**IMPLEMENTATION**

**MODULES:**

* Dataset
* Importing the necessary libraries
* Retrieving the images
* Splitting the dataset
* Building the model
* Apply the model and plot the graphs for accuracy and loss
* Accuracy on test set
* Saving the Trained Model
* **Face expression in Live webcam Feed**

**MODULES DESCSRIPTION:**

**Dataset:**

In the first module, we developed the system to get the input dataset for the training and testing purpose. We given the data set in model folder

The dataset Link: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data.

**Importing the necessary libraries:**

We will be using Python language for this. First we will import the necessary libraries such as keras for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy, matplotlib and tensorflow.

**Retrieving the images:**

We will retrieve the images and their labels. Then resize the images to (224,224) as all images should have same size for recognition. Then convert the images into numpy array.

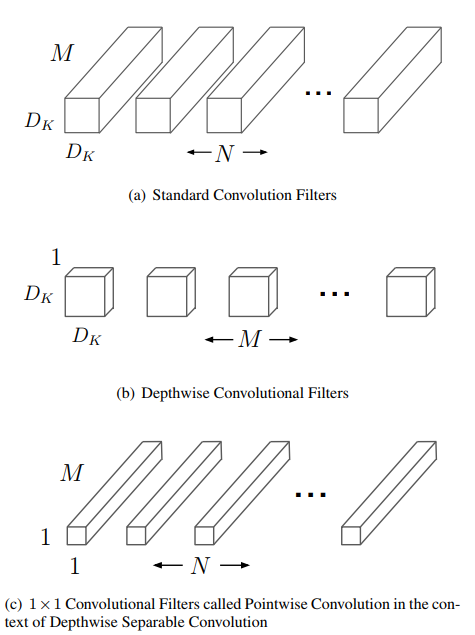
**Splitting the dataset:**

Split the dataset into train and test. 80% train data and 20% test data.

**MobileNet | CNN model**

**Architecture:**

We shall be using Mobilenet as it is lightweight in its architecture. It uses depthwise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it. This has the effect of filtering the input channels. Or as the authors of the paper explain clearly: “ *For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size*. ”



Difference between pointwise and depth wise convolutions

So the overall architecture of the Mobilenet is as follows, having 30 layers with

convolutional layer with stride 2

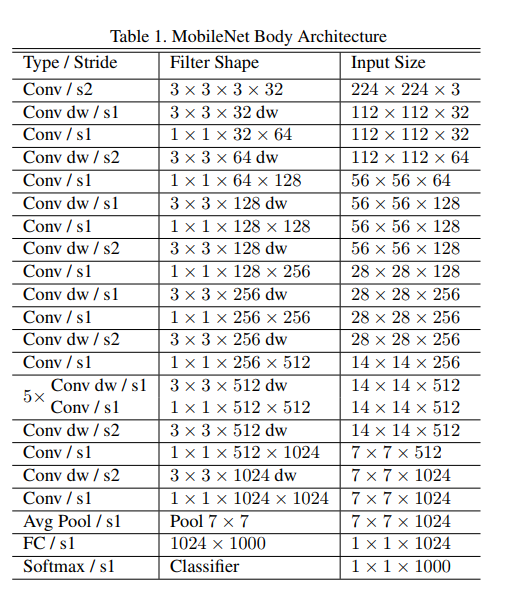
depthwise layer

pointwise layer that doubles the number of channels

depthwise layer with stride 2

pointwise layer that doubles the number of channels

etc.



Mobilenet full architecture

**Building the model:**

The concept of convolutional neural networks. They are very successful in image recognition. The key part to understand, which distinguishes CNN from traditional neural networks, is the **convolution** operation. Having an image at the input, CNN scans it many times to look for certain **features**. This scanning (convolution) can be set with 2 main parameters: stride and padding type. As we see on below picture, process of the first convolution gives us a set of new frames, shown here in the second column (layer). Each frame contains an information about one feature and its presence in scanned image. Resulting frame will have larger values in places where a feature is strongly visible and lower values where there are no or little such features. Afterwards, the process is repeated for each of obtained frames for a chosen number of times. In this project I chose a classic MobileNet model which contains only two convolution layers.

The latter layer we are convolving, the more high-level features are being searched. It works similarly to human perception. To give an example, below is a very descriptive picture with features which are searched on different CNN layers. As you can see, the application of this model is face recognition. You may ask how the model knows which features to seek. If you construct the CNN from the beginning, searched features are random. Then, during training process, weights between neurons are being adjusted and slowly CNN starts to find such features which enable to meet predefined goal, i.e. to recognize successfully images from the training set.

Between described layers there are also **pooling** (sub-sampling) operations which reduce dimensions of resulted frames. Furthermore, after each convolution we apply a non-linear function (called **ReLU**) to the resulted frame to introduce non-linearity to the model.

Eventually, there are also fully connected layers at the end of the network. The last set of frames obtained from convolution operations is flattened to get a one-dimensional vector of neurons. From this point we put a standard, fully-connected neural network. At the very end, for classification problems, there is a softmax layer. It transforms results of the model to probabilities of a correct guess of each class

**Apply the model and plot the graphs for accuracy and loss:**

We will compile the model and apply it using fit function. The batch size will be 32. Then we will plot the graphs for accuracy and loss. We got average validation accuracy of 91.00% and average training accuracy of 90.00%.

**Accuracy on test set:**

We got an accuracy of 90.00% on test set.

**Saving the Trained Model:**

Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or . pkl file using a library like pickle .

Make sure you have pickle installed in your environment.

Next, let’s import the module and dump the model into .h5 file

**Face expression in Live webcam:**

Here, we capture the video. The read() function reads one frame from the video source, which in this example is the webcam. This returns:

The actual video frame read (one frame on each loop)

A return code

The return code tells us if we have run out of frames, which will happen if we are reading from a file. This doesn’t matter when reading from the webcam, since we can record forever, so we will ignore it.

Again, this code should be familiar. We are merely searching for the face in our captured frame. The results will be angry, disgust, fear, happy, neutral, sad, surprise