Designing a Convolutional Neural Network (CNN) to Detect COVID-19 on Chest X-Ray Photos

Darren George and Jonathan Joshua
Prof. Hong Man
CPE 462: Intro to Image Processing & Coding
May 2021
Final Project Report

Abstract

Coronavirus has resulted in the deaths of over 3 million individuals as of May 2021. One of the biggest needs is the ability to quickly detect the virus in patients. Chest X-Rays can be taken of COVID patients, especially those who might have other infections that lead them to be at serious medical risk. These X-Ray images allow doctors to identify irregular, patchy, hazy, and/or reticular glass opacities which can aid in the diagnosis process[5].

We decided to utilize machine learning in hopes of detecting COVID in chest X-Ray images. Having the ability to detect COVID with a high level of accuracy in a timely manner can be useful for those caring for patients. We designed a Convolutional Neural Network (CNN) to learn and train on a dataset of various chest X-Ray images and segment the image to gain useful information in identifying whether an X-Ray image was one of a COVID patient. Various tools and services were used such as Python, Numpy, Keras, Kaggle, and Google Colab.

Using 2 different datasets we were able to achieve 77%-96% accuracy using our CNN design. The design can be altered to yield higher accuracy, and one paper mentions a team that created a neural network that achieved over 98% accuracy. The ability to use technology such as machine learning to perform image segmentation is one of great necessities in the field of healthcare.

Background of Neural Networks

The human brain has been studied for thousands of years and while we continue to discover more about our planet and the universe, we are yet to solve many questions regarding the brain. One of the more recent discoveries about our brain that is changing the way we solve complex problems is how the neuron works.

Neurons are the fundamental units of the brain and nervous system. They are cells that are electrically excitable and receive, process, and transmit information through electrical and chemical signals. Our brains have over 100 billion neurons and they all communicate vital information to each other[1]. For example, when you touch a stove, neurons in your brain receive

information that the body is in connection with something hot and sends stimuli to the nervous system so your muscles retract from the stove.

A neuron's ability to process information, alter its decision-making, and share them with billions of other neurons to result in an ultimate decision is a process that is now being mimicked by researchers to solve complex problems in the world around us, and this process is known as artificial neural networks.

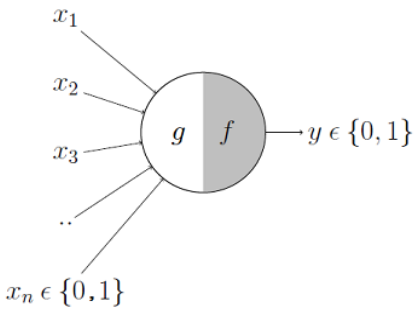


Figure 1: First Neural Network Circuit[2]

CNN's Advantages for Image Segmentation

Similar to the numerous functions of the brain when processing information, Convolutional Neural Networks (CNNs) have multiple layers that allow for a large data set to be processed and to yield accurate results. CNNs excels in pattern detection and recognition, which makes it useful for image segmentation. This advantage is further outlined when comparing it to a multilayer perceptron (MLP), a class of feedforward artificial neural network[7]. A CNN utilizes hidden layers known as convolutional layers. Similar to other layers, a convolutional layer intakes input, manipulates the input, and then passes the variant to the next layer. This process is known as a convolution operation.

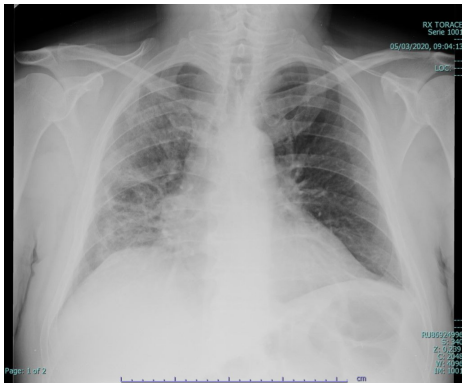
Convolution simply acts like a general purpose filter. Convolution is a mathematical way of multiplying together 2 arrays of numbers. It is commonly used in many applications because it helps one understand a system's behavior based on current and past events. CNN layers are able to detect patterns by implementing different numbers of filters different layers can have[4]. In image processing it can be used to blur, sharpen, enhance horizontal edges, etc. on an image. Different layers may utilize different filters, such as one filter can be an edge detector while another filter might be a circle detector. Basic shapes are the foundation of the beginning of our CNN, only becoming more sophisticated the deeper our network goes (in our case, specific portions of the lung X-Ray).

Problem

One of the hallmarks of the medical field is that prevention is always better than the cure. Of course with this being a worldwide pandemic with no signs of slowing down, a need for rapid testing has arisen, especially in parts of the world that do not have access to these tools or might be too costly. With our project, we wanted to take this approach to tackling this problem by implementing a CNN that can determine if a person is positive for COVID with high accuracy.

The dataset we decided to use to train our CNN was a compilation of X-Rays from both COVID positive and negative patients. The idea is that the CNN would train on enough data to learn how to detect between different lung images. Looking at the two images below, you might be able to identify which pair of lungs indicate covid positivity. The lungs on the left are normal lungs, whereas the one on the right are the covid positive lungs. Just at a quick glance, you can see just how drastic the difference is between the two lungs knowing the damage COVID has done to them. The cloudy appearance of the opacities near the ribs are common for patients with COVID, and we hope a CNN implementation would pick up details to aid in the identification process.

Normal Lung X-Ray



COVID-19 Lung X-Ray

What's great about this method over traditional rapid testing is that in addition to knowing if you have covid or not, you would also be able to determine how extensive the damage is. This is critical for severely ill patients that might be dealing with COVID along with other illnesses such as pneumonia. An improvement to our CNN design would be to include a

metric for the severity if a lung image is analyzed to have COVID-19. This application of artificial intelligence into the medical field could truly bring about the scalability that prevention didn't have before a cure even needs to be considered.

Approach

Github Link to the code:
https://github.com/dgeorge1000/COVID-19-CNN-Image-Detection-from-Lung-X-Rays

The first step in creating a CNN to detect COVID vs. non COVID lungs is to collect a dataset of x-ray images of both classifications. We tested our model on 2 different data sets which both contained chest X-Rays of patients with and without COVID. Both datasets were found on Kaggle, which is an online community of data scientists that provide validated data. The datasets we used are all authorized by a team which includes medical experts that authenticate that these are real chest X-Ray images.

The first dataset was found from the following link[3]:
`https://www.dropbox.com/sh/7hrnecm53hk0ih3/AABsfKTooQP0Tc3QddOQAfZJa?dl=0`

It contains over 5,000 chest X-Ray images of various diseases and infections. We extracted over 250 COVID and non COVID chest X-Ray images and broke it down into train and validation folders which each contained a folder of COVID and normal lung chest X-Rays. This keeps the data organized so the CNN can easily train on the train folder and validate its learning using the data in the validation folder. The Dropbox folder can be found in the following link:
`http://cb.lk/covid_19`

The second data set was downloaded from the following link[6]:
https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

We were able to extract over 6,000 COVID and non COVID X-Ray images. This data has been collected from various countries and medical institutions so the detecting differences in the varying images are more complex. We wanted to use this to test the model on a rather difficult differentiating dataset.

Below is a description of how the code works and the actual code that extracts the data, creates the CNN, and tests the model.

The code first downloads and extracts the dataset from the Dropbox folder. The CNN is built using Keras which is an open source library commonly used in building neural networks. We used a Sequential model type that allows us to build the model layer by layer. The first layer is a Conv2D layer that uses convolution to process the input images. 32 is the number of "neurons" in the layer and the kernel size is the size of the filter matrix. In this case it is a 3x3 filter map that will be convoluted on the image. We use ReLU activation function which will take an input and given a certain weight will produce a new output which will be the new input for the next layer of the neural network. The input shape is a rather arbitrary value of 224x224 pixels with 3 dimensions (RGB).

We continue to add more layers and then use max pooling. Max pooling will select the maximum output neurons covered by the filter. The output will be a map containing the most prominent features. Finally, a drop out is used to prevent overfitting by removing random neurons. This is because convolution develops co-dependency amongst nearby elements and we want the maximum output neurons to contribute predominantly to the output and not have lower output neurons that appear dependent and relevant to interfere with the classification process.

In the end we use flatten layers to convert the data into a 1D array and dense layers which ensures every input neuron is connected to every output neuron by a certain weight. We compile the model and use Adam optimization which is a commonly used optimization algorithm to change the attributes in the CNN such as weights and learning rates.

The code can be run by accessing the team's GitHub repository shown above, opening the "COVID19_Detector.ipynb" file, clicking on the Google Colab, and going to "Runtime->Run All". Scroll down and the results will be shown. To run the second dataset, change the dataset link after "!wget " and "!unzip" with the link mentioned above, and for all the paths change "CovidDataset" to "Dataset". Follow the same run steps as previously mentioned. NOTE, opening .ipynb in GitHub can result in "Sorry, something went wrong" errors. This can be because of GitHub's backend, size of the file, or network connection. Try reloading, if that does not work then the code can be seen in "COVID19_Detector.py"

```
# various libraries and imports needed
import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.layers import *
from keras.models import *
from keras.preprocessing import image

# Dataset #1: http://cb.lk/covid_19
# Dataset #2: https://www.dropbox.com/sh/7hrnecm53hk0ih3/AABsfKTooQP0Tc3QddOQAfZJa?dl=0

# download the dataset from the Dropbox link
!wget http://cb.lk/covid_19

# unzip the folder to get each image for local access
!unzip covid_19

# paths for both the training dataset and the validation set
# training will be used for the CNN to learn how to differentiate between COVID and non
COVID lung images
# validation will be used to test the CNN on data not seen to see its accuracy
TRAIN_PATH = "CovidDataset/Train"
VAL_PATH = "CovidDataset/Val"

# CNN Based Model using 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(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'])

# get the general information about the CNN created
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
_________________________________________________________________
dropout (Dropout)            (None, 110, 110, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 108, 108, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 26, 26, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 86528)             0
_________________________________________________________________
dense (Dense)                (None, 64)                5537856
```

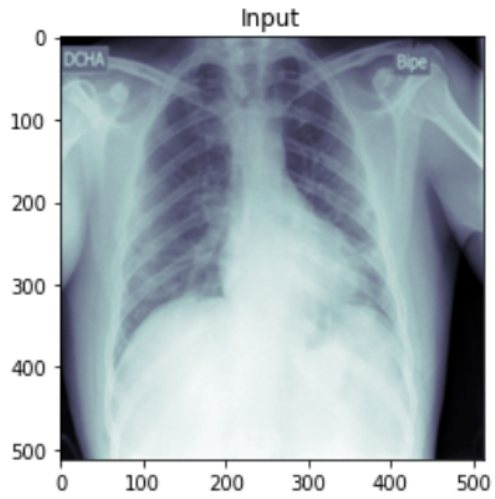
```
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 5,668,097
Trainable params: 5,668,097
Non-trainable params: 0
_________________________________________________________________
'''

# Train from scratch
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)

# using generator saves memory
train_generator = train_datagen.flow_from_directory(
    'CovidDataset/Train',
    target_size = (224,224),
    batch_size = 32,
    class_mode = 'binary')

'''
Found 224 images belonging to 2 classes.
'''

# 2 classifications
train_generator.class_indices

'''
{'Covid': 0, 'Normal': 1}
'''

validation_generator = test_dataset.flow_from_directory(
    'CovidDataset/Val',
    target_size = (224,224),
    batch_size = 32,
    class_mode = 'binary')
```

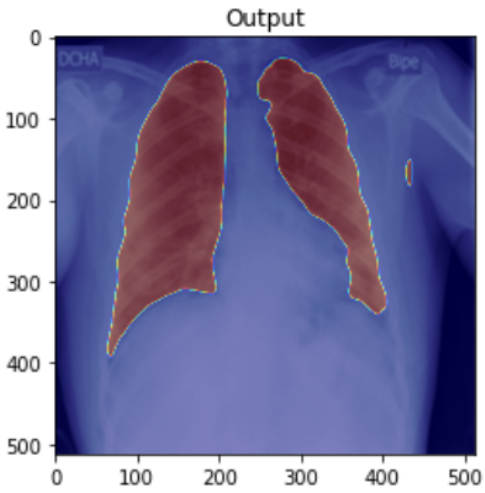
```
'''
Found 60 images belonging to 2 classes.
'''

# CNN will begin training and testing on the given data
# "accuracy" and "val_accuracy" will show how accurate the CNN is
hist = model.fit_generator(
    train_generator,
    steps_per_epoch=5,
    epochs = 8,
    validation_data = validation_generator,
    validation_steps=2
)
```

The second code called "xray_check.ipynb" was taken from the following GitHub repository: https://github.com/rani700/xray

The purpose of this code is to demonstrate visually the image segmentation of a chest x-ray. The code has been added to our GitHub repository in the link below. https://github.com/dgeorge1000/COVID-19-CNN-Image-Detection-from-Lung-X-Rays/blob/main/xray_check.ipynb

By running the code in Google Colab you can upload any image and see the image segmentation take place. NOTE, opening .ipynb in GitHub can result in "Sorry, something went wrong" errors. This can be because of GitHub's backend, size of the file, or network connection. Try reloading, if that does not work then the code can be seen in "xray_check.py". Below are images of the input and output images after the code runs that simply detected the non organ portions of the lung X-Ray.





Results

Below is the output of the performance for one of the tests we ran on the first dataset. Epochs (number of passes of the training dataset) and its steps can be altered to indicate how many runs to try. As seen the first epoch ended when a validation, or test, accuracy of 61.67%. This is expected as the CNN is trying to learn how to differentiate between the 2 lung classifications so it should be close to 50% (blind guess). As the more training occurs, the CNN becomes more accurate with the final epoch having a validation value of 96.67%.

```
# CNN will begin training and testing on the given data
# "accuracy" and "val_accuracy" will show how accurate the CNN is
hist = model.fit_generator(
    train_generator,
    steps_per_epoch=5,
    epochs = 8,
    validation_data = validation_generator,
    validation_steps=2
)
```

```
Performance shown below.
Epoch 1/8
5/5 [==============================] - 8s 2s/step - loss: 0.6761 - accuracy: 0.6000 - val_loss: 0.6456 - val_accuracy: 0.6167
Epoch 2/8
5/5 [==============================] - 7s 1s/step - loss: 0.6452 - accuracy: 0.6500 - val_loss: 0.6264 - val_accuracy: 0.9167
Epoch 3/8
5/5 [==============================] - 7s 2s/step - loss: 0.5789 - accuracy: 0.6750 - val_loss: 0.5282 - val_accuracy: 0.9500
Epoch 4/8
5/5 [==============================] - 7s 2s/step - loss: 0.4540 - accuracy: 0.7688 - val_loss: 0.3136 - val_accuracy: 0.9500
Epoch 5/8
5/5 [==============================] - 7s 2s/step - loss: 0.3411 - accuracy: 0.8687 - val_loss: 0.3142 - val_accuracy: 0.9333
Epoch 6/8
5/5 [==============================] - 7s 2s/step - loss: 0.2728 - accuracy: 0.9187 - val_loss: 0.1579 - val_accuracy: 0.9500
Epoch 7/8
5/5 [==============================] - 7s 2s/step - loss: 0.2119 - accuracy: 0.9375 - val_loss: 0.2037 - val_accuracy: 0.9833
Epoch 8/8
5/5 [==============================] - 7s 2s/step - loss: 0.2014 - accuracy: 0.9062 - val_loss: 0.1365 - val_accuracy: 0.9667
```

We tested the code on the second dataset which contains more images with increasing difficulty to differentiate between COVID and none COVID lung images (due to irregularities and different medical institutions X-Ray documentation). The final accuracy was 77.75% which is less than the first dataset. While ~78% does not seem bad, a real world application of an ML algorithm in healthcare must have much higher accuracy. However, for our applications we are satisfied with the CNN design although it can be altered to be more precise with on difficult X-Ray images.

```
# CNN will begin training and testing on the given data
# "accuracy" and "val_accuracy" will show how accurate the CNN is
hist = model.fit_generator(
    train_generator,
    steps_per_epoch=5,
    epochs = 8,
    validation_data = validation_generator,
    validation_steps=2
)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/8
5/5 [==============================] - 2s 491ms/step - loss: 0.4652 - accuracy: 0.7566 - val_loss: 0.6236 - val_accuracy: 0.6094
Epoch 2/8
5/5 [==============================] - 2s 483ms/step - loss: 0.4851 - accuracy: 0.7625 - val_loss: 0.6789 - val_accuracy: 0.6625
Epoch 3/8
5/5 [==============================] - 2s 490ms/step - loss: 0.4479 - accuracy: 0.7437 - val_loss: 0.5803 - val_accuracy: 0.6862
Epoch 4/8
5/5 [==============================] - 2s 488ms/step - loss: 0.4698 - accuracy: 0.7750 - val_loss: 0.6331 - val_accuracy: 0.6938
Epoch 5/8
5/5 [==============================] - 2s 483ms/step - loss: 0.4398 - accuracy: 0.7937 - val_loss: 0.6417 - val_accuracy: 0.7156
Epoch 6/8
5/5 [==============================] - 2s 491ms/step - loss: 0.5280 - accuracy: 0.7563 - val_loss: 0.6334 - val_accuracy: 0.7406
Epoch 7/8
5/5 [==============================] - 2s 483ms/step - loss: 0.4607 - accuracy: 0.7875 - val_loss: 0.6216 - val_accuracy: 0.7094
Epoch 8/8
5/5 [==============================] - 2s 489ms/step - loss: 0.4036 - accuracy: 0.7875 - val_loss: 0.6374 - val_accuracy: 0.7775
```

Conclusions and Contribution Breakdown

The use of chest X-Rays in detecting COVID and other diseases can be helpful for patients that need quick diagnosis. The ability to use image segmentation through artificial intelligence can be a great solution in quickly processing chest X-Rays with high levels of accuracy in a short period of time.

We were able to successfully apply machine learning, specifically a Convolutional Neural Network, that was able to detect COVID in chest X-Rays with an accuracy ranging from ~77-96% accuracy. One team of researchers designed their own neural network and achieved 98.08% accuracy[5]. An improvement to the project would be to include the severity of a lung that is identified to be COVID positive. This could aid doctors in diagnosing patients in a critical condition more effectively. While our design can be improved, we are satisfied with our work and have demonstrated the power technology can have to help mitigate even the most dire situations.

Contribution Breakdown:

Both members helped in the creation of the project and regular meetings were held to collaborate effectively. Below are some of the work we primarily focussed on:

Darren- CNN design, dataset acquisition, aided in research

Jon- Model results testing, project objectives, CNN research

Works Cited

[1] Herculano-Houzel, Suzana. “The Human Brain in Numbers: A Linearly Scaled-Up Primate Brain.” *Frontiers in Human Neuroscience*, vol. 3, 2009, doi:10.3389/neuro.09.031.2009.

[2] McCulloch, Warren, and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity (1943).” *Ideas That Created the Future*, 2021, pp. 79–88., doi:10.7551/mitpress/12274.003.0011.

[3] Mooney, Paul. “Chest X-Ray Images.” *Kaggle*, 24 Mar. 2018, www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

[4] *Neural Networks - History*, cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html.

[5] Osman, Ahmed Hamza, et al. “SOM-LWL Method for Identification of COVID-19 on Chest X-Rays.” *PLOS ONE*, vol. 16, no. 2, 2021, doi:10.1371/journal.pone.0247176.

[6] Rahman, Tawsifur. “COVID-19 Radiography Database.” *Kaggle*, 6 Mar. 2021, www.kaggle.com/tawsifurrahman/covid19-radiography-database.

[7] “Understanding of Convolutional Neural Network (CNN) - Deep Learning.” *Medium*, Medium, 21 Nov. 2019, medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.