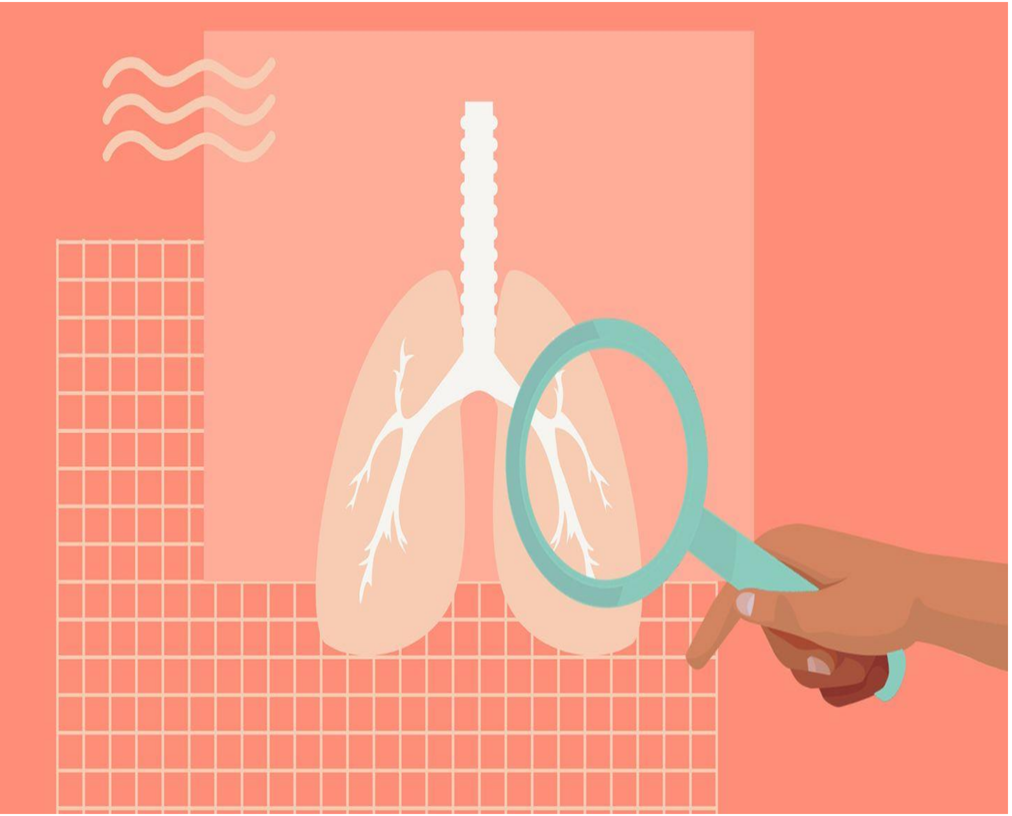
# Image classification of Covid 19 scans with neural network using CNN

# Table of Contents

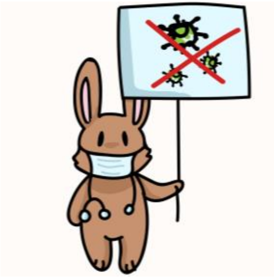


# INTRODUCTION & RESEARCH QUESTION

# Background

During the peak time of COVID-19 outbreak, RT-PCR test kits were in great shortage.

Among them, CT scans are considered useful for screening and diagnosing COVID-19.

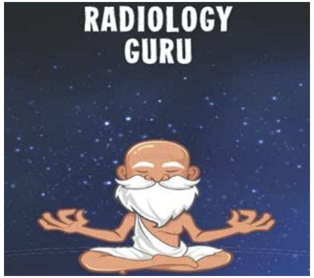


As a result, many suspected cases cannot be diagnosed in time.

- radiologists are highly occupied
- Unable to read in time


To mitigate the shortage of PCR test kits, hospitals have been utilizing alternative diagnosis methods.

- Radiologists in rural regions
- not be well-trained to recognize



# Introduction

To address these problems, we applied a deep learning method to screen COVID-19 from CTs.

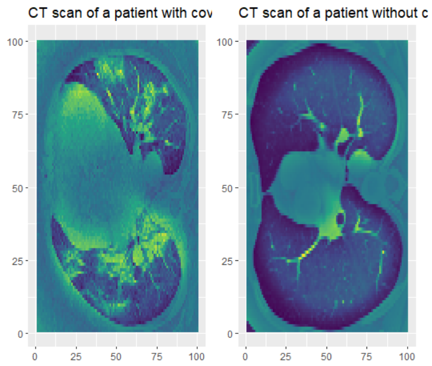
By using CNN, we extract patterns of image data, train an effective model, so that radiologists are able to make accurate diagnosis with the help of the neural network model.

# Research Question

**To what extent can we identify if a person is infected by COVID-19 by analyzing their CT scans through neural network (CNN) ?**

# DATA DESCRIPTION & PREPARATION

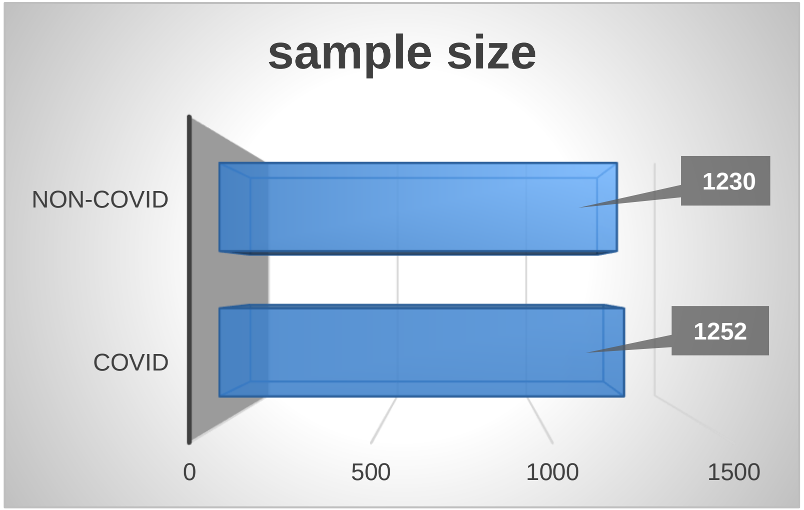
# Dataset for training




SARS-COV-2 Ct-Scan Dataset


1252 positive CT scans
1230 non-infected CT scans
2482 in total

Collected from real patients in hospitals from Sao Paulo, Brazil



# Dataset for testing

COVID-19 Lung CT Scans

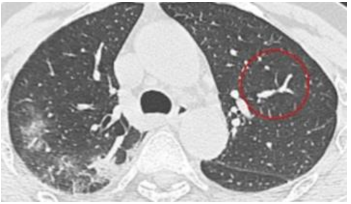
kaggle

Containing 349 CT scans that are positive for COVID-19, and 397 scans without

Sample size

Collected from COVID19-related papers from medRxiv, bioRxiv, NEJM, JAMA, Lancet, etc.

resource



Covid CT

Non-covid CT

# Data Preparation

## Image preprocessing

| | |
|---|---|
| 01 | Greyscale images |
| 02 | Resize Images to 100 x100 pixels |
| 03 | Convert images to Array Objects |
| 04 | Normalise Data |
| 05 | Merge, Label & Shuffle Data |

## Data After Image Preprocessing

- ❑ 2,481 Entries
- ❑ 10001 total columns

covid_reshaped

Filter Cols: « < 1 - 50 > »

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.0336811737 | 0.034831883 | 0.033244204 | 0.032768851 | 0.032763363 | 0.031887075 | 0.031085535 | 0.031606672 | 0.030419752 | 0.030212722 | 0.029869940 | 0.029221393 | 0.0289 |
| 2 | 0.0361629444 | 0.034847694 | 0.033514018 | 0.030818690 | 0.033024625 | 0.032663001 | 0.031349344 | 0.030341254 | 0.031033530 | 0.030577576 | 0.030110677 | 0.029923108 | 0.0298 |
| 3 | 0.0400628698 | 0.034768639 | 0.032559291 | 0.032512925 | 0.033275840 | 0.030457132 | 0.030968286 | 0.031156106 | 0.029335412 | 0.030058360 | 0.028187866 | 0.029388468 | 0.0297 |
| 4 | 0.0366947524 | 0.035132294 | 0.034759313 | 0.033951233 | 0.032552343 | 0.032509793 | 0.033855533 | 0.033797185 | 0.031420682 | 0.030872265 | 0.031522694 | 0.031476904 | 0.0322 |
| 5 | 0.0386447151 | 0.035940418 | 0.036238102 | 0.034585931 | 0.036035847 | 0.033513059 | 0.034344069 | 0.031611465 | 0.030819495 | 0.031587941 | 0.032121451 | 0.031242999 | 0.0314 |
| 6 | 0.0366947524 | 0.036537727 | 0.035949262 | 0.034877687 | 0.034486692 | 0.033433984 | 0.033801794 | 0.033659779 | 0.033007060 | 0.031237120 | 0.029485686 | 0.030875435 | 0.0315 |
| 7 | 0.0352765978 | 0.035072564 | 0.034901139 | 0.035875801 | 0.035637254 | 0.034954534 | 0.034827719 | 0.032568516 | 0.033863201 | 0.031658105 | 0.032149228 | 0.030925557 | 0.0307 |
| 8 | 0.0352765978 | 0.036026501 | 0.035217651 | 0.036382536 | 0.033562224 | 0.035307078 | 0.034466202 | 0.031734491 | 0.032495579 | 0.032289583 | 0.032393052 | 0.031410074 | 0.0286 |
| 9 | 0.0370492911 | 0.037451258 | 0.034920164 | 0.036131728 | 0.035697545 | 0.033280776 | 0.034324527 | 0.035321438 | 0.032643515 | 0.035025988 | 0.031670840 | 0.028619924 | 0.0322 |
| 10 | 0.0368720218 | 0.035148106 | 0.034930542 | 0.036484907 | 0.035536768 | 0.033928204 | 0.032258021 | 0.032325659 | 0.031916425 | 0.031082758 | 0.029819015 | 0.030474455 | 0.0304 |
| 11 | 0.0377583685 | 0.037305444 | 0.035608536 | 0.035568689 | 0.035345845 | 0.034147308 | 0.033728513 | 0.033132522 | 0.030918643 | 0.032093123 | 0.031028874 | 0.031343244 | 0.0300 |
| 12 | 0.0384674458 | 0.036444617 | 0.035734795 | 0.035732482 | 0.037963498 | 0.036157136 | 0.032477862 | 0.033159684 | 0.033479197 | 0.033159620 | 0.033129152 | 0.032078374 | 0.0299 |
| 13 | 0.0361629444 | 0.034315387 | 0.033441376 | 0.035235985 | 0.035546816 | 0.034914997 | 0.031598497 | 0.034097562 | 0.032697024 | 0.030956463 | 0.031761888 | 0.033465095 | 0.0301 |
| 14 | 0.0120543148 | 0.011951445 | 0.014336466 | 0.038460661 | 0.029768889 | 0.032826094 | 0.030880350 | 0.030509018 | 0.029881517 | 0.030647740 | 0.029896174 | 0.028920659 | 0.0287 |
| 15 | 0.0381129071 | 0.036377859 | 0.036208699 | 0.034985176 | 0.035044388 | 0.036343293 | 0.033464704 | 0.033110153 | 0.031828293 | 0.031882631 | 0.030678570 | 0.033064115 | 0.0298 |

Showing 1 to 15 of 1,252 entries, 10000 total columns

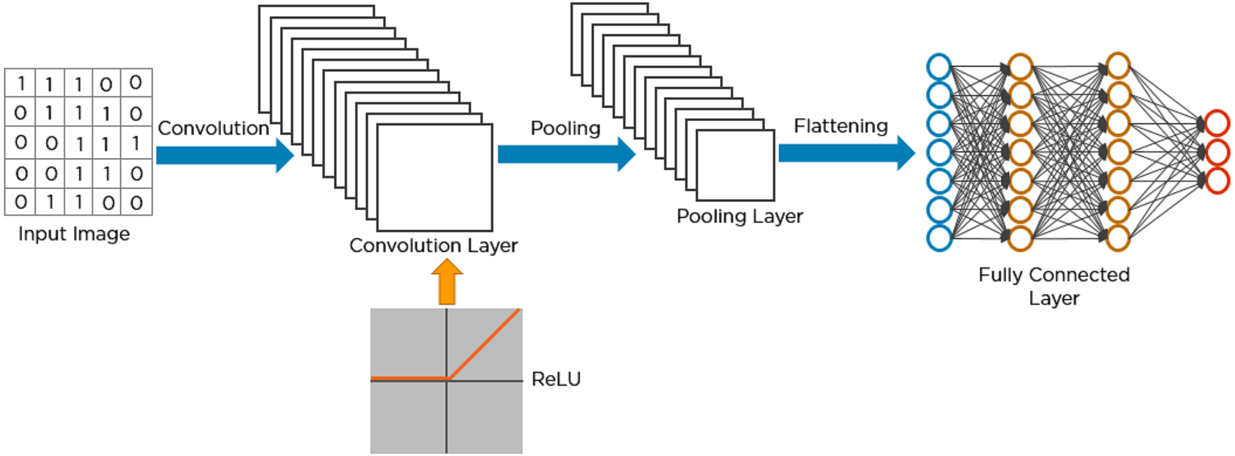
Figure1. Visual representation of images to array object conversion

# METHODS

# Methodology

A neural network is a series of algorithms that attempts to recognize underlying patterns in a set of data through a process that mimics the way the human brain works.

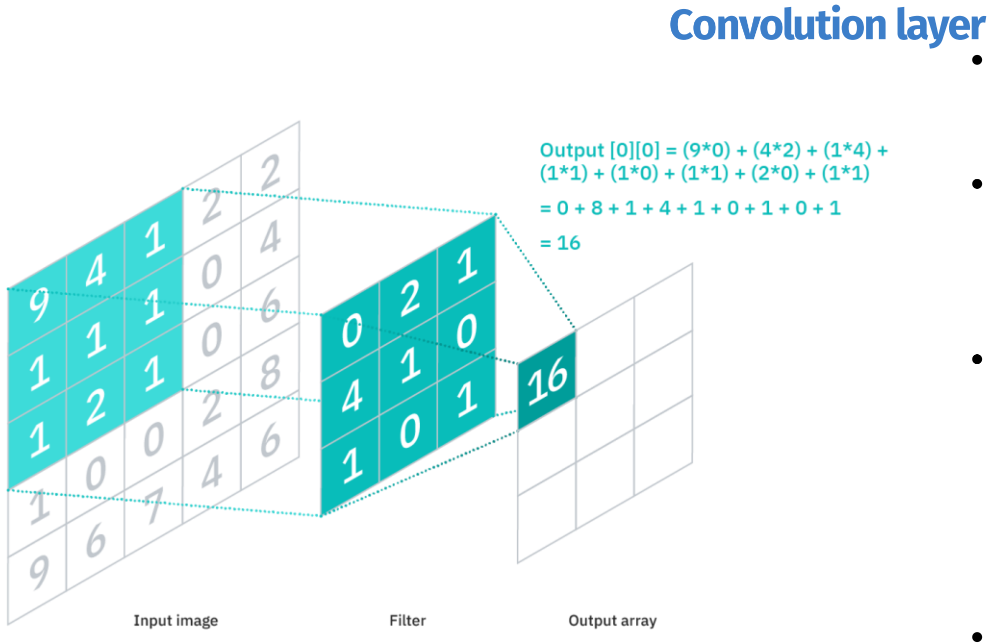
A Convolutional Neural Network, also known as CNN, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image.




A CNN generally has three main types of layers, which are:

1. Convolution layer
2. Pooling layer
3. Fully connected layer

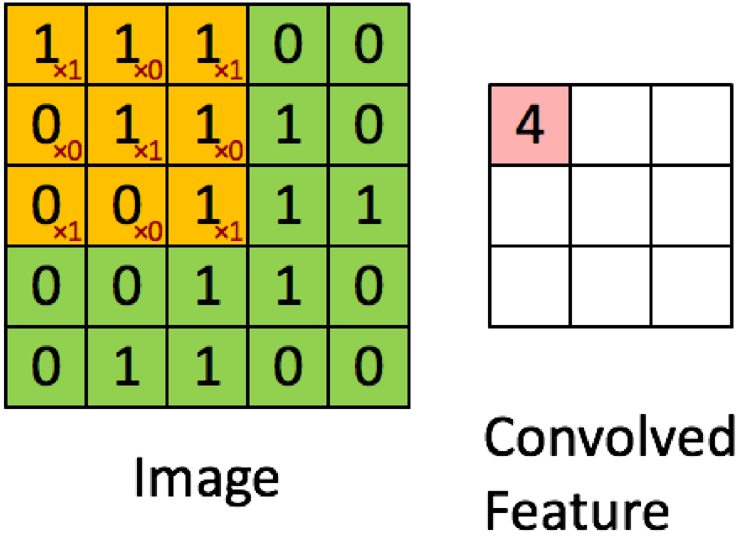
# Convolutional Neural Network




- In a CNN, every input image is considered as a matrix of values.
- A convolution layer has several filters that perform the convolution operation.
- Over multiple iterations, the filter/kernel sweeps over the entire image and after each iteration a dot product is calculated between the input pixels and the filter.
- The final output is known as feature map or convolved feature.

# Convolutional Neural Network

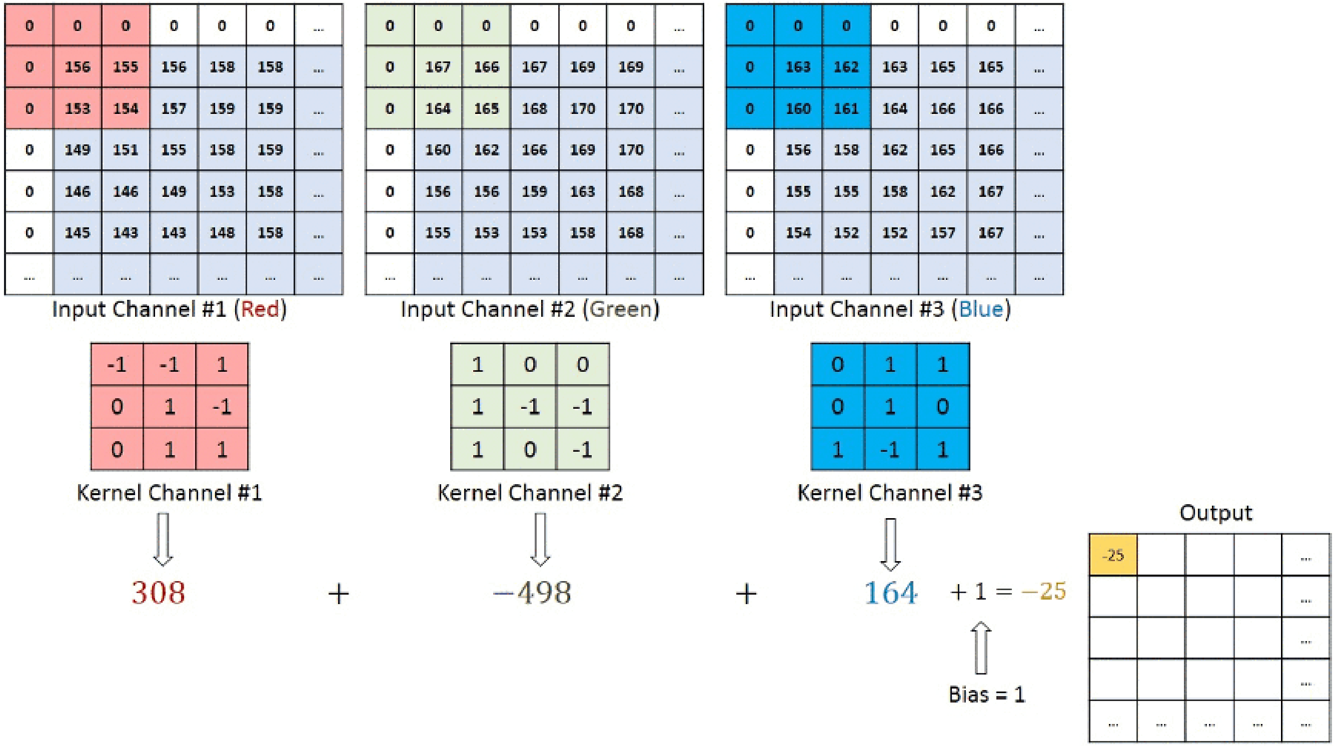
## Convolution layer




A 3x3 filter moving across each 3x3 set of pixels of the input, checking if a feature is present in the image, until it has slid over every 3x3 block of pixels from the entire image.

# Convolutional Neural Network

## Convolution layer



# Convolutional Neural Network

ReLU Layer

**Filter 1 Feature Map**

| | | | |
|---|---|---|---|
| 9 | 3 | 5 | -8 |
| -6 | 2 | -3 | 1 |
| 1 | 3 | 4 | 1 |
| 3 | -4 | 5 | 1 |

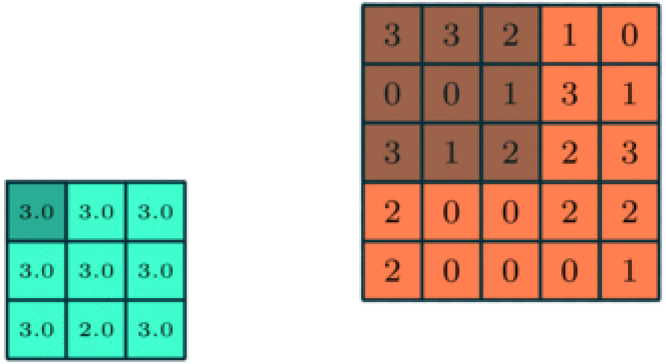
| | | | |
|---|---|---|---|
| 9 | 3 | 5 | 0 |
| 0 | 2 | 0 | 1 |
| 1 | 3 | 4 | 1 |
| 3 | 0 | 5 | 1 |

Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0.

# Convolutional Neural Network

## Pooling layer

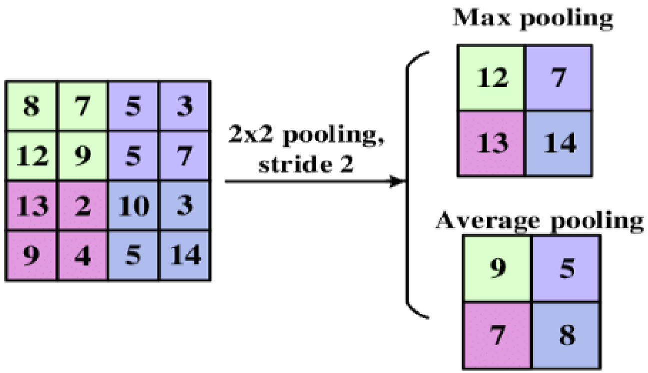



- The primary aim of this layer is to decrease the size of the convolved feature maps to reduce computational costs and complexity.
- This layer helps the network recognize features independent of their location in the image.
- This is performed by decreasing the connections between layers and independently operating on each feature map.

# Convolutional Neural Network

## Pooling layer

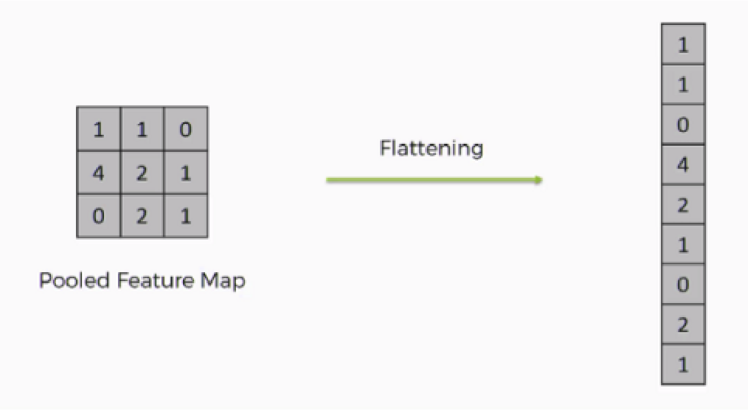
Two methods of pooling: max pooling and average pooling.

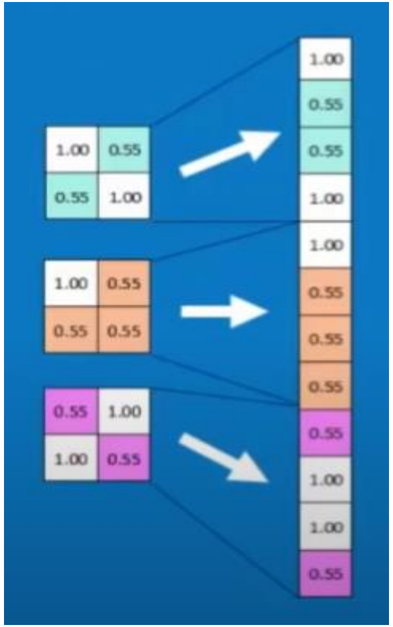


# Convolutional Neural Network

## Flattening

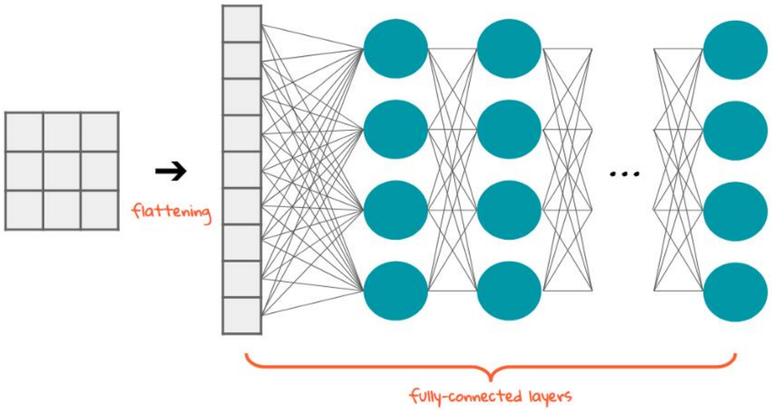
Flattening is used to convert all the 2-dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.



# Convolutional Neural Network

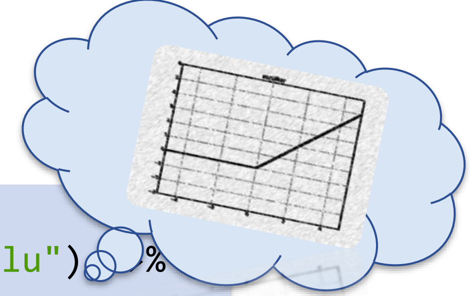
## Fully Connected Layers



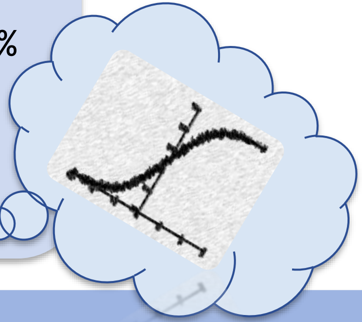

- Fully connected means that all the neurons from one layer are connected to every neuron of the next layer.
- The flattened output is fed to a feed-forward neural network and backpropagation is applied to every iteration of training.
- The Fully Connected layer consists of the weights and biases along with the neurons. These layers form the last few layers of a CNN.
- This layer is where image classification happens based on the features extracted in the previous layers.

# IMPLEMENTATION & APPLICATION

# Model Structure



```r
model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu") %>%
  layer_dropout(0.4) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(0.3) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dropout(0.2) %>%
  layer_dense(units = 2, activation = 'softmax')
```

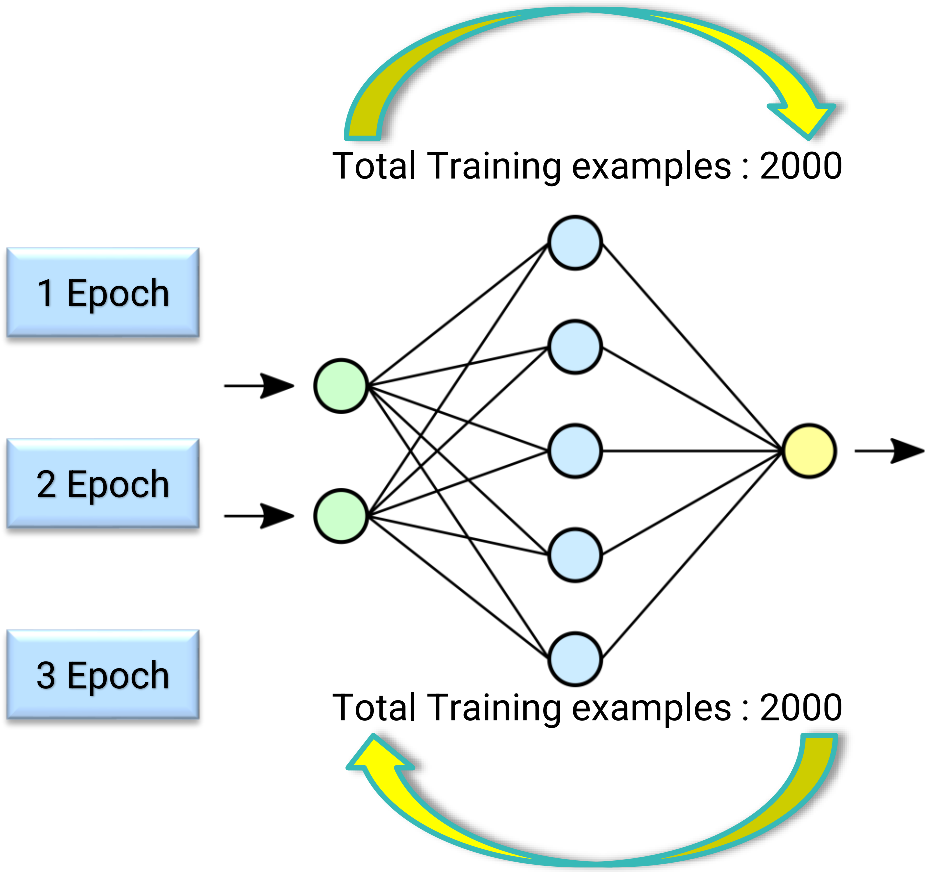


- The function for a CNN in R is called keras_model_sequential()
- **Units** indicates the number of nodes (neurons) in each layer.
- The unit sequence 256 , 128 , 64 … ( n < 64) is the one that is commonly used.
- The last layer corresponds to the output of the CNN and it has 2 units for a binary response.
- **Activation**, is associated with the basic building blocks for fitting distributions to the data.
- **ReLU** (Rectified Linear Unit) is probably the most commonly used Activation Function.
- **Softmax** is a modified version of the ReLU Activation Function.
- **Dropout Layer** is a “filter” that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others.

# Model Fit

```
fit_covid <- model %>%
  fit(x = train,
      y = train_label,
      epochs = 30,
      batch_size = 512, # Can also try 128 and 256
      verbose = 2,
      validation_split = 0.2)
```

# Model Fit | Epoch




- An **Epoch**, is one forward pass and one backward pass of all of the training examples.
- If we use **one** Epoch, the 2000 examples (observations) will **forward and backward through the neural network once**.
- After the first Epoch, the weights will be decent but… 🤔
- **Passing the complete data set** through the neural network (NN) **only once** is **not sufficient**. Therefore the **data shall pass multiple times** in the NN, in order to **prevent underfitting**.
- Providing data repeatedly to the NN, can improve the weights further 🥳

# Model Fit | Batch, Batch Sizes and Iterations

- Since the epoch process might be problematic due to computer system limitations (**feeding all the data at once**), it is preferable to **divide** the data into **smaller batches**.

- The Batch Size is a **Hyperparameter** that **determines** the total **number of training examples** present **in each batch**.

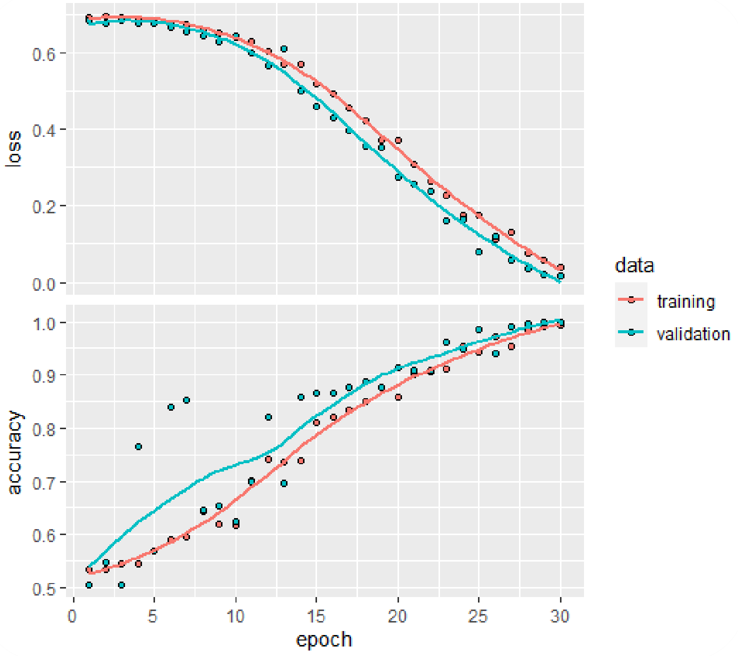
- Larger batch sizes result in faster progress in training but they don’t necessarily converge as fast. Smaller batch sizes train slower but can converge faster.

Iterations is the number of batches needed to perform one epoch.

# RESULTS & INTERPRETATION

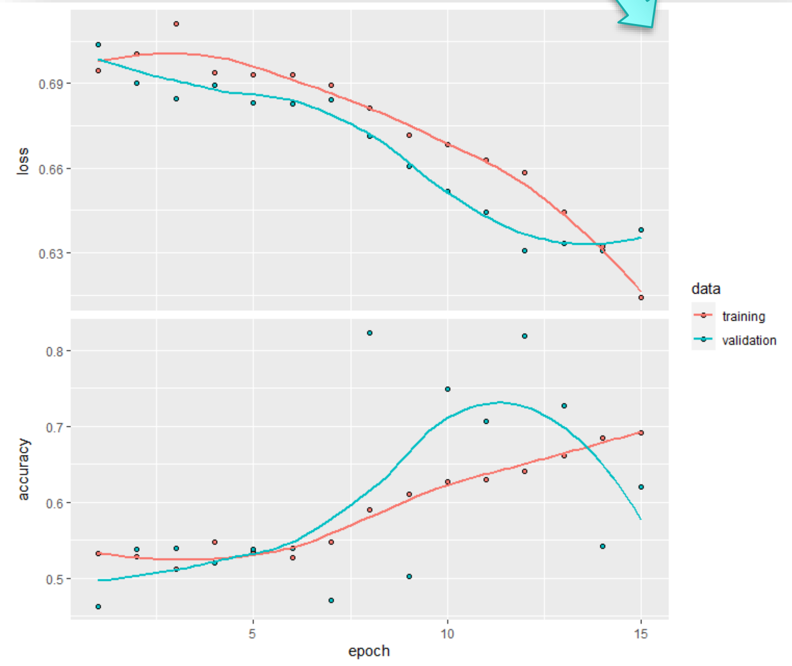
# Accuracy Loss vs Epoch

Early Stop approach by plotting the loss function against the number of Epochs on the training set.

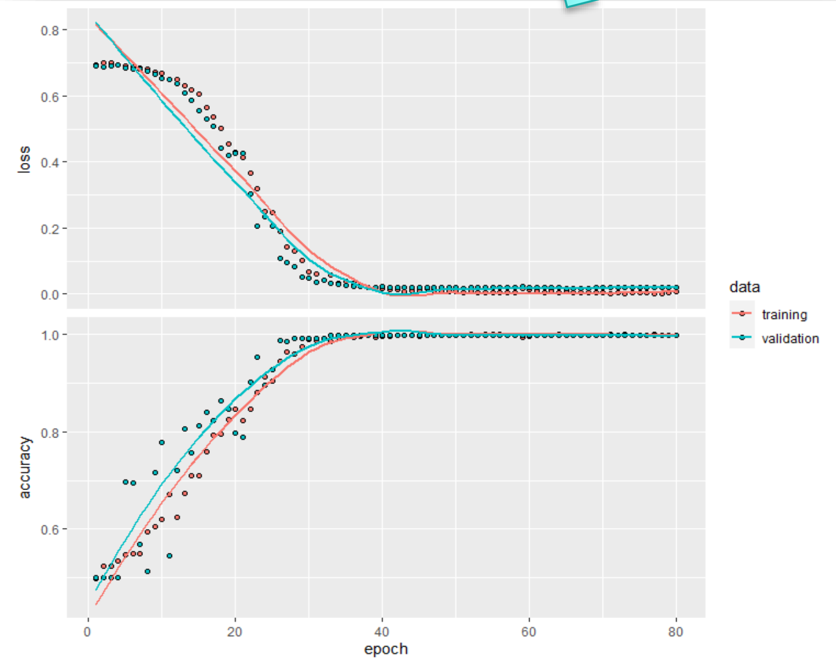



- The training of the model should stop when the loss rate of validation data is minimum.
- At the point that the loss is minimized if we choose to increase the number of epochs the model will be prone to overfitting.
- Generally the models improve with more epochs of training, until a certain point. They will start to plateau (suddenly increase) in accuracy as they converge. The optimal number of epochs is the one that starts to level out.

# Model Fit | Underfitting and Overfitting

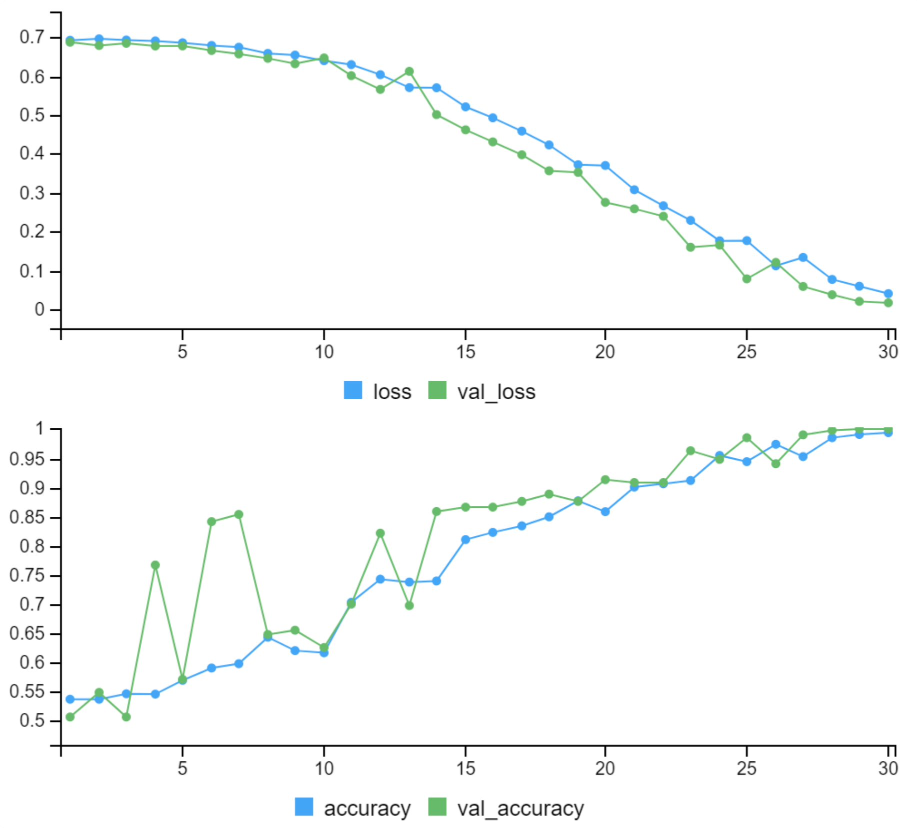



Epochs: 15 Batch Size: 512




Epochs: 80 Batch Size: 512

# Learning History Plot



# Results | Prediction Accuracies

*Test Set (from dataset 1)*

Actual Values

| Predicted Values | 0 | 1 |
|---|---|---|
| 0 | **473** | **2** |
| 1 | **2** | **473** |

**99.6% Accuracy**

*Validation Set (from dataset 2)*

Actual Values

| Predicted Values | 0 | 1 |
|---|---|---|
| 0 | **639** | **107** |
| 1 | **107** | **639** |

**85% Accuracy**

# STRENGTHS, LIMITATIONS & POTENTIAL IMPROVEMENTS

# Strengths


### 01. Unsupervised method

A CNN model does not require any human supervision for the task of identifying important features of every picture.

### 02. High Accuracy With Image Recognition

Very high accuracy in recognizing and classifying CT Scans.

### 03. Minimize Computation

CNNs can minimize computation in comparison with a regular neural network especially in regard with object/ image classification problems.

# Limitations

1

## Small Dataset

A lot of training data is needed for the CNN model to be effective.

2

## Accuracy Of Labels

We can not be sure if Ct scans were labeled correctly from the medical experts

3

## Training process takes a lot of time

The training process could take a particularly long time if the computer does not have a good GPU

4

## Object's Position & Orientation

CNN may fail to encode the position and orientation of objects.

# Solution Proposal

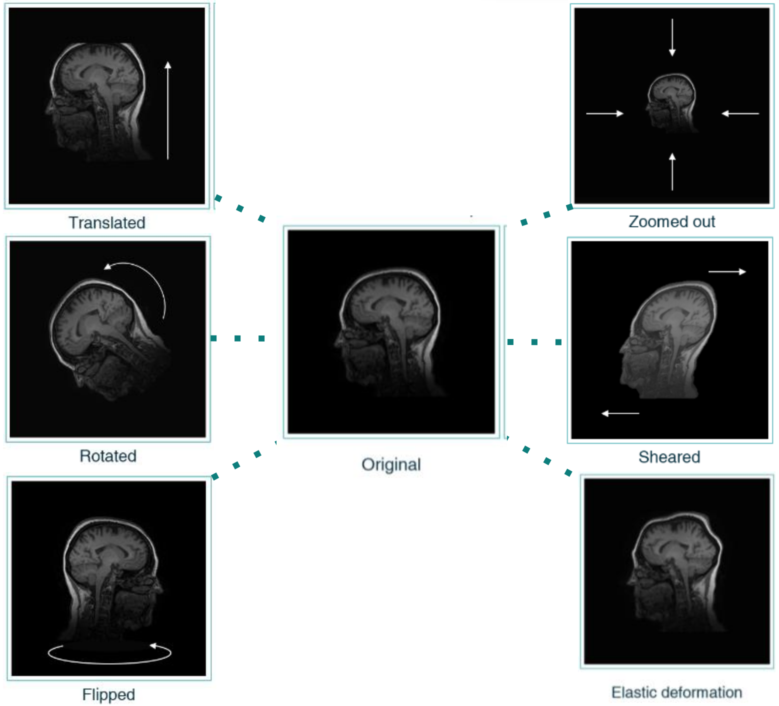
## Image Augmentation

**What Image Augmentation does ?**
Image Augmentation artificially creates training images through different ways of processing the images

### Translating
Shifting the region of interest, with respect to the center of your training image

### Rotating
Rotating the training images by a random amount of degrees

### Flipping
The image information is mirrored horizontally or vertically




### Zooming in/out
Randomly zooming in and zooming out can add more variation to our training data

### Shearing
The image is stretched in two opposite directions at the same time

### Elastic deformation
It is similar to stretching, however, with more freedom.

# Future Research

## Expand Dataset

Collect more images of Ct Scans to train our model incuding Ct Scans of pneumonia, not just covid-19

## Python

Reproduce our code in Python Enviroment in order to apply Image Augmentation

# Related Paper Research

SCAN ME

# THANK YOU FOR YOUR ATTENTION

# Appendix | Loss Function, Optimizer, Metrics

- Compile **defines the loss function, the optimizer and the metrics**. That's all. It has nothing to do with the weights and you can compile a model as many times as you want without causing any problem to pretrained weights. You need a compiled model to train (because training uses the loss function and the optimizer).

- **Adaptive Moment Estimation (Adam)** is an algorithm for optimization technique for gradient descent. The methods is really efficient when working with large problem involving a lot of data or parameters.

- **Adam,** involves **two gradient descent methodologies**. The one is the **Momentum** which is used to **accelerate the gradient descent** algorithm by taking into consideration the "**exponentially weighted average**" of the gradients. The usage of averages makes the algorithm converge towards minima faster. The other is the **Root Mean Square prop (RMSP)** which is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it **takes the exponential moving average**.

- Binary cross entropy is an evaluation metric **compares each of the predicted probabilities to actual class output which can be either 0 or 1**. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value

# Bibliography


- Yang X, He X, Zhao J, et al. COVID-CT-dataset: a CT scan dataset about COVID-19[J]. arXiv preprint arXiv:2003.13865, 2020.
- Soares, Eduardo, Angelov, Plamen, Biaso, Sarah, Higa Froes, Michele, and Kanda Abe, Daniel. "SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification." medRxiv (2020).