MODEL BUILDING FOR FRUIT DISEASE PREDICTION

Convolution Layers:

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function

1.Convolutional Layer:

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).
The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.
The convolution layer in CNN passes the result to the next layer once applying the convolution operation in the input. Convolutional layers in CNN benefit a lot as they ensure the spatial relationship between the pixels is intact

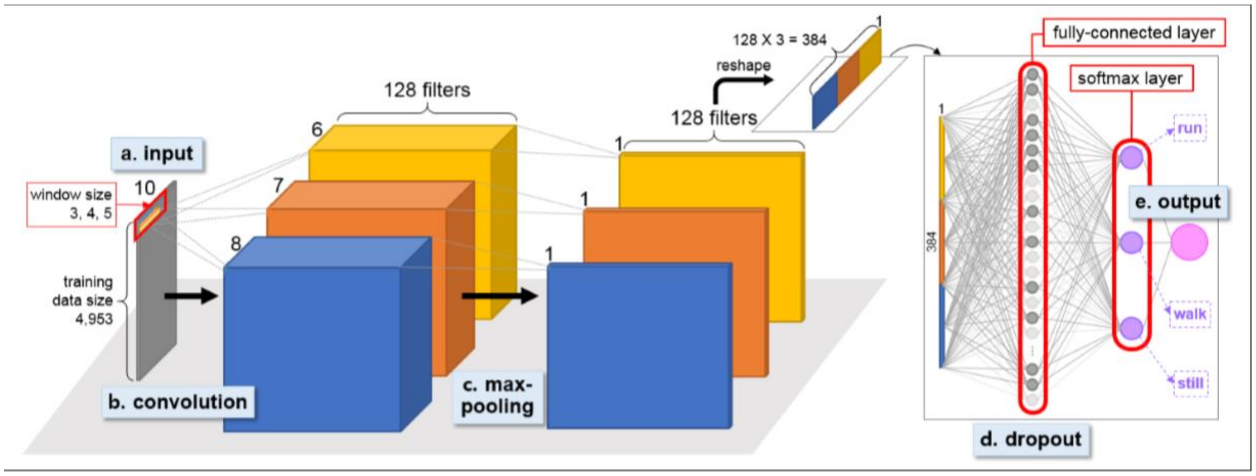


Fig 1.0 Convolution Operation

2. Pooling Layer:

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. It basically summarizes the features generated by a convolution layer.
In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.
This CNN model generalizes the features extracted by the convolution layer, and helps the networks to recognize the features independently. With the help of this, the computations are also reduced in a network.

In Pooling Layer consists two types of pooling Layer:

1. Max Pooling
2. Average Pooling

Max Pooling and Average Pooling:

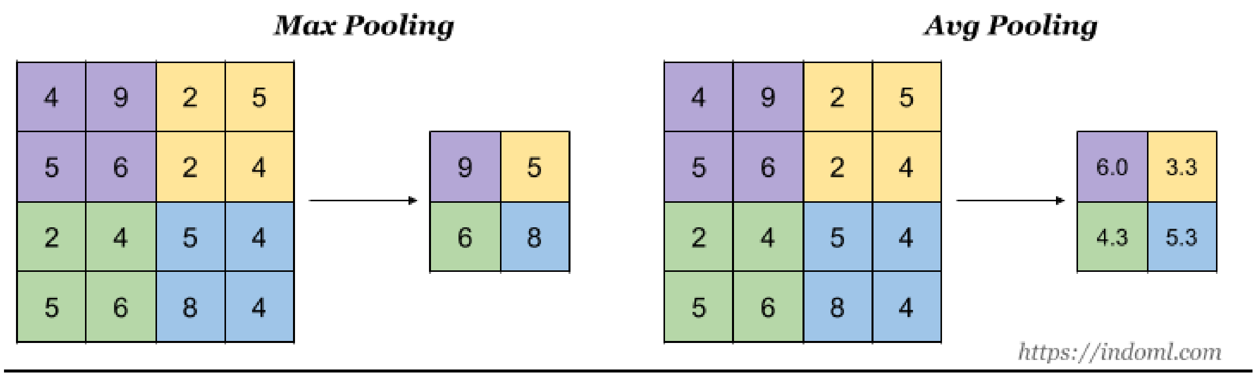


Fig 1.1 Pooling Layers in CNN

3. Fully Connected Layer or Dense Layer:

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place. The reason two layers are connected is that two fully connected layers will perform better than a single connected layer. These layers in CNN reduce the human supervision
Hidden Layer also called as Fully Connected layer,

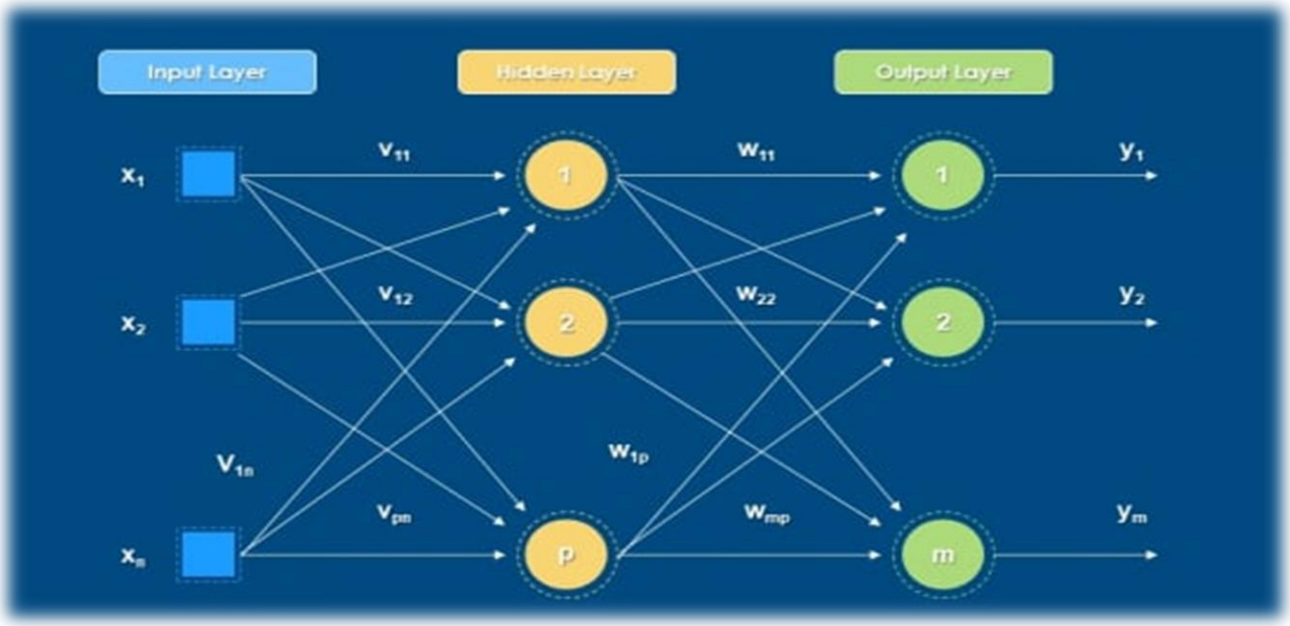


Fig 1.2 Fully Connected layer – Hidden layer

Two Types of Dense Layers are:

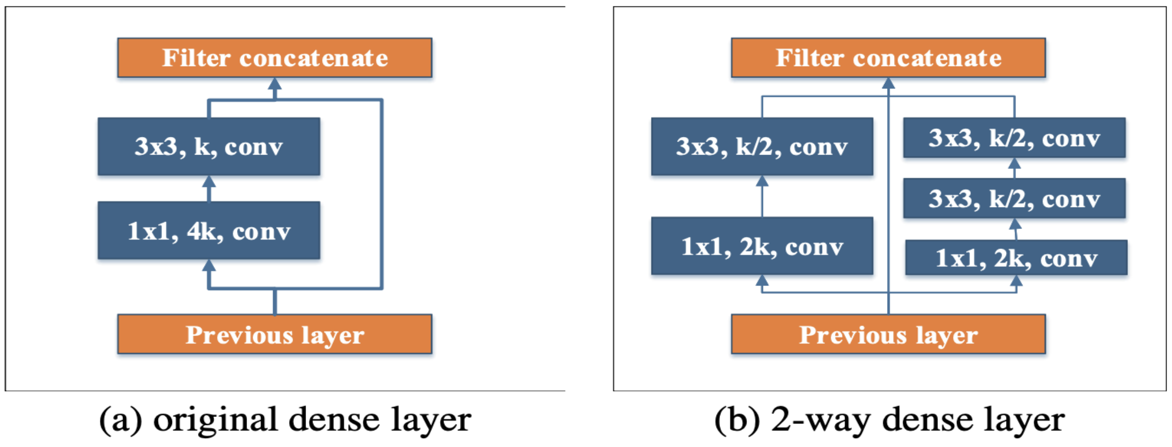


Fig 1.3 Two types of Dense Layer

4. Dropout:

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.
Dropout results in improving the performance of a machine learning model as it prevents overfitting by making the network simpler. It drops neurons from the neural networks during training.

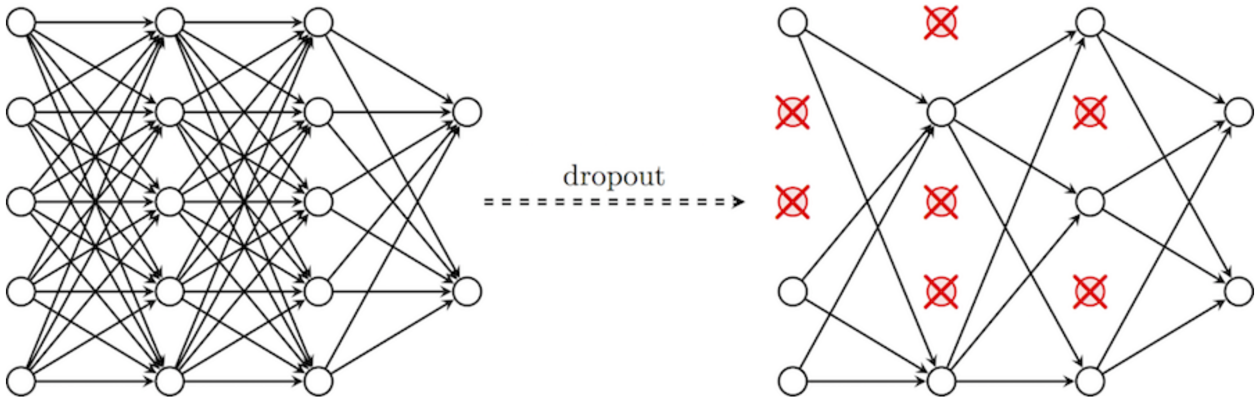


Fig 1.4 Dropout Layer

5.Activation Functions:

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.
It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used. In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not. It decides whether the input to the work is important or not to predict using mathematical operations.

Function of Activation Function:

The input layer neuron node will directly pass the input attribute value to the next layer. layer. In a multilayer neural network, there is a functional relationship between the output of the upper node and the input of the lower node. This function is called the activation function

Activation Functions in Neural Networks

- Linear or Identity Activation Function.
- Non-linear Activation Function.
- Sigmoid or Logistic Activation Function.

- Tanh or hyperbolic tangent Activation Function.
- ReLU (Rectified Linear Unit) Activation Function.
- Leaky ReLU.

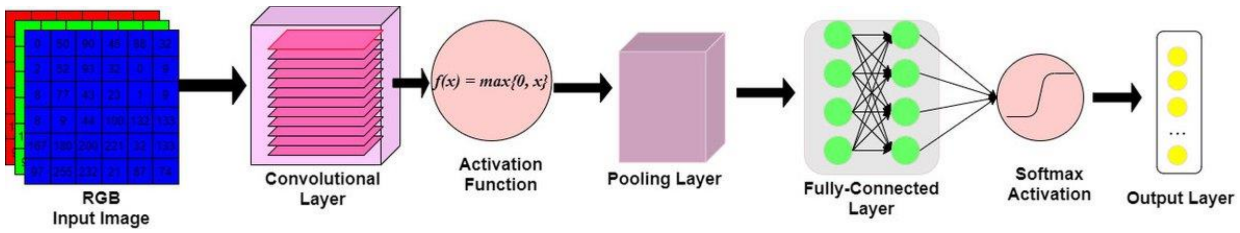


Fig 1.5 Activation Function in CNN

Buliding the CNN model in Image Classification for the Project:

model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(4,4),input_shape=(28, 28, 1), activation='relu',))

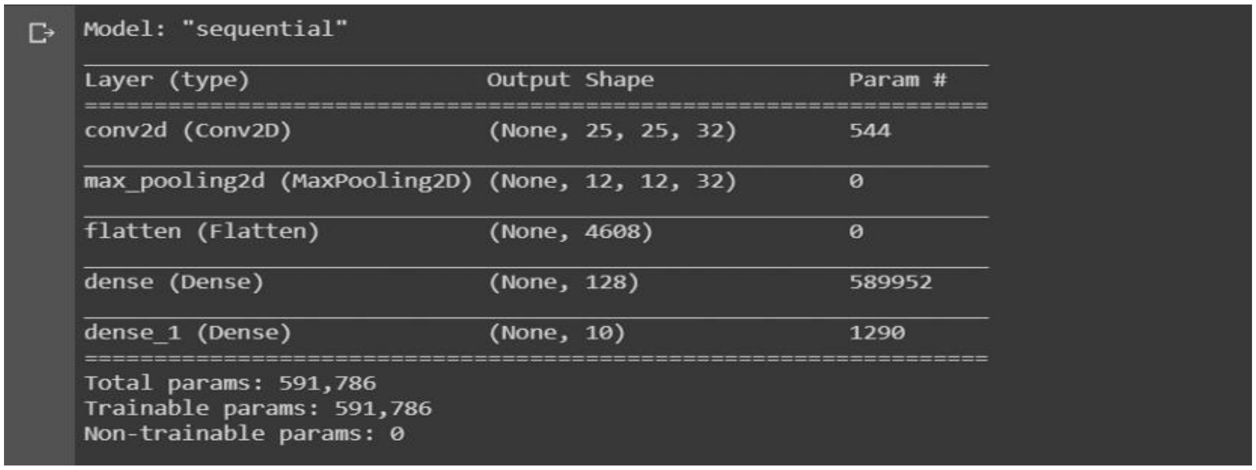
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])



```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 25, 25, 32)        544
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)       0
_________________________________________________________________
flatten (Flatten)           (None, 4608)              0
_________________________________________________________________
dense (Dense)               (None, 128)               589952
_________________________________________________________________
dense_1 (Dense)             (None, 10)                1290
=================================================================
Total params: 591,786
Trainable params: 591,786
Non-trainable params: 0
```