Neural Networks and Fuzzy Logic

Assignment – 1

Submitted to: Dr. Tanmay Tulsidas Verlekar
Submitted by:

- Sarthak Dalmia (2018A7PS0290G)
- Atharva Trivedi (2018A7PS0228G)
- Aman Rakesh Patel (2018A7PS0200G)
- Rishit Mayur Patel (2018A7PS0189G)

Date of Submission: 26/02/2021

Introduction

The main objective of this project to get ourselves familiarized with the internal workings of CNN and get some hands-on experience of it. For this we will be developing CNN for classifying human facial expressions into 7 categories.

The motivation behind this assignment is to get a good knowledge in the internal workings of convolution neural network and understanding how different parameters affect the overall accuracy and fitness of a model.

Data Pre-processing

The dataset given had two columns with the first column being 'emotion' and second column containing the image as space separated integer values of pixels under the column name 'pixels'. First, we had to convert this data into matrix form of 48 by 48 pixels of 1 dimension. After this we rescaled the image pixel value to between 0 and 1 to avoid exploding problem.

Method

We have designed a convolutional neural network with 7 convolutional layers and 2 dense layers. Different number of filters has been used in the model with varying kernel size in each layer. The activation function used in the final fully connected layer is 'softmax'. The optimizer used for the model is Adam and the learning rate used is 0.001 (or 1e-3). The loss function chosen is 'categorical_crossentropy'. The batch size used is 64 and the number of epochs is 15. The data was split between validation and train data in the ration of 1:4.

Results and Analysis

Initially we trained a simple LeNet architecture with the parameters as shown below and achieved initial accuracy was 56.75%. After changing the model by adding a few more convolutional layers and such as Batch Normalization to increase the performance and dropout to mitigate the overfitting of the model, the train accuracy increased to 75.77% and validation accuracy became 65.726% for dropout of 0.3(throughout) and validation split of 0.2. The validation recall and validation precision achieved was 60.28% and 70.49% respectively.

We have reported below a few more results of models with different dropouts and validation splits:

- Validation Split = 0.2

Dropout	Accuracy	Val Accuracy
0.2	71.76%	62.90%
0.3	62.71%	60.55%
0.4	58.15%	58.43%

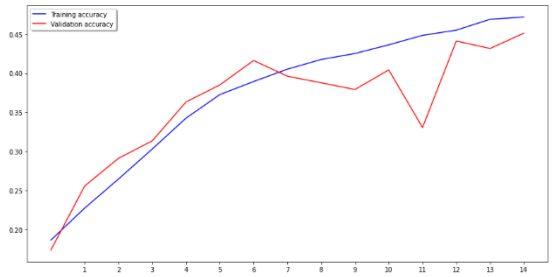
- Dropout = 0.2

Split	Accuracy	Val Accuracy
0.2	71.13%	62.84%
0.3	83.84%	66.74%
0.4	90.11%	69.92%

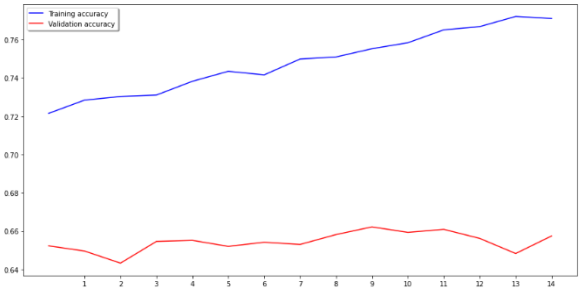
- Dropout = 0.3

Split	Accuracy	Val Accuracy
0.2	75.58%	65.26%
0.3	71.09%	64.71%
0.4	58.70%	59.87%

- Dropout =0.3, Split =0.2 and optimizer = SGD



- Dropout =0.3, Split =0.2 and optimizer = Adam(lr=1e-3)



As we can see with lower dropout values the model overfits on the training data hence giving low performance on the validation data. Increasing the dropout

helps in mitigating the overfitting to some extent. However, larger dropout values harm the overall performance of the network since the network becomes less complex.
On decreasing the train test split ratio we can again see the model overfitting since the size of the training data decreases.

Conclusion

CNNs are widely used for purposes of image classification and are used in some of the present state-of-the-art approaches. We demonstrated the use of CNNs in classifying human facial expression with a validation accuracy of 65.26% and a train accuracy of 75.58%. This is significantly different from performance that can be achieved from random selection (1/7) which demonstrates that CNNs have a great use in the upcoming and foreseeable future. We also demonstrated how a model can overfit the training data and how this can be countered using techniques like dropout and how test train split ratio can affect the performance.

Model: "sequential_9"

Layer (type)	Output Shape	Param #
conv2d_63 (Conv2D)	(None, 48, 48, 32)	160
batch_normalization_72 (Batc	(None, 48, 48, 32)	128
conv2d_64 (Conv2D)	(None, 48, 48, 32)	4128
batch_normalization_73 (Batc	(None, 48, 48, 32)	128
max_pooling2d_45 (MaxPooling	(None, 24, 24, 32)	0
dropout_54 (Dropout)	(None, 24, 24, 32)	0
conv2d_65 (Conv2D)	(None, 24, 24, 64)	18496
batch_normalization_74 (Batc	(None, 24, 24, 64)	256
conv2d_66 (Conv2D)	(None, 24, 24, 64)	36928
batch_normalization_75 (Batc	(None, 24, 24, 64)	256
max_pooling2d_46 (MaxPooling	(None, 12, 12, 64)	0
dropout_55 (Dropout)	(None, 12, 12, 64)	0
conv2d_67 (Conv2D)	(None, 12, 12, 128)	73856
batch_normalization_76 (Batc	(None, 12, 12, 128)	512
max_pooling2d_47 (MaxPooling	(None, 6, 6, 128)	0
dropout_56 (Dropout)	(None, 6, 6, 128)	0
conv2d_68 (Conv2D)	(None, 6, 6, 256)	295168
batch_normalization_77 (Batc	(None, 6, 6, 256)	1024
max_pooling2d_48 (MaxPooling	(None, 3, 3, 256)	0
dropout_57 (Dropout)	(None, 3, 3, 256)	0
conv2d_69 (Conv2D)	(None, 3, 3, 256)	262400
batch_normalization_78 (Batc	(None, 3, 3, 256)	1024
max_pooling2d_49 (MaxPooling	(None, 1, 1, 256)	0
dropout_58 (Dropout)	(None, 1, 1, 256)	0
flatten_9 (Flatten)	(None, 256)	0
dense_18 (Dense)	(None, 512)	131584
batch_normalization_79 (Batc	(None, 512)	2048
dropout_59 (Dropout)	(None, 512)	0
dense_19 (Dense)	(None, 7)	3591

Total params: 831,687
Trainable params: 828,999
Non-trainable params: 2,688