physicians labeled about 2800 subjects inside the Coughvid dataset, most of the subjects in the three datasets were self-labeled and prone to error.[21] Most importantly, when many subjects refused to fill out a history of respiratory and pulmonary disease, that could strongly impact the model's training as these diseases could drastically impact the produced spectrogram.

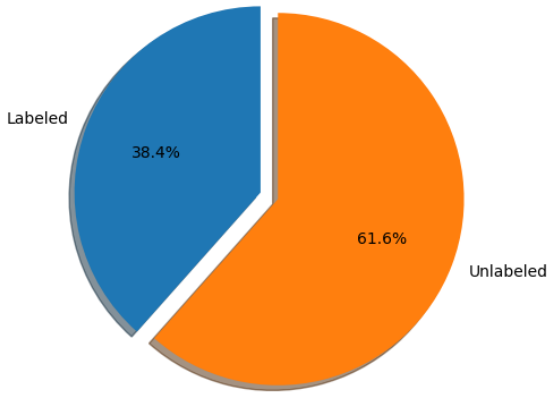


FIG. 18. Expert Labelled vs User Labeled Data

Finally, the distribution of negative and positive subjects is unbalanced. Surprisingly, there are more positive subjects than negative. Looking at the pie chart, we can see that there are about two times more positive samples than negative samples, which would make the model better at spotting positive subjects when making sure a subject is healthy is also essential.

2. Fundamental Flaw of CNN & Spectrograms

a. Logistics of Axis in Spectrogram and CNN A CNN from a theoretical point of view should not be used with spectrograms. CNN is fundamentally an image recognition architecture that is meant to find visual patterns despite the scale and position of the image. This means that a CNN can classify images and recognize them despite them being reversed

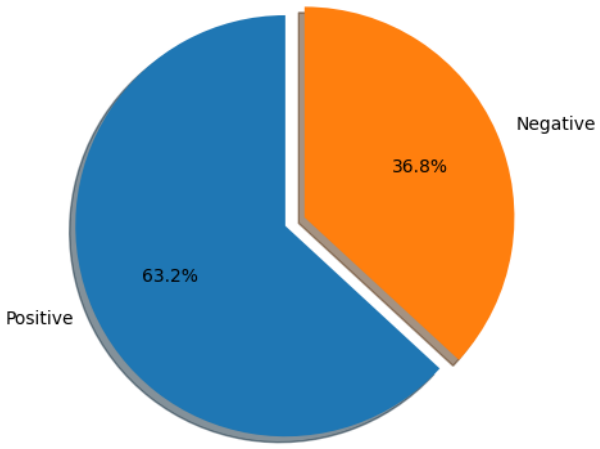


FIG. 19. Negative vs Positive Samples in the Dataset

or resized in different locations. It does not matter where the object is located in an image. A cat is still a cat, even if upside down or stretched. However, this is not the case for spectrograms. The axis on a spectrogram represents frequency strength and time. If an image is shifted upwards, the frequency is higher, which would mean a completely different thing compared to when it is lower. Hence, the premise of CNN entirely contradicts the spectrogram's axes.[23]

b. Non Local Properties The pattern presented in the spectrogram also contradicts how normally a CNN is used to capture visual patterns. Usually, in a CNN, pixels next to each other based on distance are considered an entire visual object. The CNN can analyze the color and formation relationship between surrounding pixels and deduce the observed object. However, spectrograms present audio patterns in an entirely different way. Spectrograms contain non-local properties meaning that often important features are separated across the diagram, such as the increase in frequency strength that could be divided across the entire diagram across the entire image, which could be crucial to critical features such as gender, and age. [23]

D. Conclusion

In conclusion, the combination of CNN and spectrogram undoubtedly performed positively in this experiment with relatively small dataset size. Though there are still some issues in the logistics of such technology, and more data is needed to be collected to explore further the feasibility of its application in the actual world diagnosis, current observations show that it can be solid support for physicians in diagnosing Covid-19.

V. APPENDIX

Classification Report

precision-recall f1-score support

0 0.64 0.86 0.73 546

1 0.89 0.71 0.79 910

accuracy 0.77 1456

macro avg 0.77 0.79 0.76 1456

weighted avg 0.80 0.77 0.77 1456

Covid CNN

```
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
import tensorflow as tf
import numpy as np
from glob import glob
import os, sys
# import necessary libraries and configurations

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)

folders = glob(os.path.join(sys.path[0], "train/*"))
data_dir = os.path.join(sys.path[0], "train/")

batch_size = 32
image_height = 512
image_width = 1024
image_size = [image_height, image_width]

train_set = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.1,
  subset="training",
  seed=123,
  image_size = (image_height, image_width),
  batch_size = batch_size)

val_set = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.1,
  subset="validation",
```

```
  seed=123,
  image_size = (image_height, image_width),
  batch_size = batch_size)

normalization_layer = tf.keras.layers.experimental.preprocessing.Rescaling(1.

train_set = train_set.map(lambda x, y: (normalization_layer(x), y))

val_set = val_set.map(lambda x, y: (normalization_layer(x), y))

inception = InceptionV3(input_shape=image_size + [3], weights='imagenet', inc

for layer in inception.layers:
    layer.trainable = False

x = Flatten()(inception.output)

prediction = Dense(len(folders), activation='softmax')(x)

model = Model(inputs=inception.input, outputs=prediction)

opt = keras.optimizers.Adam(learning_rate=0.0002)

model.compile(optimizer=opt,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=
              metrics=['accuracy'])

history = model.fit(
train_set,
validation_data = val_set,
epochs = 15,
```

```
verbose = 1
)

np.save(os.path.join(sys.path[0], "history.npy"),history.history)
model.save(os.path.join(sys.path[0], "trained_model.h5"))
model.save_weights(os.path.join(sys.path[0], "checkpoint"))

Data Evaluation

from sklearn.metrics import auc
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import matthews_corrcoef
import matplotlib.pyplot as plt
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
import os, sys

# import necessary libraries and configurations

history = np.load(os.path.join(sys.path[0], "history.npy"),allow_pickle='TRUE

plt.plot(history['loss'], label='Train Set Loss')
plt.plot(history['val_loss'], label='Validation Set Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Train Loss and Validation Loss History')
plt.legend()
plt.savefig(os.path.join(sys.path[0], "loss_history.png"))
plt.show()
```

```
# plot the loss

plt.plot(history['accuracy'], label='Train Set Accuracy')
plt.plot(history['val_accuracy'], label='Validation Set Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Train Accuracy and Validation Accuracy History')
plt.legend()
plt.savefig(os.path.join(sys.path[0], "accuracy_history.png"))
plt.show()

# plot the accuracy

def roc_auc(test_predict, test_labels):
  fpr, tpr, _ = roc_curve(test_labels, test_predict)
  roc_auc = auc(fpr, tpr)

  plt.figure()
  lw = 2
  plt.plot(fpr, tpr, color='darkorange',
           lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
  plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('ROC Curve')
  plt.legend(loc="lower right")
  plt.savefig(os.path.join(sys.path[0], "roc_auc.png"))
  plt.show()
```

```
  # plot the AUC & ROC

def precision_recall(test_labels, test_predict):
  from object_detection.utils.metrics import compute_precision_recall
  precision, recall = compute_precision_recall(test_predict, test_labels,728)
  plt.figure()
  plt.step(recall, precision, where='post')
  plt.xlabel('Recall')
  plt.ylabel('Precision')
  plt.title('Precision Recall Curve')
  plt.xlim((0, 1))
  plt.ylim((0, 1))
  plt.savefig(os.path.join(sys.path[0], "recall_vs_precision.png"))
  plt.show()

def conf_matrix(test_labels, test_preclasses):
  cf_matrix = confusion_matrix(test_labels, test_preclasses)
  group_names = ["True Neg","False Pos","False Neg","True Pos"]
  group_counts = ["{0:0.0f}".format(value) for value in
                  cf_matrix.flatten()]
  group_percentages = ["{0:.2%}".format(value) for value in
                       cf_matrix.flatten()/np.sum(cf_matrix)]
  labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
            zip(group_names,group_counts,group_percentages)]
  labels = np.asarray(labels).reshape(2,2)
  sn.heatmap(cf_matrix, annot=labels, fmt="", cmap='Blues')
  plt.savefig(os.path.join(sys.path[0], "confusion_matrix.png"))
  plt.show()

def class_report(test_preclasses, test_labels):
```

```
  text_file = open(os.path.join(sys.path[0], "classification_report.txt"), "w
  n = text_file.write(classification_report(test_preclasses, test_labels))
  text_file.close()

test_predict = np.load(os.path.join(sys.path[0], "test_predict.npy"))
test_labels = np.load(os.path.join(sys.path[0], "test_labels.npy"))
test_labels = test_labels.astype('float64')
test_preclasses = []

for prediction in test_predict:
  if prediction > 0.5:
    test_preclasses.append(1)
  else:
    test_preclasses.append(0)

roc_auc(test_predict, test_labels)
precision_recall(test_labels, test_predict)
conf_matrix(test_preclasses, test_labels)
class_report(test_preclasses, test_labels)
print(matthews_corrcoef(test_labels, test_preclasses, sample_weight=None ))

Result Processing

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import os, sys

# import necessary libraries and configurations

model = load_model(os.path.join(sys.path[0], "trained_model.h5"))
model.load_weights(os.path.join(sys.path[0], "checkpoint"))
```

```
# load model for testing purposes

data_dir = os.path.join(sys.path[0], "test/")

batch_size = 32
image_height = 512
image_width = 1024
image_size = [image_height, image_width]

validation_datagen = ImageDataGenerator(rescale=1/255)
validation_generator = validation_datagen.flow_from_directory(
        directory = data_dir,
        classes = ['Healthy', 'Sick'],
        target_size=(image_height,image_width),
        batch_size=batch_size,
        class_mode='binary',
        shuffle=False)

STEP_SIZE_TEST=validation_generator.n//validation_generator.batch_size
validation_generator.reset()

test_predict = model.predict(validation_generator, verbose = 1)
test_predict = np.array(test_predict)[:,1]

np.save(os.path.join(sys.path[0], "test_predict.npy"),test_predict)
np.save(os.path.join(sys.path[0], "test_labels.npy"),validation_generator.cla
```

REFERENCES

[1] Advanced guide to inception V3 on cloud TPU nbsp;—nbsp; google cloud. URL: https://cloud.google.com/tpu/docs/inception-v3-advanced.

[2] Madhushree Basavarajaiah. Which pooling method is better? maxpooling vs Minpooling vs average pooling Aug. 2019. URL: https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9.

[3] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning. Jan. 2021. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20combines%20the%20best%20properties,do%20well%20on%20most%20problems..

[4] Category:convolution. URL: https://commons.wikimedia.org/wiki/Category:Convolution.

[5] Category:convolution. URL: https://commons.wikimedia.org/wiki/Category:Convolution.

[6] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: BMC genomics 21.1 (2020), pp. 1–13.

[7] Coughs: Causes, symptoms, and treatments. URL: https://www.medicalnewstoday.com/articles/220349.

[8] CS231n Convolutional Neural Networks for Visual Recognition. URL: https://cs231n.github.io/convolutional-networks/.

[9] Imad Dabbura. Gradient Descent Algorithm and Its Variants. Sept. 2019. URL: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3.

[10] Steven Dye. An intro to kernels. Mar. 2020. URL: https://towardsdatascience.com/an-intro-to-kernels-9ff6c6a6a8dc.

[11] Rafay Khan. Where did the Binary Cross-Entropy Loss Function come from? Dec. 2020. URL: https://towardsdatascience.com/where-did-the-binary-cross-entropy-loss-function-come-from-ac3de349a715.

[12] K Lambers et al. "Learning to look at LiDAR: The use of R-CNN in the automated detection of archaeological objects in LiDAR data from the Netherlands". In: Journal of Computer Applications in Arc 2.1 (2019), pp. 31–40.

[13] Conor Mc. Machine learning fundamentals (I): Cost functions and gradient descent. Apr. 2021. URL: https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220.

[14] Christopher Thomas BSc Hons. MIAP. An introduction to Convolutional Neural Networks. May 2019. URL: https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7.

[15] Mlearnere. Learning from audio: Spectrograms. Apr. 2021. URL: https://towardsdatascience.com/learning-from-audio-spectrograms-37df29dba98c.

[16] Enas Elsayed Mohamed et al. "Voice changes in patients with chronic obstructive pulmonary disease". In: Egyptian Journal of Chest Diseases and Tuberculosis 63.3 (2014), pp. 561–567.

[17] Ananya Muguli et al. "DiCOVA Challenge: Dataset, task, and baseline system for COVID-19 diagnosis using acoustics". In: arXiv preprint arXiv:2103.09148 (2021).

[18] Sarang Narkhede. Understanding AUC - roc curve. June 2021. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[19] Sarang Narkhede. Understanding confusion matrix. June 2021. URL: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62?gi=c10f1bccfd5e.

[20] Michael A Nielsen. Neural networks and deep learning. Vol. 25. Determination press San Francisco, CA, 2015.

[21] Lara Orlandic, Tomas Teijeiro, and David Atienza. "The COUGHVID crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms". In: Scientific Data 8.1 (2021), pp. 1–10.

[22] Madhurananda Pahar et al. "COVID-19 Cough Classification using Machine Learning and Global Smartphone Recordings". In: Computers in Biology and Medicine (2021), p. 104572.

[23] Daniel Rothmann. What's wrong with spectrograms and cnns for audio processing? Aug. 2021. URL: https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd.

[24] Sagar Sharma. Activation functions in neural networks. July 2021. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[25] Koo Ping Shung. Accuracy, precision, recall or F1? Apr. 2020. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[26] Koo Ping Shung. Accuracy, precision, recall or F1? Apr. 2020. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[27] Miguel Fernández Zafra. Understanding convolutions and pooling in Neural Networks: A simple explanation May 2020. URL: https://towardsdatascience.com/understanding-convolutions-and-pooling-in-neural-networks-a-simple-explanation-885a2d78f211.