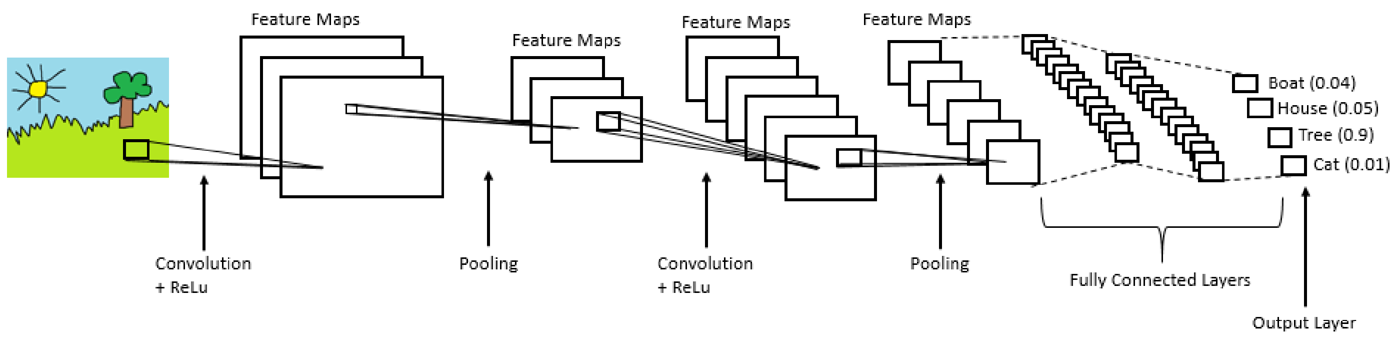
AI-AUG-Major Project

By AI08B5

a) Which Neural Network and why?

Convolutional neural network (CCN) is used because it uses relatively little pre-processing compared to other image classification algorithms. CNN effectively uses adjacent pixel information to effectively down sample the image first by convolution and then uses a prediction layer at the end. It automatically detects the important features without any human supervision. Its learning process is fast and has less error rate comparatively. Before CNN, we need to spend so much time on feature selection which in comparison to CNN were not that accurate nor fast.



b) Which optimizer and why?

Adam optimization is used as it is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. This algorithms leverages the power of adaptive learning rates methods to find individual learning rates for each parameter. It also has advantages of Adagrad, which works really well in settings with sparse gradients, but struggles in non-convex optimization of neural networks. Optimizers such as Nadam and simple SGD were considered but SGD proved to be slower in this case as it doesn't have that momentum in order to converge.

c) Which accuracy metric and why?

The accuracy metric we used for our model was **Accuracy class**. It is simply counting the number of times we predicted the right class over the total number of predictions. We chose it since it is the most straightforward and intuitive metric for classifier performance. Also, it is easy to calculate, easy to interpret, and generates a single number to summarize the model's capability.

d) Which loss function and why?

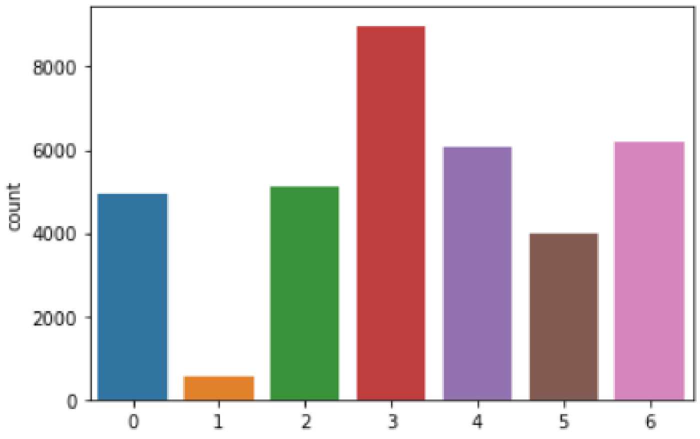
The loss function we used in our model is **CategoricalCrossentropy class**. It is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one. Our model uses the categorical crossentropy to learn to give a high probability to the correct digit and a low probability to the other digits. Our model uses a single Categorical feature as target. This will automatically create a one-hot vector from all the categories identified in the dataset. Each one-hot vector can be thought of as a probability distribution, which is why by learning to predict it, the model will output a *probability* that an example belongs to any of the categories.

e) Brief information on how cleaning was done?

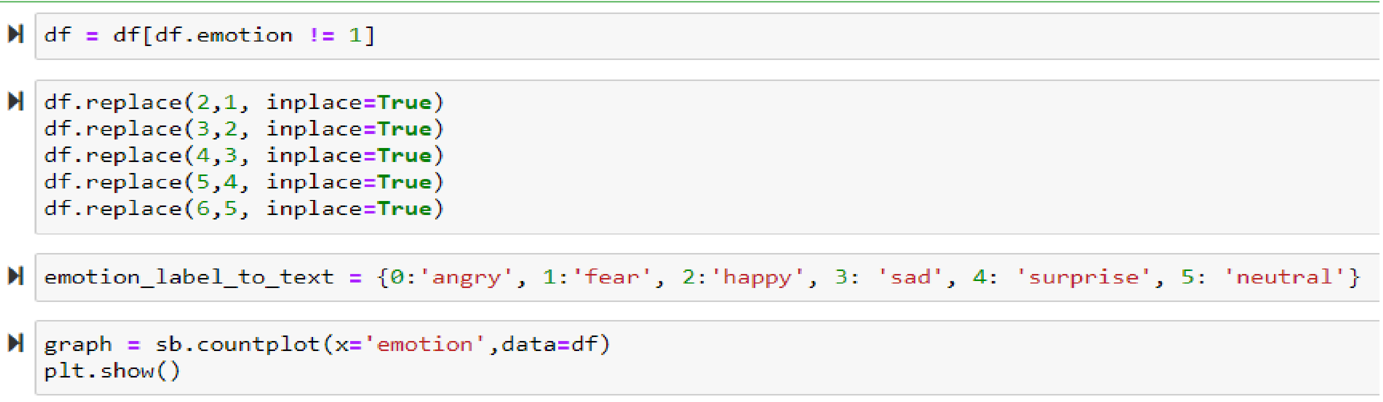
Firstly the dataset was converted into a dataframe using pandas. Then a countplot was plotted to see distribution of data using

“graph = sb.countplot(x='emotion',data=df)

plt.show()”



It was observed that emotion 1 which was disgust has significantly less examples which disturbs the distribution of the dataset so it was decided to drop that emotion.

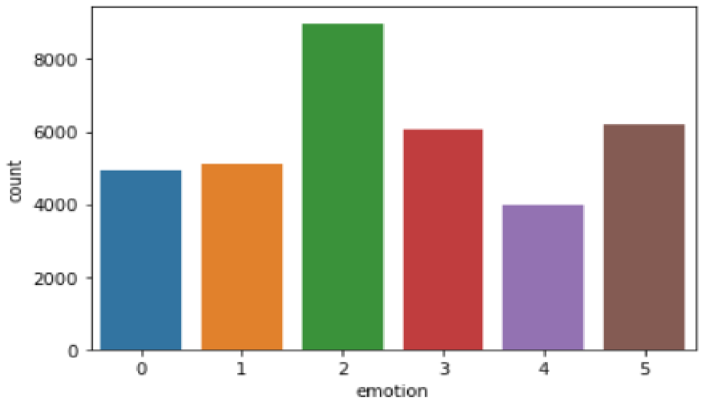


```
df = df[df.emotion != 1]
```

```
df.replace(2,1, inplace=True)
df.replace(3,2, inplace=True)
df.replace(4,3, inplace=True)
df.replace(5,4, inplace=True)
df.replace(6,5, inplace=True)
```

```
emotion_label_to_text = {0:'angry', 1:'fear', 2:'happy', 3: 'sad', 4: 'surprise', 5: 'neutral'}
```

```
graph = sb.countplot(x='emotion',data=df)
plt.show()
```



Further examining of dataset for any discrepancy or null values was done using

df.describe()

df.head()

df.info()

print(df.isnull().sum())

f) How data was got into the right shape (if any)

Most convolutional neural networks are designed in a way so that they can only accept images of a fixed size. This creates several challenges during data acquisition and model deployment. The common practice to overcome this limitation is to reshape the input images so that they can be fed into the networks. The Convolution2D layers in Keras are designed to work with 3 dimensions. For example they have 4-dimensional inputs and outputs. This covers colour images (nb_samples, nb_channels, width, height), but more importantly, it covers deeper layers of the network, where each example has become a set of feature maps i.e. (nb_samples, nb_features, width, height).

We have reshaped our dataset inputs (X_train and X_test) to the shape that our model expects when we train the model. The first number is the number of images (28709 for X_train and 3589 for X_test). Then comes the shape of each image (48x48). The last number is 1, which signifies that the images are greyscale.

X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)

X_test = X_test.reshape(X_test.shape[0], 48, 48, 1)

Train and test images (48 x 48).We reshape all data to 48x48x1 3D matrices. Keras needs an extra dimension in the end which correspond to channels. Our images are grey scaled so it use only one channel.

g) What functions/features of OpenCV were used?

1) **cv2.CascadeClassifier()**
   Cascade classifier class for object detection.
2) **cv2.VideoCapture(0)**
   to get a **video capture** object for the camera. Set up an infinite while loop and use the read() method to read the frames using the above created object.
3) **cv2.cvtColor()**
   Method is used to convert an image from one color space to another.

4) **cv2.detectMultiScale()**
   Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles.
5) **cv2.rectangle()**
   Method is used to draw a **rectangle** on any image.
6) **cv2.resize()**
   To **resize** an image in Python, by default, does only change the width and height of the image.
7) **cv2.putText()**
   Method is used to draw a text string on any image.
8) **cv2.imshow()**
   To show the frames in the video. Breaks the loop when the user clicks a specific key
9) **cv2.waitKey()**
   It is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event.

10) **cv2.destroyAllWindows()**
Destroys all the windows we created.

h) Which dataset have you used?

Dataset used: fer2013.csv

Available at: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

The training set consists of 28,709 examples. The public test set used for the leader board consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.