ENEE633 – Statistical Pattern Recognition | Project – 2

Due: December 14th, 2018 11:59 PM (EST)

Shubham Deshkar | UID: 115484689

INTRODUCTION

This project serves as a direct succession to the previous project in which traditional machine learning techniques and algorithms were evaluated based on their performance and computational ease.

The next obvious step would be to evaluate newer techniques like Deep Learning and Transfer Learning and expand these on larger and more complex problems such as handwritten digits image recognition (MNIST dataset) and monkey species identification in color.

1. DEEP LEARNING USING CONVOLUTIONAL NEURAL NETWORKS

a. The Architecture:

In 1998, known researcher Yann LeCun published the importance of gradient based learning [1] using the convolutional neural network. This was used to solve the problem of recognizing images of handwritten digits which was used for different purposes. This breakthrough network was dubbed as Lenet-5. Hence, it was fitting to follow the similar architecture for the purposes of this project as well.

The network comprised of 7 layers. 3 convolutional layers, 2 sub-sampling layers and 2 fully connected layers. The summary is given as follows:

Name	Layer	Filters	Output Shape	Parameters
C1	2D convolution	6 – 5x5	[28, 28, 6]	156
S2	Average pooling	2x2	[14, 14, 6]	0
C3	2D convolution	16 – 5x5	[10, 10, 16]	2416
S4	Average pooling	2x2	[5, 5, 16]	0
C5	2D convolution	120 – 5x5	[1, 1, 120]	48120
Flatten	Stack to Line	-	[120]	0
F6	Fully Connected	-	[84]	10164
F7	Fully Connected	-	[10]	850

Total parameters = 61,706

b. Pre-processing of the images:

It should be noted that the original data was 28x28 pixels grayscale images which were pre-processed carefully to make feature selection better with the network and to make the computations easy and fast.

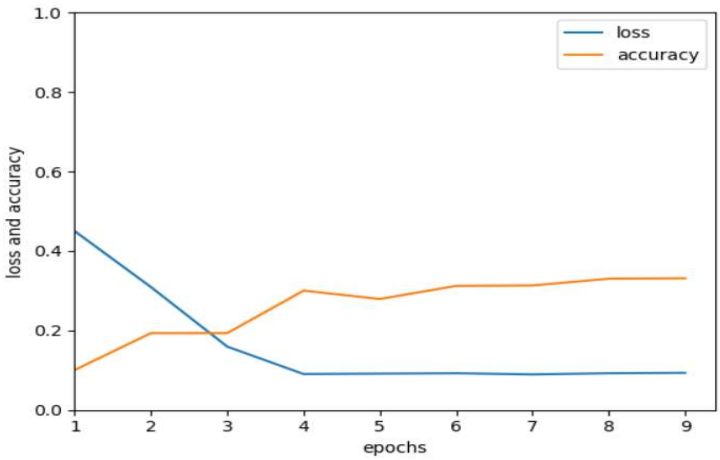
All the images were bordered and padded with zeros on all four sides to increase the height and width to 32 pixels. This essentially increased the dimensionality of the problem, but the artificial neurons in the last fully connected layers corresponds to a receptive field of 20x20, which is the actual size of the written digit in the original image of 28x28 pixels. This also enables filters to capture the edges and endpoints of the strokes.

Additionally, the pixel value was scaled between -0.1 corresponding to 0 and 1.175 corresponding to 255 which essentially centers the data (mean = 0) and variance is unity.

c. Results:

The learning method adopted here was Stochastic Gradient Descent (SGD, learning rate 0.1), the same as the one described in the paper [1] with the loss function as Mean Squared Error (MSE). The batch size was selected as a standard number of 64. Note that the activation function used in all the layers was Sigmoid Function. The results are as follows:

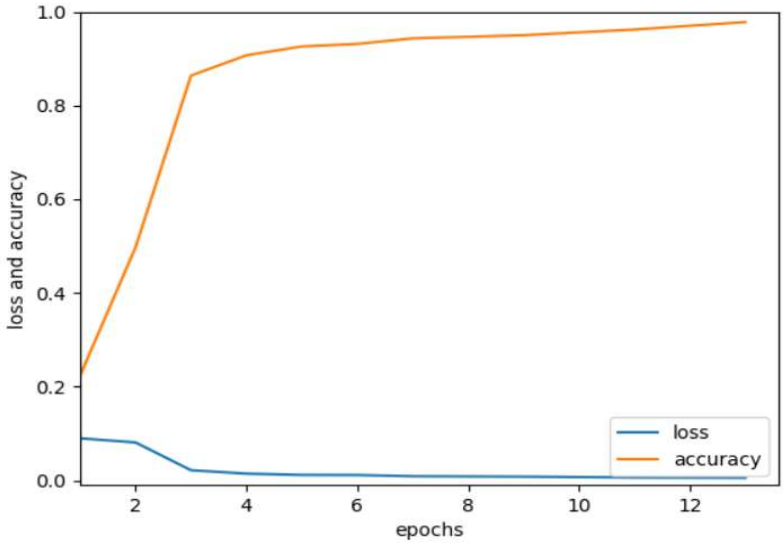
Epochs	Loss	Accuracy
1	0.450	0.101
2	0.309	0.193
3	0.159	0.193
4	0.090	0.300
5	0.091	0.279
6	0.092	0.312
7	0.089	0.313
8	0.092	0.330
9	0.093	0.331



Through the results obtained here, it is clear that sigmoid activation function was not a good fit as the problem of *vanishing gradient* caused the accuracy to remain more or less unchanged.

Later, the activation function was changed to *Rectified Linear Unit* (ReLu) and the results are as follows:

Epochs	Loss	Accuracy
1	0.0897	0.2235
2	0.0809	0.4973
3	0.0216	0.8638
4	0.0143	0.9073
5	0.0115	0.9260
6	0.0113	0.9315
7	0.0088	0.9433
9	0.0079	0.9500
11	0.0060	0.9622
13	0.0055	0.9781



The results were drastically improved, and the accuracy went up with the number of epochs and subsequently the error went down with it and hence the network was named *MyNet*.

2. MULTICLASS SUPPORT VECTOR MACHINES ON MNIST DATASET

Support vector machines, one of the very powerful off the shelf machine learning algorithm was used to evaluate its performance on multiclass dataset against its renowned binary classification. The MNIST dataset used, lies in $\mathbf{R}^{784}$ which is quite high and hence it can be safely assumed that the data is linearly inseparable.

The code for SVM classification was already developed in the previous project and hence was now expanded to use on multiclass 1 vs. 1 class classification. The method was evaluated using *Principal Component Analysis* (PCA) transformed data and *Kernel tricks*. The number of components were selected such that it captured ~97.3% variance in the data.

	Linear	RBF	Poly (d=2)	Poly (g=1)	Poly (d=3)
Raw data	81.23%	92.03%	86.12%	74.33%	83.92%
With PCA	82.17%	91.78%	87.35%	75.60%	87.87%

Although it required a lot of computation time for training and testing, the results were fairly acceptable. Different values of degree and gamma factor in polynomial kernel were used to evaluate the best fit and polynomial kernel of *degree 2* performed relatively better than others although not by a good margin. It can be concluded that *RBF kernel* was the best fit for this problem as it provided the maximum separation and hence the classifiers obtained after computation did good job of classifying images in the test set.

Also, after PCA transformation the computation was also reduced significantly in the range of 100-200 seconds against ~20 mins. Hence, PCA proved to be very helpful in increasing the accuracy by a fair amount and decreasing the computational effort.

3. TRANSFER LEARNING ON MONKEY DATASET

a. Size of the problem and preprocessing of images:

The dataset provided in this problem was of 10 classes of species of monkeys. All the images were colored in 3 channels (*RGB*) and the sizes of images were highly inconsistent. This posed problems while feeding data to the network and computational efforts as networks only accepted images of fixed size throughout training and validation phases. Moreover, the size of the images was large and colored as compared to simple MNIST data.

b. Resize or Reshape (Rescale):

To make the dataset uniform and consistent with respect to its dimensions, the shape of each individual image needs to be similar. Hence, there are two possibilities of either cropping/padding or rescaling the entire image to squish/stretch to the given size.

Since it was not a good idea to rescale the image as it causes the features to be distorted, resizing was chosen to go ahead with [2].

	Average row size	**Average column size**
Training set	783	954
Testing set	812	985

Hence from the given data, an approximate mean of 900 was chosen and images were either padded or cropped to fit in this dimension. Even so, the dimensionality of each image is now 900x900x3 which is huge as compared to MNIST dataset.

c. Initial attempts at training:

Since the architecture of *MyNet* was already in place it was attempted to train the network from scratch for this classification. But as discussed earlier, the dimensionality of the problem was extremely high and due to limited computational resources, it was not possible to train the network.

After various iterations and cutting down layers in order to get the least functionality up and running, these were the final layers of the *SimpleNet*:

Name	**Layer**	**Filter size**	**Output shape**	**Parameters**
C1	2D convolution	6 – 5x5	[448, 448, 6]	456
S2	Max pooling	5x5	[89, 89, 6]	0
S3	Max pooling	5x5	[17, 17, 6]	0
Flatten	Stack to line	-	1734	0
F5	Fully connected	-	30	52,050
F6	Fully connected	-	10	310

Total parameters = 52,816

As expected, the network performed very poorly with average of upto less than ~25% accuracy. Hence, multiple attempts were made to improve the base accuracy, results are as follows:

- With SGD optimizer and MSE loss function:
  learning rate: 0.01 – loss: 0.1810 – accuracy: 0.1013
  learning rate: 0.1 – loss: 0.1778 – accuracy: 0.1138
  learning rate: 0.3 – loss: 0.1552 – accuracy: 0.1390
- With Sparse Categorical Cross Entropy (SCCE) as loss function:
  learning rate: 0.01 – loss: 13.592 – accuracy: 0.1711
  learning rate: 0.1 – loss: 11.521 – accuracy: 0.2668
- With AdaDelta optimizer and SCCE loss function:
  loss: 13.8194 – accuracy: 0.3400

d. Using a pre-trained model (Transfer Learning from ImageNet challenge):

Amongst a very wide range of available pre-trained model *VGG-16* was chosen for the purpose of this project keeping in mind the size of the network, number of parameter and most importantly that VGG-16 in 2014 was the 1st runner up of the *ImageNet challenge* right after *GoogleNet*.

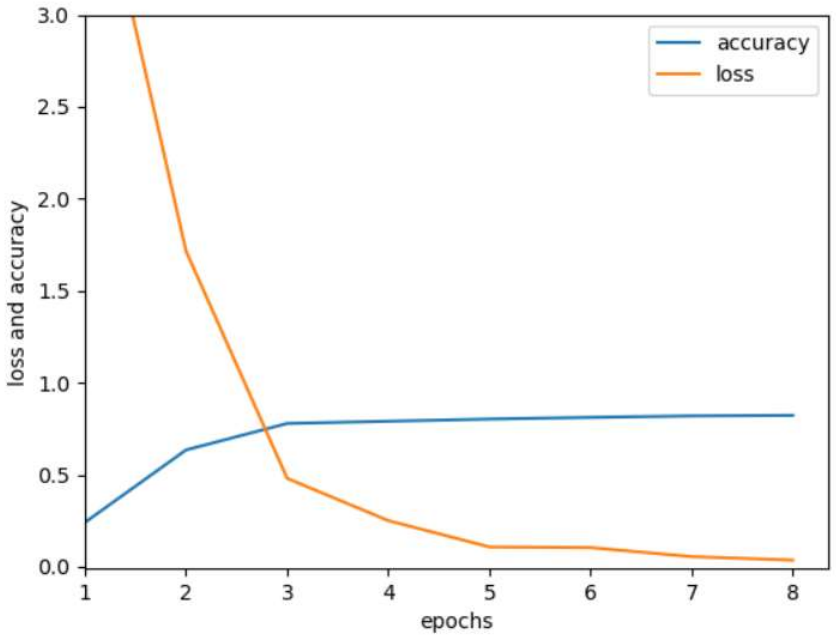
The model was imported from the *Keras* library in *Tensorflow* and the learned parameter (weights and biases) were loaded from the ones obtained from the ImageNet challenge. The top few layers of dense connections (fully connected layers) were then removed to make new layers of same dimensions but untrained parameters to make it work for the new problem of monkey's species classification. Note that VGG-16 only accepted images of size 224x224, 3 channels (RGB) hence, the original images were again preprocessed [2] before feeding into the network and were permanently saved as numpy arrays (*.npy*).

The top 3 layers removed were:

Name	Layer	Output shape	Parameters
fc1	Fully connected	4096	102764544
fc2	Fully connected	4096	16781312
predictions	Fully connected	1000	4097000

The output shape of the last layer (predictions) was then changed to 10 according to the number of classes in the species of monkey. Then the network was evaluated after training only the last layers and the results were fair and acceptable.

Epochs	Loss	Accuracy
1	4.1401	0.241
2	1.7162	0.635
3	0.4806	0.779
4	0.2510	0.791
5	0.1079	0.803
6	0.1043	0.812
7	0.0552	0.820
8	0.0363	0.823



4. CONCLUSIONS:

- Support Vector Machine were used to classify the MNIST dataset and it performed not as good as Convolutional Neural Networks. Given the importance of applications of MNIST classification and its use-cases, SVMs are not a good fit for such classifications. Moreover, the time and computational effort consumed by SVM was huge and unreasonable. Even though, RBF kernel provided good separation between the classes and also increased the efficiency, it was not at par with Deep Learning. Hence, SVMs prove not so good for image classifications as compared to Neural Nets.
- Already researched and tested LeNet-5 (*MyNet* here) in 1998 still proves to be a good classifier for MNIST recognition and achieved a great accuracy of ~97%. This trained model could be further packaged into an application and will perform just as well in the real world. It is also important to understand the problem at hand and tailor the network to suit the needs and hence preprocessing [2] is an important step before starting to look into the architecture of the network.
- The use and motivation for Transfer Learning was completely justified in this project. Given the dimensionality and size of the problem along with limited computing resources, extreme difficulties were faced to try to train a network from scratch. But the use of pretrained model made it easier to solve the task and was fairly easy to configure and use using the vetted Tensorflow and Keras libraries.

REFERENCES:

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha_ner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, Nov 1998.

[2] https://stackoverflow.com/questions/41907598/how-to-train-images-when-they-have-different-size