# Deep learning and ML engineering - Face emotion recognition

Technical documentation

Mohammed Arifuddin Atif
arifuddinatif63@gmail.com

## Introduction

The Indian education landscape has been undergoing rapid changes for the past 10 years owing to the advancement of web-based learning services, specifically, eLearning platforms.Digital platforms might overpower physical classrooms in terms of content quality but when it comes to understanding whether students are able to grasp the content in a live class scenario is yet an open-end challenge.

In a physical classroom during a lecture, the teacher can see the faces and assess the emotion of the class and tune their lecture accordingly, whether he is going fast or slow. He can identify students who need special attention. Digital classrooms are conducted via video telephony software program (exZoom) where it’s not possible for medium scale class (25-50) to see all students and access the mood. Because of this drawback, students are not focusing on content due to lack of surveillance. While digital platforms have limitations in terms of physical surveillance but it comes with the power of data and machines which can work for you. It provides data in the form of video, audio, and texts which can be analysed using deep learning algorithms. Deep learning backed system not only solves the surveillance issue, but it also removes the human bias from the system, and all information is no longer in the teacher’s brain rather translated in numbers that can be analysed and tracked.

## Problem statement

We are tasked with solving the challenge of detecting the faces and assess the emotions of the students in a class which would help teachers to tune the lectures accordingly, by applying deep learning algorithms to live video data. The solution to this problem is by recognizing facial emotions.

## Overview of data

The dataset consists of 35887 images with size 48x48 each. There are 7 types of emotions in this dataset which are - angry , sad , happy , fear , disgust , neutral , surprised.

## Steps involved

### I. Installing dependencies

A dependency is a logical, constraint-based or preferential relationship between two activities such that the completion or the initiation of one is reliant on the completion or initiation of the other.The required dependencies for this project are:

1) Python 3
2) Tensor flow
3) Streamlit
4) Streamlit-Webrtc
5) Opencv

### II. Loading the dataset

- We used a function load_data to load the data into our environment.This function reads the pixel intensities of the images and loads the data accordingly.
- The pixel values range from 0 to 255 which represents the intensity or darkness of that particular pixel.
- After loading the data we reshaped into a 48x48 array and normalized the data by dividing all values by 255.
- After this we did one-hot encoding to the images and converted them into arrays.

## III. Training the model

First we split the data into train and test sets, then we used sequential model with different hyperparameters to train the model.The Sequential API is the best method when you are trying to build a simple model with a single input, output, and layer branch. It is an excellent option for newcomers who would like to learn fast.

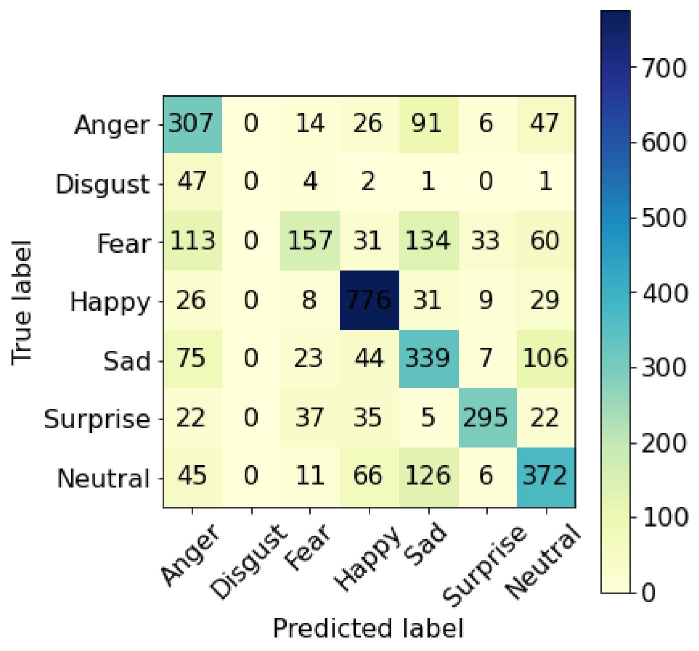
Hyperparameters used in the model:

1. epochs = 75
2. batch_size = 64
3. learning_rate = 0.001

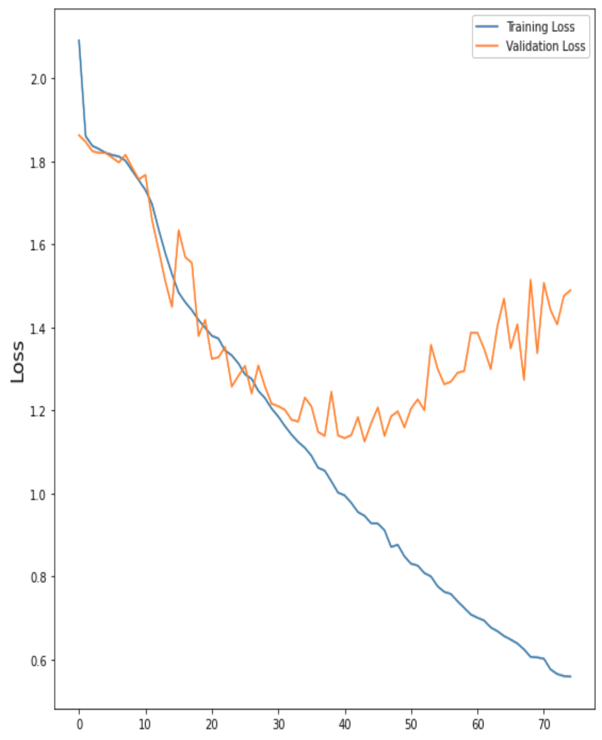
## IV. Evaluating metrics of our model

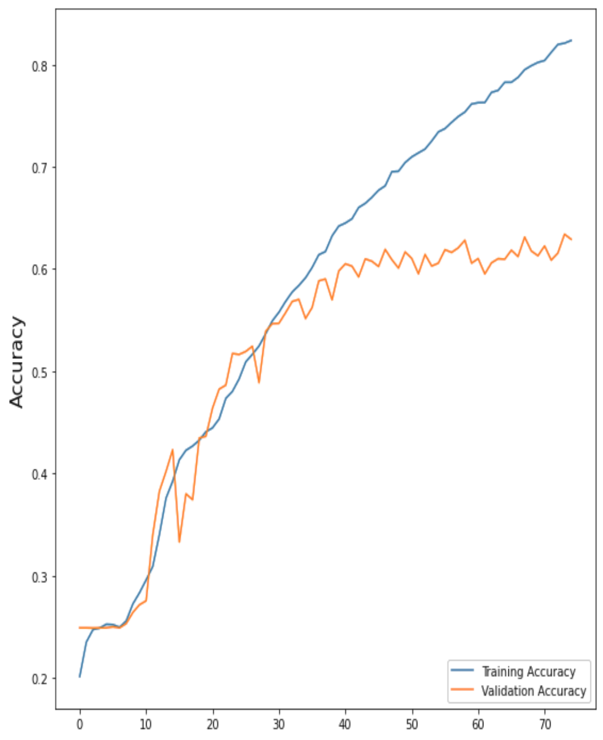
- We were able to achieve 62.5% accuracy for our model.

- We also used confusion matrix to provide a relation between true labels and predicted labels.
- We then generated a classification report showing all the relevant metrics of our model.
- After this we generated a loss and accuracy graph which clearly depicts the loss and accuracy of training and validation sets with each epoch.



Optimizer : Adam



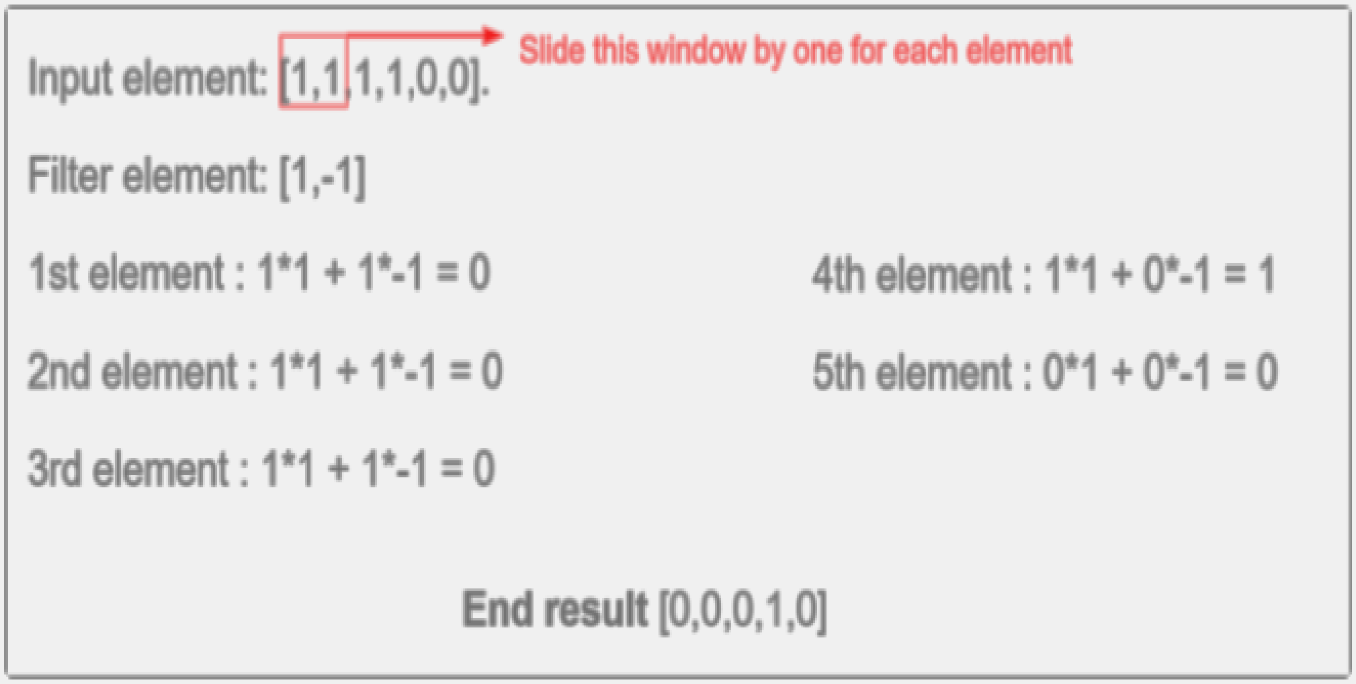


## About FER model

### Convolutional neural network(cnn)

CNN is a type of neural network model which allows us to extract higher representations for the image content. Unlike the classical image recognition where you define the image features yourself, CNN takes the image's raw pixel data, trains the model, then extracts the features automatically for better classification.

A convolution sweeps the window through images then calculates its input and filter dot product pixel values. This allows convolution to emphasize the relevant features.



Input element: [1,1,1,1,0,0].

Slide this window by one for each element

Filter element: [1,-1]

1st element : 1*1 + 1*-1 = 0

2nd element : 1*1 + 1*-1 = 0

3rd element : 1*1 + 1*-1 = 0

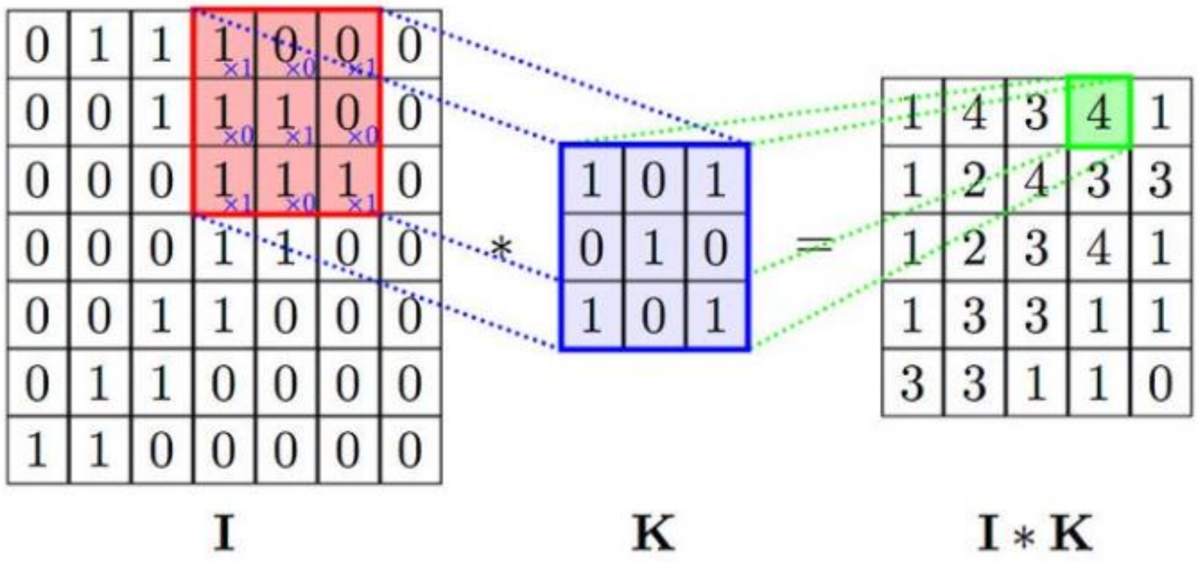
4th element : 1*1 + 0*-1 = 1

5th element : 0*1 + 0*-1 = 0

End result [0,0,0,1,0]

1D Convolution Operation with features(filter)

We will encase the window elements with a small window, dot multiplies it with the filter elements, and save the output. We will repeat each operation to derive 5 output elements as [0,0,0,1,0]. From this output, we can know that the feature change(1 becomes 0) in sequence 4. The filter has done well to identify the input values. Similarly, this happened for 2D Convolutions as well.

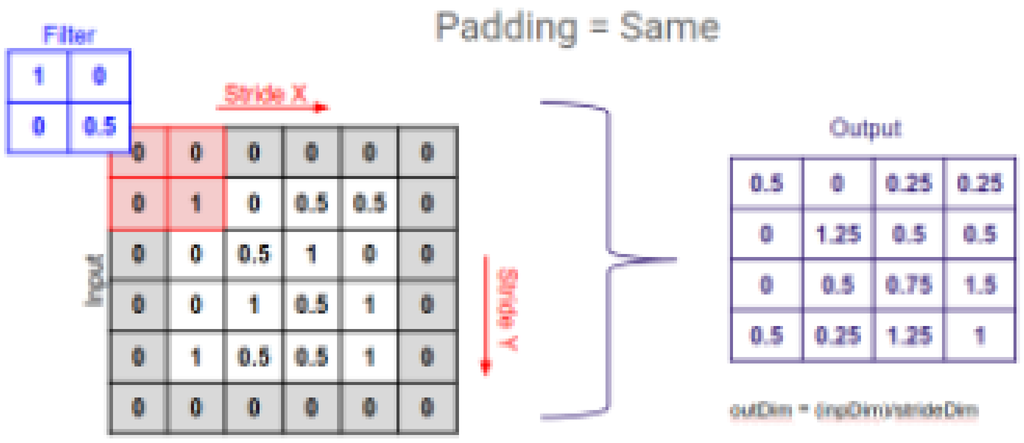


2D Convolution Operation with features(filter) — Source

With this computation, you detect a particular feature from the input image and produce feature maps (convolved features) which emphasizes the important features. These convolved features will always change depending on the filter values affected by the gradient descent to minimize prediction loss.

**Padding :**Padding is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one pixel border added to the image with a pixel value of zero.

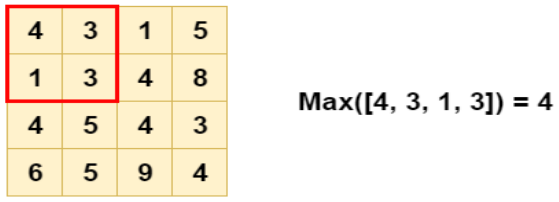
Padding works by extending the area of which a convolutional neural network processes an image. The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for more accurate analysis of images.

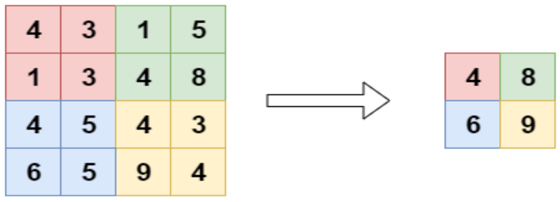


**Pooling :** A pooling layer is another building block of a CNN. Pooling Its function is to progressively reduce the spatial size of the representation to reduce the network complexity and computational cost.There are two types of widely used pooling in CNN layer:

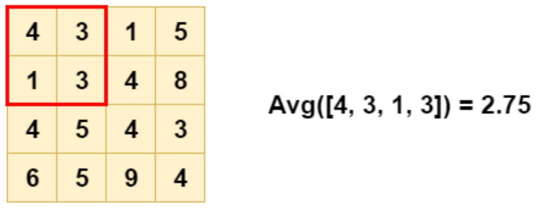
1. Max pooling
2. Average pooling

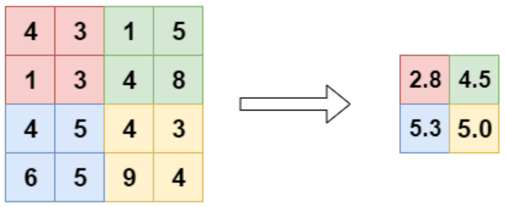
**Max pooling :** Max pooling is simply a rule to take the maximum of a region and it helps to proceed with the most important features from the image. Max pooling selects the brighter pixels from the image. It is useful when the background of the image is dark and we are interested in only the lighter pixels of the image.





**Average pooling :** Average Pooling is different from Max Pooling in the sense that it retains much information about the “less important” elements of a block, or pool. Whereas Max Pooling simply throws them away by picking the maximum value, Average Pooling blends them in. This can be useful in a variety of situations, where such information is useful.





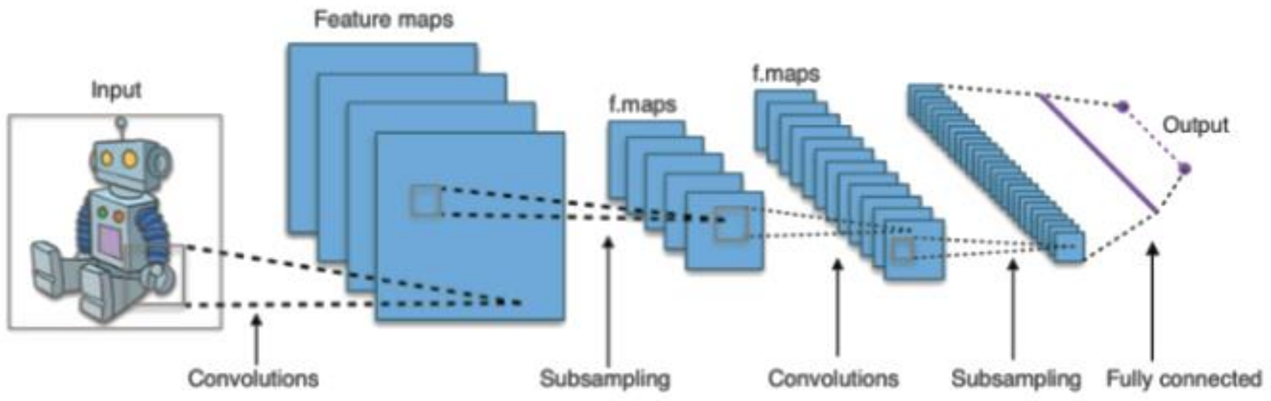
**Activation function :** Activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data.

When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron. That is exactly what an activation function does in an ANN as well. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell.

After each convolutional and max pooling operation, we can apply Rectified Linear Unit (ReLU).It is the most widely used activation function.One of the advantages ReLU has over other activation functions is that it does not activate all the neurons at the same time. The ReLU function mimics our neuron activations on a “big enough stimulus” to introduce nonlinearity for values $x>0$ and returns 0 if it does not meet the condition. This method has been effective to solve diminishing gradients. Weights that are very small will remain as 0 after the ReLU activation function.

Types of activation functions are:

1. Sigmoid
2. Tanh
3. ReLU
4. Leaky ReLU
5. Maxout
6. ELU



CNN architectures with convolutions, pooling (subsampling), and fully connected layers for softmax activation function

## Conclusion

We are finally at the conclusion of our project!

We came all the way from loading the  dataset and applying necessary changes to the data like reshaping and normalizing. We then trained our model and evaluated the metrics of our model.

We were able to achieve a validation accuracy of 62.43% and with test data an accuracy of 63.17%.