International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 01 | January - 2023 Impact Factor: 7.185 ISSN: 2582-3930

Detecting Covid-19 Induced Pneumonia from Chest X-ray Using CNNs

KUPPIREDDY HEMA PRIYA

SCOPE & Sri Venkateswara College of Engineering, Student

Abstract - On the well-being and health of the global population, the COVID-19 pandemic continues to have a devastating impact. Health care systems around the world are facing challenges as a result of the rapid increase in patients with COVID-19. It is impossible to test every patient for the respiratory disease using conventional techniques (RT-PCR) due to the limited availability of testing kits. A crucial step in the fight against COVID-19 is the effective screening of infected patients, with radiologic imaging employing chest radiography being one of the main screening techniques.

The purpose of this study was to automatically detect COVID-19 pneumonia patients using digital chest X-ray images while improving detection accuracy using convolutional neural networks (CNNs). The dataset includes COVID-19, viral pneumonia, and typical lung X-ray images. The CNN-based model has been proposed in this study to provide an accurate classification of coronavirus-pneumonia-infected patients using chest X-ray radiographs. The results demonstrate that CNNs are useful for classifying X-ray pictures as covid-free or influenced by covid. To raise accuracy to standards that can be used in medicine, more research and experimentation with professionals are required.

***Key Words*:** COVID-19, Healthcare, chest radiography, convolutional neutral networks (CNNs), medicine.

1. INTRODUCTION

The coronavirus disease (COVID-19) pandemic erupted and became a significant public health issue on December 2019 in Wuhan, China. There is no unique medication or vaccine has also not yet been discovered against the coronavirus. The virus causing the COVID-19 pandemic disease is the severe acute respiratory syndrome coronavirus-2 (SRS-CoV-2). The COVID-19 virus family is a large group of viruses that cause diseases including Middle East Respiratory Syndrome (MRS-CoV) and Severe Respiratory Syndrome (SRS-CoV). A new specialty that was discovered in 2019 and has never been observed in humans is called COVID-19. Early findings indicate that COVID-19 only causes mild symptoms in about 99 percent of cases, with the remaining cases being serious or critical. The estimated number of Coronavirus cases worldwide as of October 4, 2020 is 35,248,330. Of these, 1,039,541 (4%) people passed away, and 26,225,235 (96%) were found. The total number of registered patients is 7,983,554. Of these, 66,267 (1%) had more serious illnesses, whereas 7,917,287 (99%) had only mild illnesses. The COVID-19 epidemic is a major concern for the entire world right now. Deaths from pneumonia caused by the SRS-CoV-2 virus are increasing day by day.

Chest radiography (X-ray) is one of the most significant procedures used to diagnose pneumonia globally. The chest X-ray is a clinical method that is quick, inexpensive, and common. In comparison to computed tomography (CT) and magnetic resonance imaging (MRI), chest X-rays provide patients a lower dose of radiation. However, it requires expertise, experience, and knowledge to correctly diagnose a patient using X-ray images. Using a chest X-ray is far more difficult to detect than using a CT or MRI. Only specialized physicians can diagnose COVID-19 by examining the chest X-ray. The number of specialists who can make this diagnosis is less than the number of general practitioners.

Even during normal times, there are not enough doctors per person in countries all over the world. Greece ranks top with 607 doctors per 100,000 residents, according to data from 2017. In other nations, this number is far smaller.

In the event of disasters like the COVID-19 pandemic, which require healthcare at the same time due to an inadequate supply of hospital beds and healthcare staff, the health system may degrade. Medical professionals, nurses, and careers are at danger from COVID-19, which is also a very contagious illness. Early diagnosis of pneumonia is crucial in terms of both containing the patient and the patient's recovery period, which will help to restrict the spread of the illness.

The use of computer-aided diagnosis (CD) has made it easier for medical professionals to accurately diagnose chest X-ray pneumonia. The use of artificial intelligence approaches is recommended since they can handle impending data sets that exceed human capacity in the field of medical care.

The addition of CD methods to radiologists diagnostic systems immediately reduces the workload for physicians and enhances efficiency and quantitative analysis. Deep learning and medical imaging-based CD systems are increasingly being used as research areas.

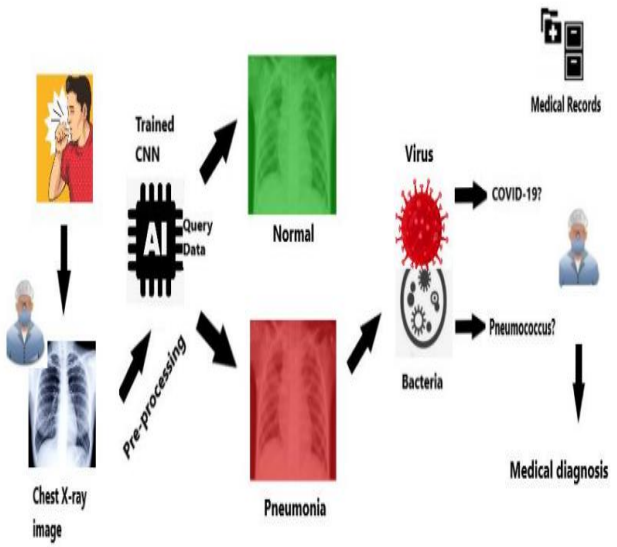


Fig 1. Flowchart of expected working of the proposed model

2. HARDWARE REQUIREMENTS

As with all machine learning/deep learning projects, more powerful hardware is always preferable. The hardware used to train the model includes:

• Processor: Intel Core i5 8th Gen

• RAM: 16 GB

• GPU: Nvidia GeForce MX130

3. SOFTWARE REQUIREMENTS

The software used for the development of the model are as follows:

• Python

• Anaconda (Prompt and Jupiter Notebook)

• Google Collaboratory

• Python modules used were:

▪ NumPy

▪ Matplotlib

▪ Keras

▪ Tensorflow

▪ Seaborn

4. EXISTING WORK

Pre-trained models for object detection from earlier efforts include the VGG16 and VGG19 models. Each of these models is larger than 300MB and has more than 100,000,000 parameters The estimated 15 million annotated photos in ImageNet span 22,000 distinct categories, and VGG16 was successful in achieving 97% accuracy on this dataset. These models are excellent for locating and identifying objects. We questioned whether it was possible to build a model with fewer parameters and layers while yet maintaining the same level of accuracy as pre-trained models.

Despite the fact that artificial intelligence (AI) has been used extensively in the past to identify and diagnose medical conditions, as the pandemic has spread and the seriousness of the situation has become apparent to the general public, it must have occurred to everyone what they could do to stop the spread, aid in the search for a cure, and the like. Always, diagnosis comes before any form of treatment. Many studies and publications have been written in light of this.

5. DEVELOPED MODEL

Once data preprocessing is completed successfully, the model we created is compact and simple to apply. Convolutional Neural Networks (CNN) were used for this project's image classification because they would produce superior results. The model is a dense final layer on top of a seven-layered CNN (fully connected layer). The first layer serves as a placeholder for the input layer and is fixed at 224 × 224 x 3, making it an RGB image. Convolution + ReLU and Max Pooling layers are combined into the following six layers. With each subsequent layer, more features can be recognized while maintaining accuracy.

The results of the experiments showed that increasing the number of layers beyond 8 did not significantly improve accuracy; rather, it increased processing time and put additional stress on the hardware. The model's accuracy in creating the confusion matrix was 98.7%.

6. DATASET DESCRIPTION

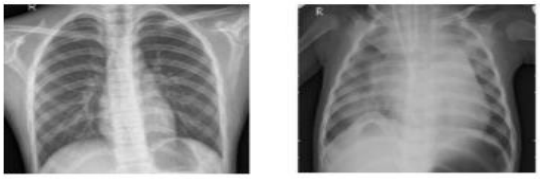
CORD-19:
(https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge/download)

It is a source for more than 100,000 full-text articles and more than 200,000 scientific publications on COVID-19, SRS-CoV-2, and related coronaviruses. The international research community is provided with this openly accessible data set in order to help recent advancements in the processing of natural language and other ICT techniques to produce new insights in support of the ongoing fight against this infectious disease.

COVID-CHESTRAY-DATASET:

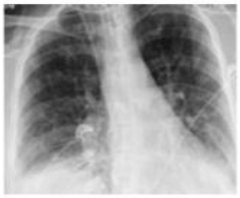
(https://github.com/ieee8023/covid-chestxray-dataset)

It is an open database of chest X-ray and CT images of patients who have COVID-19 or other viral and bacterial pneumonias (MRS, SRS, and RDS) or who are suspected of having them. Data is gathered from public sources as well as directly from hospitals and physicians.



a) Normal b) Bacteria

c) Non-covid



d) Covid-19

Text categorization (sometimes referred to as text classification) is the process of assigning pre-defined categories to free-text writings. It can provide conceptual representations of document collections and has useful real-world applications. Academic papers are frequently categorized by technical fields and sub-domains, while patient reports in healthcare organizations are frequently indexed from multiple perspectives, using taxonomies of clinical conditions, different kinds of medical procedures, insurance coverage codes, and so forth. For example, news stories are frequently organized by subject categories (topics) or geographic codes. Another frequent use of text categorization is spam filtering, which divides email communications into two groups: spam and non-spam.

Depending on the blog platform you're using, specific tags might benefit your blog in several different ways. The most commonly used tags are displayed in a larger size in tag clouds, which are plugins that some blogs use to display every tag that has been applied to an article in the sidebar. Most blog systems often provide a web page for each weblog category. What does this mean, exactly?

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit the use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Imagine that one of your practice areas is auto accidents, and that "motorcycle accidents" is one of the tags you frequently employ. The probability that search engines will find your blog increases each time you write a blog entry with that tag since it is added to a page that displays all blog entries with the tag "motorcycle wrecks."

Weblog tags are mostly used to organize your content into categories. Google believes that everything on the internet should be neat and orderly. Think of blog tags as discrete buckets that categorize posts. You wouldn't want to add tags only for the sake of adding them, though.

A particularly successful social media tactic for visual marketing may involve the use of photo tagging. It can strengthen the context of your article and aid in its popularity. Image tagging is the process of adding descriptive information to a photo when it is submitted.

You gain a sizable competitive advantage when you incorporate popular keywords into your hashtags and picture labeling. You may compile a list of hashtags that are currently trending on social media for ease of use and rapid access. You'll have a better chance of connecting with a wider audience of individuals who are already interested if you use trending hashtags.

Social media is currently the best platform for interacting with your target audience. Its impact is being used by several educational institutions to enhance admissions and placements. They don't always fully utilize digital marketing aspects like the idea of tagging, though.

In this piece, we explained how hashtags function and how organizations can use them to successfully increase their social engagement. Nowadays, more and more people use hashtag searches to get information. Scholarly brands should be aware of how to maximize this innovative research methodology.

7. STRUCTURE OF DATA USED AND CNN

The deep learning algorithm known as a CNN was created especially for computer vision problems. The CNN can learn features and rank their importance in order to recognise any input message. When compared to other methods, CNNs require the least amount of preprocessing. Prior to feeding the data into the model, the primary preprocessing in this research consisted of resizing the photos to 224x224x3. Basically, CNN shrinks the photos to make processing faster. The tricky part is to achieve this without sacrificing crucial features and traits for classification.

A generic CNN is made up of an input and output layer, and multiple hidden layers as well. Layers can be either convolution + RELU or pooling layers. The convolution operator used in the model is Conv2d along with a 2x2 filter size. The convolutional layer is also known as the kernel. Conv2d creates a convolutional kernel and then the input images are processed (convolved) on this layer to give outputs to the next layer.

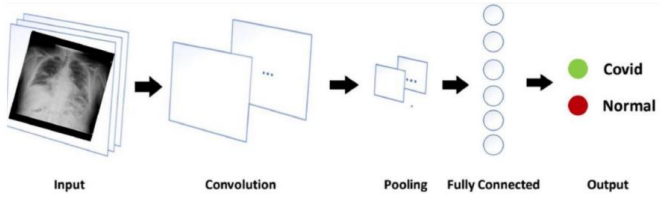


Fig 2. High level Architectural Design of The Proposed System

By reducing the number of dimensions, pooling layers helps to shrink the size of the image. It helps keep defining characteristics intact. The MaxPooling2D method is used here to return the highest value from the area of the picture that the kernel has covered (filter). The filter collects features from the pixels of the image by acting as a sliding window over them. MaxPooling eliminates characteristics that are not significant or relevant to classification by selecting the maximum of each stride.

Another CNN element that aids in speedy picture processing is dropout. Tensors are the name for the output that any convolutional layer produces. A dropout layer lessens the likelihood of overfitting by randomly dropping a portion of the tensors in each layer. Dropping entails taking a node, together with all of its inputs and outputs, momentarily out of the network.

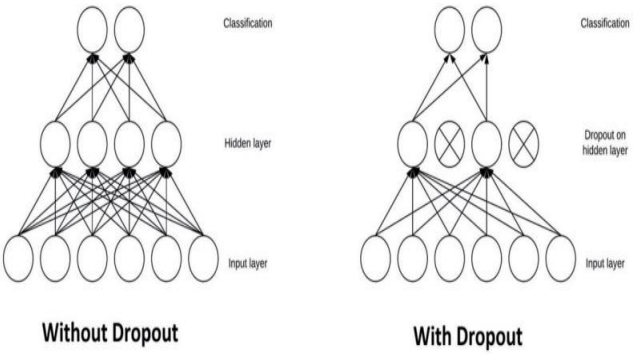


Fig 3. Difference between using and not using a Dropout layer in a CNN

The completed matrix from each convolutional layer is flattened to a one-dimensional matrix and then fed into the last layer. Each node in the dense, or completely connected, final layer receives input from each of the tensors in the preceding layer. All of the feature combinations from the previous levels are combined into one dense layer to provide outputs.

8. RESULTS AND DISCUSSIONS

8.1. Preprocessing and cleaning data

Preprocessing and cleansing of the data are necessary because we cannot feed the data straight into the model. The goal is to maintain a 50:50 class ratio, or equal representations of covid and noncovid scenarios. The Python script is displayed in the gist file that is linked below. It was created to arbitrarily choose a set number of photographs and copy those to new folders from which the data may be fed into the model while maintaining control over the class ratios.

The code is available at

https://gist.github.com/V41SH/321083bb61a8a5e9700afe9da613c755

Few screenshots of code execution are as follows:

```
In [3]: import pandas as pd
        import os
        import shutil
        import random

In [4]: # Create the data (Github dataset)
        FILE_PATH = 'covid-chestxray-dataset/metadata.csv'
        IMAGES_PATH = 'covid-chestxray-dataset/images'

In [5]: df = pd.read_csv(FILE_PATH)
        print(df.shape)

        (950, 30)

In [4]: df.head()
```

Out[4]:

	patientid	offset	sex	age	finding	RT_PCR_positive	survival	intubated	intubation_present	went_icu	...	date	location	folder
0	2	0.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	...	January 22, 2020	Cho Ray Hospital, Ho Chi Minh City, Vietnam	images
1	2	3.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	...	January 25, 2020	Cho Ray Hospital, Ho Chi Minh City, Vietnam	images
2	2	5.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	...	January 27, 2020	Cho Ray Hospital, Ho Chi Minh City, Vietnam	images
3	2	6.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	...	January 28, 2020	Cho Ray Hospital, Ho Chi Minh City, Vietnam	images
4	4	0.0	F	52.0	Pneumonia/Viral/COVID-19	Y	NaN	N	N	N	...	January 25, 2020	Changhua Christian Hospital, Changhua City, Ta...	images

5 rows × 30 columns

```
In [38]: # Now we shall consider those x-rays which are of of COVID related Pneumonia
         # and of the front(posterioanterior) view of the patient.
         # We are moving these images to the new folder created above

         count = 0
         for (i,row) in df.iterrows():
             if row["finding"] == "Pneumonia/Viral/COVID-19"  and row["view"]=="PA":
                 filename = row["filename"]
                 image_path = os.path.join(IMAGES_PATH, filename)
                 image_copy_path = os.path.join(TARGET_DIR, filename)
                 shutil.copy2(image_path, image_copy_path)
                 print("Moving image ", count, " to new folder.")
                 count += 1
```

```
Moving image  0  to new folder.
Moving image  1  to new folder.
Moving image  2  to new folder.
Moving image  3  to new folder.
Moving image  4  to new folder.
Moving image  5  to new folder.
Moving image  6  to new folder.
Moving image  7  to new folder.
Moving image  8  to new folder.
Moving image  9  to new folder.
Moving image  10  to new folder.
Moving image  11  to new folder.
Moving image  12  to new folder.
Moving image  13  to new folder.
Moving image  14  to new folder.
Moving image  15  to new folder.
Moving image  16  to new folder.
Moving image  17  to new folder.
Moving image  18  to new folder.
```

```
In [42]: COVID_PATH = 'finalDataset/covid'
         NORMAL_PATH = 'finalDataset/Normal'

         TEST_COVID_PATH = 'finalDataset/Test/covid'
         TEST_NORMAL_PATH = 'finalDataset/Test/normal'

         TRAIN_COVID_PATH = 'finalDataset/Train/covid'
         TRAIN_NORMAL_PATH = 'finalDataset/Train/normal'

In [43]: images = os.listdir(COVID_PATH)
         random.shuffle(images)

In [44]: # Splitting the data into testing and training datasets (40% - 60%)

         # Covid testing data
         for i in range(78):
             image_name = images[i]
             image_path = os.path.join(COVID_PATH, image_name)

             target_path = os.path.join(TEST_COVID_PATH, image_name)
             shutil.copy2(image_path, target_path)
             print("Copying image ",i)
```

```
Copying image  0
Copying image  1
Copying image  2
Copying image  3
Copying image  4
Copying image  5
Copying image  6
Copying image  7
Copying image  8
Copying image  9
Copying image  10
Copying image  11
Copying image  12
Copying image  13
Copying image  14
Copying image  15
Copying image  16
```

Fig 4,5,6. Few snapshots of the executed dataset creation code

8.2. Model creation and training

The code written to create and train the model is available at: https://gist.github.com/V41SH/f3901f6e008a9ab209fffa92970d2224

Screenshots of important areas of the executed code along with outputs are as follows,

```
In [4]: # CNN based model in Keras

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(Conv2D(256,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(1028,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

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

model.compile(loss=keras.losses.binary_crossentropy,optimizer='adam',metrics=['accuracy'])
```

Fig 7. Developed model in Python

```
In [5]: model.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 64)      18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 110, 110, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 108, 108, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 10, 10, 1028)      2369540
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 1028)        0
_________________________________________________________________
flatten (Flatten)            (None, 25700)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1644864
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 4,439,813
Trainable params: 4,439,813
Non-trainable params: 0
_________________________________________________________________
```

Fig 8. Summary of the developed model

The developed model comprises roughly 4.4 million parameters, as was already mentioned. The model must then be trained using the data. Since we need a data stream to input the data, we utilise tensorflow.keras.preprocessing.image. to resize the photos as well as pass the data into ImageDataGenerator in batches of 32.

```
In [5]: train_datagen = image.ImageDataGenerator(
            rescale = 1/255,
            shear_range = 0.2,
            zoom_range = 0.2,
            horizontal_flip = True
        )

        test_dataset = image.ImageDataGenerator(rescale=1/255)

In [6]: train_generator = train_datagen.flow_from_directory(
            'finalDataset/Train',
            target_size = (224,224),
            batch_size = 32,
            class_mode = 'binary'
        )

        Found 236 images belonging to 2 classes.

In [7]: train_generator.class_indices
Out[7]: {'covid': 0, 'normal': 1}

In [8]: validation_generator = test_dataset.flow_from_directory(
            'finalDataset/Test',
            target_size = (224,224),
            batch_size = 32,
            class_mode = 'binary'
        )

        Found 156 images belonging to 2 classes.

In [9]: validation_generator.class_indices
Out[9]: {'covid': 0, 'normal': 1}
```

Fig 9. Preprocessing

```
In [10]: hist = model.fit(
             train_generator,
             steps_per_epoch=8,
             epochs=20,
             validation_data = validation_generator,
             validation_steps=2
         )

Epoch 1/20
8/8 [==============================] - 48s 6s/step - loss: 0.7690 - accuracy: 0.4915 - val_loss: 0.6918 - val_accuracy: 0.5000
Epoch 2/20
8/8 [==============================] - 46s 6s/step - loss: 0.7050 - accuracy: 0.5593 - val_loss: 0.6874 - val_accuracy: 0.9531
Epoch 3/20
8/8 [==============================] - 59s 7s/step - loss: 0.6835 - accuracy: 0.5424 - val_loss: 0.6455 - val_accuracy: 0.5000
Epoch 4/20
8/8 [==============================] - 50s 6s/step - loss: 0.6018 - accuracy: 0.6695 - val_loss: 0.4634 - val_accuracy: 0.8750
Epoch 5/20
8/8 [==============================] - 52s 6s/step - loss: 0.5818 - accuracy: 0.7076 - val_loss: 0.4499 - val_accuracy: 0.8281
Epoch 6/20
8/8 [==============================] - 49s 6s/step - loss: 0.5392 - accuracy: 0.7627 - val_loss: 0.6972 - val_accuracy: 0.8281
Epoch 7/20
8/8 [==============================] - 40s 5s/step - loss: 0.4839 - accuracy: 0.8347 - val_loss: 0.3427 - val_accuracy: 0.8906
Epoch 8/20
8/8 [==============================] - 40s 5s/step - loss: 0.4134 - accuracy: 0.8178 - val_loss: 0.4349 - val_accuracy: 0.9062
Epoch 9/20
8/8 [==============================] - 39s 5s/step - loss: 0.4431 - accuracy: 0.8559 - val_loss: 0.3325 - val_accuracy: 0.9219
Epoch 10/20
8/8 [==============================] - 38s 5s/step - loss: 0.4316 - accuracy: 0.8093 - val_loss: 0.2887 - val_accuracy: 0.8906
Epoch 11/20
8/8 [==============================] - 35s 4s/step - loss: 0.3174 - accuracy: 0.8771 - val_loss: 0.1879 - val_accuracy: 0.9375
Epoch 12/20
8/8 [==============================] - 40s 5s/step - loss: 0.2050 - accuracy: 0.9322 - val_loss: 0.1910 - val_accuracy: 0.9375
Epoch 13/20
8/8 [==============================] - 40s 5s/step - loss: 0.1862 - accuracy: 0.9661 - val_loss: 0.0960 - val_accuracy: 0.9688
Epoch 14/20
8/8 [==============================] - 39s 5s/step - loss: 0.1061 - accuracy: 0.9788 - val_loss: 0.2371 - val_accuracy: 0.9375
Epoch 15/20
8/8 [==============================] - 45s 6s/step - loss: 0.1133 - accuracy: 0.9619 - val_loss: 0.0998 - val_accuracy: 0.9375
Epoch 16/20
8/8 [==============================] - 37s 5s/step - loss: 0.2063 - accuracy: 0.9492 - val_loss: 0.1286 - val_accuracy: 0.9375
Epoch 17/20
8/8 [==============================] - 37s 5s/step - loss: 0.1411 - accuracy: 0.9661 - val_loss: 0.0641 - val_accuracy: 0.9844
Epoch 18/20
8/8 [==============================] - 36s 5s/step - loss: 0.1455 - accuracy: 0.9619 - val_loss: 0.0930 - val_accuracy: 0.9688
Epoch 19/20
8/8 [==============================] - 44s 6s/step - loss: 0.1337 - accuracy: 0.9661 - val_loss: 0.1107 - val_accuracy: 0.9688
Epoch 20/20
8/8 [==============================] - 37s 5s/step - loss: 0.1086 - accuracy: 0.9703 - val_loss: 0.0548 - val_accuracy: 0.9844
```

Fig 10. Training (along with accuracy scores)

For a better understanding of the accuracies, we can use model.evaluate_generator which considers the inputs and predicted outputs and then returns the accuracy score.

```
In [23]: model.evaluate_generator(train_generator)
Out[23]: [0.12600766122341156, 0.9618644118309021]

In [24]: model.evaluate_generator(validation_generator)
Out[24]: [0.10991428792476654, 0.9743589758872986]
```

Fig 11. Evaluate_generator output

As we can see from the above output, the model has about 96% accuracy on the training data and 97.4% accuracy on the testing data.

8.3. Predictions And Confusion Matrix

A classification problem's predictions can be visualized well using a confusion matrix. False Positive (FP), True Negative (TN), False Negative (FN), and True Positive (TP) are all represented in the top left sector, respectively (TN). The True Positive and True Negative sections would have a high score on the ideal model, whereas the other two would have a score of 0. High TP and TN areas and low FP and FN areas are characteristics of a good model.

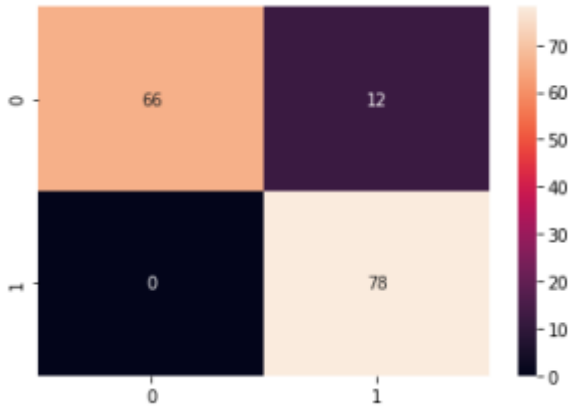


Fig 12. Confusion matrix

8.4 Discussion

As shown above, the confusion matrix provides a visualization through which we can deduce that the model is performing well. The FP and FN areas are very less in comparison to the TP and TN areas, which is a good sign. Thus, we can conclude that the model is trained to a satisfiable extent and is accurate enough in its predictions. We learnt how Convolutional Neural Networks work completely from scratch including its various intricacies. Creating our own model from scratch taught use more than we could form using any of the pretrained models.

9. CONCLUSIONS AND FUTURE WORK

Doctors can always appreciate a second opinion to aid in the diagnosis of a condition, and programs and models like this could prove beneficial to quickly identify ailnesses. Obviously, to actually get such models to a production level, many experiments, testing and consultations with experts are required. The model would have to be tested for accuracy on not just the dataset we have used here, but a relatively larger dataset. Once the dataset size increases, this model might require more tweaking in order to fine tune the quirks to produce the best results.

In conclusion, a model to predict Covid-19 induced pneumonia from chest x-rays was developed with an accuracy of above 98.4%.

10. REFERENCES

[1] Apostolopoulos ID, Mpesiana TA. Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. Phys Eng Sci Med. 2020 doi:10.1007/s13246-020-00865-4.
[2] Bukhari SUK, Bukhari SSK, Syed A, Shah SSH. The diagnostic evaluation of convolutional neural network (CNN) for the assessment of chest X-ray of patients infected with covid-19. 2020; medRxiv 2020.03.26.20044610.
[3] World Health Organization: Covid-19 pandemic. https://www.who.int/emergencies/diseases/novel-coronavirus-2019.
[4] Chowdhury MEH, Rahman T, Khandakar A, Mazhar R, Kadir MA, Mahbub ZB, Islam KR, Khan MS, Iqbal A, Al-Emadi N, Reaz MBI. Can ai help in screening viral and covid-19 pneumonia? arXiv:2003.13145; 2020.
[5] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition (CVPR); 2016.P.770–778. 10.1109/CVPR.2016.90. https://arxiv.org/abs/2003.12338
[6] Ghoshal B, Tucker A. Estimating uncertainty and interpretability.
[7] In deep learning for coronavirus (covid-19) detection. arXiv:2003.10769; 2020. Davies HE, Wathen CG, Gleeson FV. The risks of radiation.
[8] Exposure related to diagnostic imaging and how to minimise them. Bmj. 2011;342:d947.doi: 10.1136/bmj.d947.Luz E, Silva PL, Silva R, Silva L, Moreira G, Menotti D. Towards
[9] Luz, E., Silva, P., Silva, R., Silva, L., Guimarães, J., Miozzo, G., Moreira, G. and Menotti, D., 2021. Towards an effective and efficient deep learning model for covid-19 patterns detection in x-ray images. Research on Biomedical Engineering, pp.1-14.

11. APPENDIX

iPython notebooks with outputs:
https://drive.google.com/drive/folders/13IE8wbhEOcaG_UkEAnvtjyp3IP_f8IzL?usp=sharing

Python code:
https://gist.github.com/V41SH/321083bb61a8a5e9700afe9da613c755
https://gist.github.com/V41SH/f3901f6e008a9ab209fffa92970d2224