Fundamentals of Machine Learning Project-1

December 2, 2019

1 Abstract

This report is intended to formulate the hand-written character classification using Convolutional Neural Network. The prima facie of our approach is to achieve an adept methodology using a balanced combination of the fundamental of machine learning course and the state-of-the-art technique used today to classify handwritten characters into classes. In this proposition we have used a standard Five-layer Convolutional Neural Network with batch normalization at each layer and max pooling in the first three layers. The provided dataset with different sizes has been standardized using padding to achieve a uniform dimension dataset. We have compared the performance of the aforementioned CNN with and without the batch normalization to emphasize on the importance of the batch normalization. Along with the aforesaid we have compared the results generated from different combinations of learning rates and batch sizes. To validate the authenticity of the CNN we have employed 10-fold cross validation which ensured a proficient training model. We have reserved some data from the provided dataset to test our model depending on which we evaluate the performance of the trained model.
Keywords- Convolutional Neural Network, Batch Normalization, K-fold Cross Validation, Max pooling, Relu, Softmax

2 Introduction

Handwritten character classification is a prevalent and well coveted topic in academic and industrial domain based on the requisites of the present day. A standard neural network has the ability to replicate the complexity of any function based on the depth of the network and the number of neurons employed in each layer. Though it is well equipped to replicate the behavior of an adequate decision-making algorithm, it does that with a certain level of ambiguity associated with it given the fact that a basic neural network doesn't focus on specific patterns associated with the images fed to it. On the contrary, a CNN (Convolutional Neural Network) has the ability to extract the required patterns depending on the filters associated with each neuron. The output of each layer contains some definite behavior of the input image based on the information stored in each pixel. This is achieved by incorporating a filter in each neuron in the convolution layer which gives the network its name and eventually the number of neurons required to build the network reduces. In other words, CNN can capture the complexities in an image with less number of neurons and less number of layers as compared to a general neural network.

3 Implementation

The dataset used for this application is generated using handwritten texts from random subjects viz. 6400. Along with the given dataset we have used 2000 images from EMNIST (Extended Modified National Institute of Standards and Technology) dataset to improve our accuracy of our network and to train the model for class '-1'. The dataset of 6400 designed by the random subjects from the university are then standardized using padding so as to generate images of same size. We have incorporated false padding to increase the size of each sample to the required size which makes it easier to generalize the number of inputs to the network. On the other hand, the dataset from EMNIST is provided to us in the form of 1D vector. We used false padding in this dataset as well so that it can match the size of the already existing sample. Out of the complete dataset we used three percent of the data as test data, three percent as validation data and the remaining data for training.

4 Architecture

The employed CNN model has eleven layers. The first layer is the input layer which is very generic to any neural network. The second, fourth, sixth and the eighth layer are the convolution layer followed by max pool layers in the third, fifth and the seventh layer. The ninth and tenth layers are fully connected followed by the output layer.

The activation function used in all of the layers is Relu activation. Relu activation is defined by f(x) = tanh(x). We use Softmax in the output layer, while the other activation functions get an input value and transform it regardless of the other elements, the Softmax considers the information about the whole set of numbers we have.

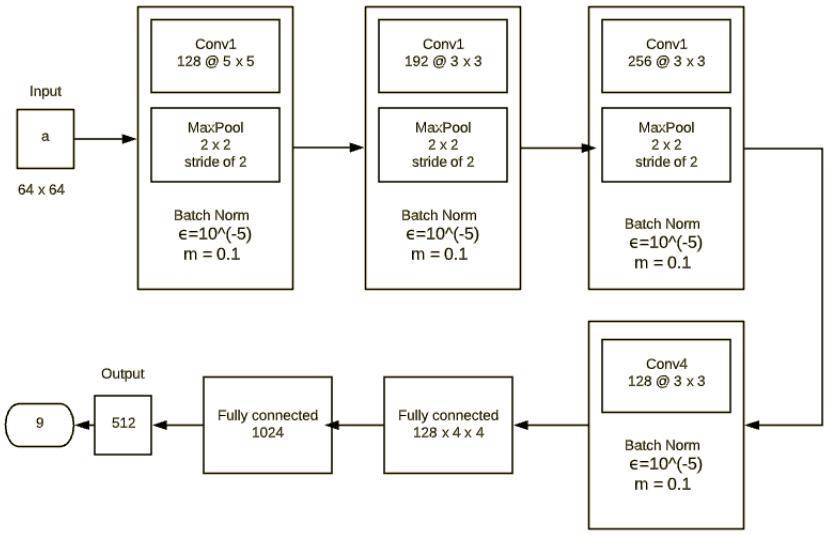

Figure 1: Architecture

Normalization as a concept is very synonymous with any machine learning algorithm that we deal with. Normalization or standardization is a process which eliminates the bias induced due to uneven range of features of a sample. If the features of a dataset have ranges that are very different, then the weights associated with the feature which has the highest range dominates the model and the decision made by a model tends to ignore the contribution of the feature with less range. We normalize the output of the neurons in the intermediate layers as well. This helps us to ameliorate the problem of uneven weight updates in the previous layer which dominates the behavior of the later layers. We term it batch normalization because we normalize the output of each layer according to the batch of data used. In our experiment, we have compared the decision of CNN with and without batch normalization.

After every convolution layer we incorporated a Maxpool layer. Maxpool layer uses a predefined mask which is shifted throughout the output of the previous layer and the maximum of the output overlapping with the mask is selected. By doing so, the output of the Maxpool layer preserves the dominant information from the previous layer and reduces the size which in turn enhances computation speed.

We have implemented a dropout measure which reduces the output of every neuron to zero if the output of that neuron has probability less that 0.5. This reduces the dependence of the network on those neurons. In case of different neurons being dropped for different inputs, the network updating the synaptic weights of different neurons results in creating a very robust weight updating process which reduces the dependence of neurons on each other. This is because the set of neurons active for different inputs are different. This has proved to be very productive for our network.

5 Experiment

Our dataset consists of 8400 samples out of which 6400 were made from random samples collected from our class and 2000 were downloaded from EMNIST dataset. Each data sample has been converted to a dimension of 64 X 64 pixels which is the standard we have opted for. The dataset consists of the letters a, b, c, d, h, i, j and k from the list of English alphabets and we need to classify them into classes 1, 2, 3, 4, 5, 6, 7 and 8 respectively. Apart from the above mentioned we have to describe a class '-1' for anything that does not belong to the above classes. As mentioned above we have used a Convolutional Neural Network with multiple layers to achieve our target. We have trained our network empirically for 12, 20, 25 and 50 epochs to decipher the conversions of global minima pertinent with our dataset. Along with the aforesaid, we have trained our model with a roster of 8 different combinations of learning rates and batch sizes viz. 0.001, 0.003, 0.005 and 0.01 as learning rates, and 32 and 64 as batch sizes for our complete dataset.

type	Patch size/ stride	#channels	Output size	params
Input				
convolution	5 X 5/ 1	128	128 X 60 X 60	3328
max pool	2 X 2/ 2		128 X 30 X 30	
convolution	3 X 3/ 1	192	192 X 28 X 28	221376
max pool	2 X 2/ 2		192 X 14 X 14	
convolution	3 X 3/ 1	256	256 X 12 X 12	442624
max pool	2 X 2/ 2		256 X 6 X 6	
convolution	3 X 3/ 1	128	128 X 4 X 4	295040
fully connected			1 X 1024	
fully connected			512 X 1024	524800
Output(SoftMax)			1 X 9	5120

Figure 2: Parameters

6 Conclusion

Out of the two batch sizes we have selected, we have found that a mini-batch of 64 samples proved to be adept for our network. With a learning rate of 0.001, the network without batch-normalization surfaced an accuracy of 97.469%. With the same mini-batch size of 64 samples, the network with batch-normalization along with the learning rate of 0.01 produced a result with 92.31% accuracy on test data set. Similarly, with the same mini-batch size as input to the network with dropout and learning rate 0.001 produced a result of 72.43%. It is evident from the plot of accuracy vs run (pp4) that the network without batch normalization generated the best possible result on the validation dataset. Since the network without batch normalization produced the best results, we have determined the accuracy and loss for different epochs on training dataset. This can be deduced from the plot of accuracy vs epoch loss vs epoch (pp4). Finally, the performance of the network without batch normalization has been measured using the confusion matrix on the test dataset. This is shown in confusion matrix (pp4).

7 References

1. LeCun, Yann. "LeNet-5, convolutional neural networks." URL: http://yann. lecun. com/exdb/lenet 20 (2015): 5.
2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
3. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).
4. https://pytorch.org/

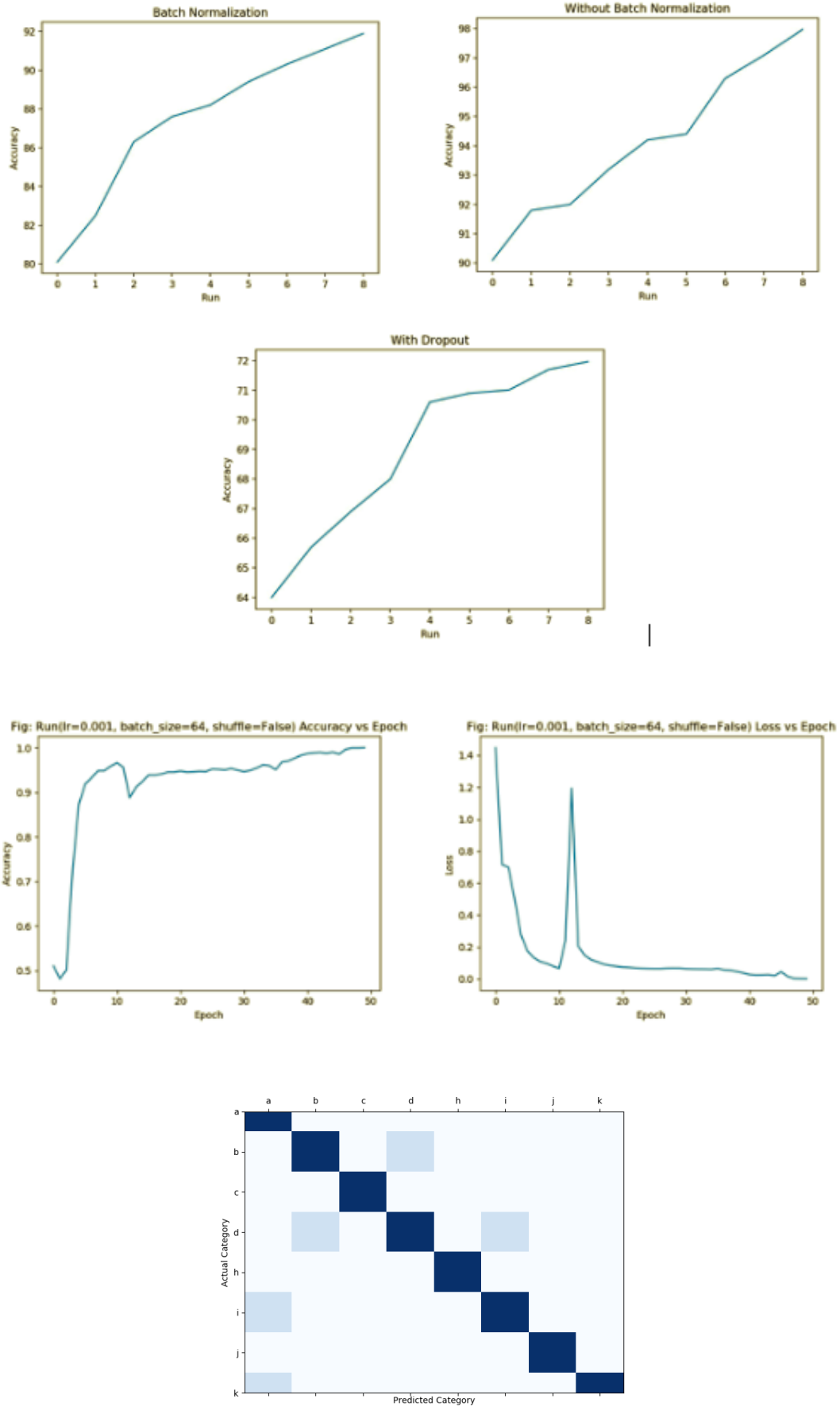

Figure 4: Top left: Batch normalization; Top right: Without batch normalization; Center: With Dropout; Bottom Left: Accuracy vs epoch; Bottom Right: Loss vs epoch; Bottom Center: Confusion matrix