MLP Coursework 2: Learning rules, BatchNorm, and ConvNets

s1717961

Abstract

Deep learning has become increasingly crucial in training complex models like convolutional network and deep neural network. The main focus of this paper is trying to compare learning rules, batch-normalization in deep systems and convolutional schemes. All these experimental results were obtained through training deeper networks in terms of EMNIST digits and letter dataset. The impact of different learning rules, adding batch normalization and depth of convolutional networks are regarded as major research problems. According to related analysis involved in experiments including corresponding graphs, deeper network with two convolutional layers and max poolings outperforms deep neural networks introduced in this report and Adam learning rule as well as adding batch-normalization help to achieves better outcomes.

1. Introduction

The main objectives of this paper are focused on two aspects, investigating deep neural networks' performance with respect to impact of different learning rules and adding dropout or batch normalization, and comparing a convolutional networks' behavior with one and two convolutional layers. In order to form valid comparison, a baseline system has been built through some basic procedures. In general, hyper-parameters setting like learning rate, choice of activation function and if dropout should be added in this baseline system are decided according to results of validation sets. In the end of this report, test sets will be used to report the performance of the final selected model that are chose on the basis of a range of experiments.

Five main sections will consist of the main content of this paper. Firstly, how to build a baseline system will be explained in the following part. Although the structure of this system is involved with a wider range of choices, section two is expected to display a few setting decisions in detail and show the reasons why specific choices are made. Then, relevant questions about better learning rules among stochastic gradient descent, RMSProp, and Adam, and whether to use batch normalization in building the deep baseline system so as to achieve decent performance are supposed to be discussed in section 3 and 4 respectively. Thirdly, unlike deep neural networks that are introduced in the first three sections, convolutional networks containing one and two convolutional layers will be built in experiments in sections five without changing most systems' settings which can investigate impact of depth in convolutional networks. After going through these procedures, appropriate models should be selected based on results of validation data, section 6 is used to show a final evaluation of accuracy using the test sets regarding selected optimal models. Finally, experiments and related outcomes of each section are summarized in final part.

Besides, a EMNIST (Extended MNIST) Balanced dataset is used to train and test in all experiments in this paper. To be precise, a reduced label set of 47 classes rather than all 62 classes in EMNIST are elected as dataset considering some confusions between upper-case and lower-case letters. The EMNIST digits dataset has 100,000 training images and the same numbers of 15,800 validation images and test data each showing a 28ÃŮ28 grey-scale pixel image of one of the 26 letters and 10 digits. Plus, in order to standardize the network architecture, the batch size is set to 100, 100 epochs are trained and 100 hidden units are included for per hidden layer in terms of all experiments. In this report, we apply held-out validation and test sets that composed of examples coming from the same distribution as the training set. After all hyper-parameter optimization is complete in the first few sections, the generalization error may be estimated using the test set.

2. Baseline systems

In order to make comparisons with the following sections distinct learning rules and networks with batch normalization, a basic deep neural network is built as a reasonable reference. SpeciiňĄcally, training data is divided into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is the validation set, used to estimate the generalization error during or after training, allowing for the hyper-parameters to be updated accordingly.

At first, in order to ensure comparability, this deep network consists of two hidden layers just like the network architecture of section five's one of convolutional networks. Next, 5 activation functions, sigmoid, ReLU (Restricted Linear Unit) and three variations of ReLU those using a non zero slope, Leaky ReLU, ELU (Exponential Linear Unit) and SELU (Scaled Exponential Linear Unit) are included in the neural network so as to find one appropriate function for this baseline system. After three runs of the network (ensure valid results rather than by accident), average values of final outcomes of three aspects, training set error, validation error and validation accuracy are illustrated in table1. It shows that ReLu and its variants fail to appear obvious

differences while sigmoid function shows unsatisfactory results. However, One obvious weakness to ReLU is that as for cases for which their activation is zero, they lose learning ability via gradient-based methods because a rectifying linear unit is zero for half of its inputs. LReLU is selected as the activation function for the latter all experiments due to less parameter settings.

Final Results	Sigmoid	Relu	LRelu	ELU	SELU
Error(train)	1.52	0.565	0.574	0.596	0.568
Error(validation)	1.53	0.612	0.612	0.636	0.617
Accuracy(validation)	0.583	0.809	0.810	0.806	0.811

Table 1. Results for five activation functions

2.1. Learning rates

Also, stochastic gradient learning rule is chosen to be the default learning rule for the baseline system which will form comparisons with latter two learning rules, RMSProp and Adam. At the same time, A crucial parameter for the SGD algorithm, the learning rate of this basic network should be set carefully which seems to control the stride length of gradient descent. For example, we may miss the optimal data point and the cost function will increase significantly if this rate was set to be too large whereas learning may become stuck with a high cost value if learning rate was too small(Bishop, 2014).

Typically, 0.01 is set to be the starting value, I try to optimize this model with a large learning rate (0.01 or so), and then progressively reduce this rate(0.01, 0.005, 0.0025). Gaps between training and validation accuracy (overfitting) narrow as the learning rates decrease even though there are just little overfitting, which can be seen from the following figure 1 that describes classification accuracy. In particular, the green and yellow lines come together in the end when learning rate is 0.0025 though its accuracies in these 100 epochs increase slowly in comparison with those of other rates because the low rate makes it slow to converge. Thus, this value is used to be the default learning rate of the baseline network and applied in latter all experiments in order to discourage overfitting. In addition to this, regularization is often used to mitigate overfitting behavior through changing cost functions and weights of networks while dropout reduces overfitting by changing network itself.

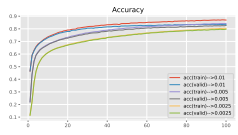


Figure 1. Accuracy of three learning rates

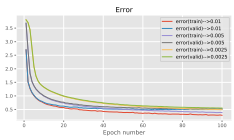


Figure 2. Error of three learning rates

2.2. Dropout

Regularization has been well known for most machine learners which introduces a new lamda variable into cost function as well as penalizes the weight. So this paper will display a different method and compare the potential influence of adding dropout in each hidden layer in the baseline system through randomly deleting half the hidden neurons in the network, whereas leaving the input and output neurons untouched. In fact, it seems like more networks will be trained with dropout, and the different networks will overfit in different ways(Nielsen, 2017), in the hope that the net effect of dropout will be to reduce overfitting. In this trail, baseline system is used when keeping learning rate, learning rule, hidden units number, depth and activation function so on as before chosen. Clearly, as we can seen from the following two figures(figure 2 and figure 3), dropout does mitigate the overfitting behavior since smaller gaps are obtained but network with dropout appear bad performance with lower classification accuracies and higher errors. As a result, dropout will not be added in latter experiments in order to discourage overfitting in the consideration of its unsatisfactory behaviors. However, batch normalization, another method used to avoid overfitting, will be introduced in section four.

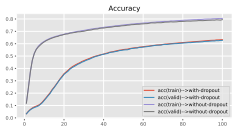


Figure 3. Accuracy of network with/without dropout

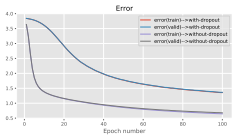


Figure 4. Error of network with/without dropout

Finally, the baseline system in this paper is made up of two hidden layers without dropout, SGD learning rule, 0.0025 learning rate, Leaky ReLU activation function, 100 batch size, 100 epochs and 100 hidden units are included per hidden layer in all experiments.

3. Learning rules

This section is focused on one research question mentioned in introduction, investigating potential impacts of distinct learning rules on EMNIST Balanced based on earlier baseline experiments when all other hyper-parameters are kept the same as before. Theoretical background of these three learning rules is expected to be explained simply for the sake of readily and understandability. Also, trials of three learning rules are conducted on both the same training set and validation set in order to observe their different performance.

The relevant equations of default learning rule stochastic gradient descent and two other learning rules are displayed as follows:

Stochastic(or "on-line") gradient descent (SGD) is probably the most popular optimization algorithms for machine

learning in general and for deep learning(John Duchi, 2011). One of its features is that if the objective function is convex, SGD may converge to a global minimum but it may converge to a local minimum due to different initial learning rate. It is clear from figure 5 and 6 that the accuracies of SGD start with the lowest value than accuracies of other two rules and errors are opposite.

SGD (Stochastic Gradient Descent):

$$\Delta w_i(t) = -\eta g_i(t) \tag{1}$$

the next weight is:

$$w_i(t+1) = \Delta w_i(t) + w_i(t) \tag{2}$$

Additionally, Learning rates have been regarded as a very difficult value to set as it will affect model performance dramatically(Bishop, 2014). So this section will further describe two other variants of it using adaptive learning rates instead of hand picking manually as in SGD(Bottou, 2012), in spite of all of them select a random sample from training set.

RMSProp

$$M_i(t) = \alpha M_i(t-1) + (1-\alpha) g_i(t) \tag{3}$$

$$S_i(t) = \beta S_i(t-1) + (1-\beta) g_i(t)^2 \tag{4}$$

$$\Delta w_i(t) = \frac{-\eta}{\sqrt{S_i(t)} + \epsilon} M_i(t) \tag{5}$$

Actually, the main difference between RMSProp and Adam is whether it includes bias-correction terms which may lead to infinitely large bias in some cases.

Adam

$$S_i(t) = \beta S_i(t-1) + (1-\beta) g_i(t)^2 \tag{6}$$

$$\Delta w_i(t) = \frac{-\eta}{\sqrt{S_i(t)} + \epsilon} g_i(t) \tag{7}$$

Apparently, networks with RMSProp and Adam learning rules achieve better performance regardless of validation accuracies or errors, which we can see from the close four lines from red line to light green in figure 5 and 6 as we expected. In other words, they eventually converge considerably faster than SGD.

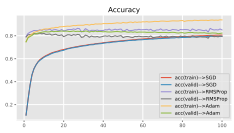


Figure 5. Accuracy of three learning rules

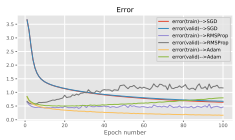


Figure 6. Error of three learning rules

Average value	SGD	RMSProp	Adam
Error(train)	0.639	0.464	0.163
Error(validation)	0.674	1.16	0.8
Accuracy(validation)	0.796	0.793	0.823

Table 2. Results for three learning rules

Specifically, Adam demonstrates better convergence than other methods and RMSProp comes the second according to table 2. Adam shows marginal improvement over RMSProp even though both of them appear overfitting on the basis of gaps between outcomes of training sets and validation sets and slightly decrease in accuracies of validation data(see light green line and grey line). Thus, Adam was selected as the learning rule for the latter experiments based on the results demonstrate in table 2, it obtain the highest average validation classification accuracy with 0.823, about 0.03 more than that of RMSProp and SGD. Thus, Adam was selected as the learning rule for the latter experiments. It is not only because the method is straightforward to implement and requires little memory, but also because it takes the bias-correction terms into account so that achieve better performance. However, appeared overfitting in experiments should not be ignored, the next section will describe an effective method batch normalization in order to prevent overfitting.

4. Batch Normalization

The focus of this section is to explain one of research questions of this paper, batch normalization method. At first, this method tries to obtain zero mean version for each training mini-batch, which is similar to the preprocessing of inputs using Principal Components Analysis (PCA). For instance, this version may be gained through deducting mean of each row of inputs. To accomplish this, experiments in this section tried to build two neural network model with two fully connected hidden layers with 100 hidden units each, LReLU activation, Adam learning rule and 0.0025 learning rate are used for this experiment with minibatch size of 100. Setting is similar to previous experiments except that one of networks added batch normalization layer before each LReLU activation. Then, these normalized activations in this network have the same mean 0 and variance 1 as during training. Detailed procedures of batch normalization method is illustrated in the following equations divided by forward and backward operations.

4.1. Equations

Batch Normalization

forward procedure(Ioffe & Szegedy., 2015):

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{8}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{9}$$

$$\widehat{x_i} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{10}$$

$$y_i = \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{11}$$

backward procedure:

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i}\cdot\gamma \tag{12}$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m}\frac{\partial \ell}{\partial \hat{x}_i}\cdot(x_i - \mu_{\mathcal{B}})\cdot\frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{\frac{-3}{2}} \tag{13}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = (\sum_{i=1}^{m}\frac{\partial \ell}{\partial \hat{x}_i}\cdot\frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2}\cdot\frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m} \tag{14}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{a}_i}\cdot\frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2}\cdot\frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}}\cdot\frac{1}{m} \tag{15}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m}\frac{\partial \ell}{\partial y_i}\cdot\hat{x}_i \tag{16}$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m}\frac{\partial \ell}{\partial y_i} \tag{17}$$

Simply, this method addresses its advantage from making normalization a part of the model architecture by fixing the distribution of the layer inputs because it has been convinced that the network training converges faster if its inputs are whitened(et al, 1998b). We can see from red and blue lines in the following figure 7 and 8, this method not just speeds up convergence but also regularizes the model.

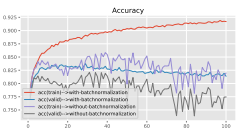


Figure 7. Accuracy of network with/without batch normalization

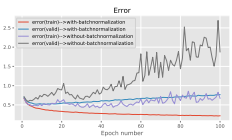
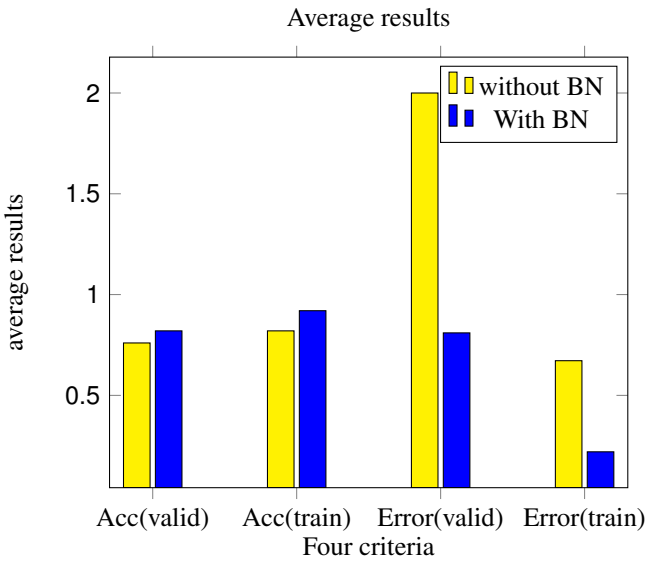


Figure 8. Error of network with/without batch normalization



In particular, validation accuracy of network with batch normalization decrease from the highest value 0.84 to around 0.82 in the end while the another model's validation accuracy drops by more than 0.6 when its highest number is 0.82(the lowest value in the network with batch normalization). Similarly, validation error rises considerately(the gray line) from 0.6 to more than 2 whereas this error increase by only 0.3 in terms of the model adding this method. In addition, gaps between training sets and validation sets narrow obviously with respect to deep neural network using batch normalization, which means that overfitting behavior has been effectively discouraged using this method.

5. Convolutional networks

The impacts of three learning rules and batch normalization method on deep neural network on the basis of previous built baseline system have been explained in the above sections. As the heart of advances in deep learning, convolutional neural networks(CNNs) will be described briefly here plus the description of max pooling in the hope to form comparison with the above best model then find better model in the next test section. To do this, two networks containing one and two convolutional layers will be built when all other parameter settings keep the same as the baseline system, such as Adam learning rule, 0.0025 learning rate, LReLU and 100 hidden units each hidden layer.

5.1. Description of convolutional networks

In general, inputs will be divided into mini bathes(the first for loop of codes happened here) and three channels of inputs under each batch will be conducted(the second for loop). In contrast with fully connected hidden layer, hidden layer in a convolution is connected to a specific region of input space called local receptive field, then a shared weights matrix called kernel with these local receptive fields will result in a feature map, in particular, signal.convolve2d was used to computed the product between each element of the kernel and the input element it overlaps then these outputs

will be summed up element wise. After these procedures, max pooling is supposed to be applied in order to take the maximum value in a small region(like 2*2 in trails of this section) of the above feature map so that output size is reduced further, which could help to prevent overfitting partly due to deducting some data.

Plus, unlike fully-connected layers as we built in previous deep neural network, the output size of a convolutional layer is dependent on the input size because its output shape is supposed to be impacted by the input shape and relevant kernel shape. Besides, a convolution can be regarded as a linear transformation and use parameters multiple times like the same weights.(Vincent Dumoulin, 2016) The most difference is that only a few input units rather than all of them produce a given output.

5.2. Experiments for convolutional networks

In this experiment, I try to compare behavior of two networks with one and two convolutional layers as well as max pooling layers so as to check whether different depth affects the final performance. As before, nearly all parameter settings remain unchanged in order to ensure comparability, like 0.0025 learning rate, LReLU activation and Adam learning rule, 100 batch size and 100 epochs so on. Specially, zero padding and 1 strides length are applied in expected two convolutional models, at the same time, strides 2 are adopted in max pooling so as to avoid overlapping. Specifically, In the convolutional network with two convolutions, there are default 5 feature maps in the first convolutional layer and 10 feature maps. In addition to this, inputs need to be reshape into 1 dimension before convolutional layer and final outputs also need to be reshape again.

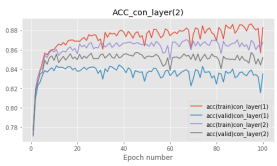


Figure 9. Accuracy of of network containing one and two convolutional layers

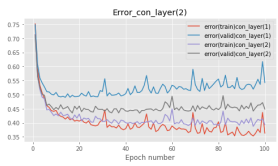
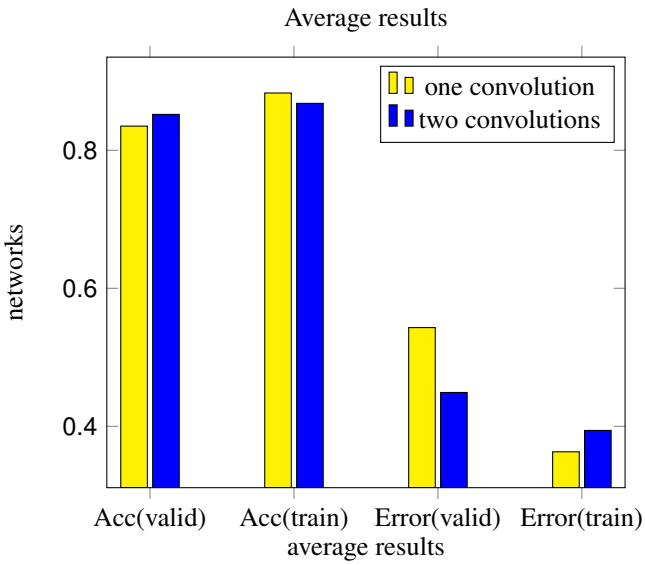


Figure 10. Error of network containing one and two convolutional layers

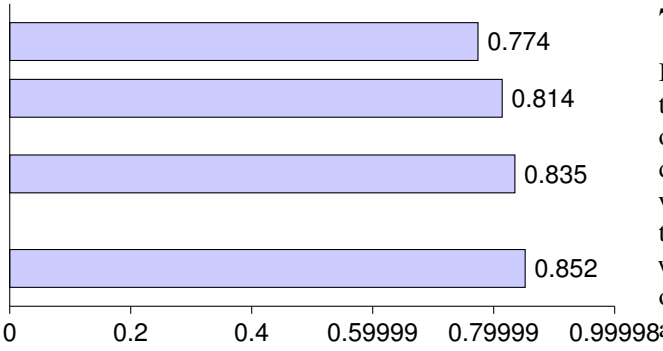


According to the above graphs, it is obvious to note that validation accuracy of two convolutional layers model is 0.07 more than that of another convolutional network and its excellent performance also can be see from lower validation error even though model of two convolution and two max pooling fails to remain such appealing behavior in training set. Therefore, convolutional model with deeper architecture achieve better performance in this experiments between one convolution, max pooling and two convolutions and two max poolings. Apart from this, deeper convolutional model may discourage overfitting to some extent because gaps between red and blue lines of network with one convolution is bigger than gaps between errors and accuracies of training set and validation sets from purple and gray lines in figure 9 and 10.

6. Test results

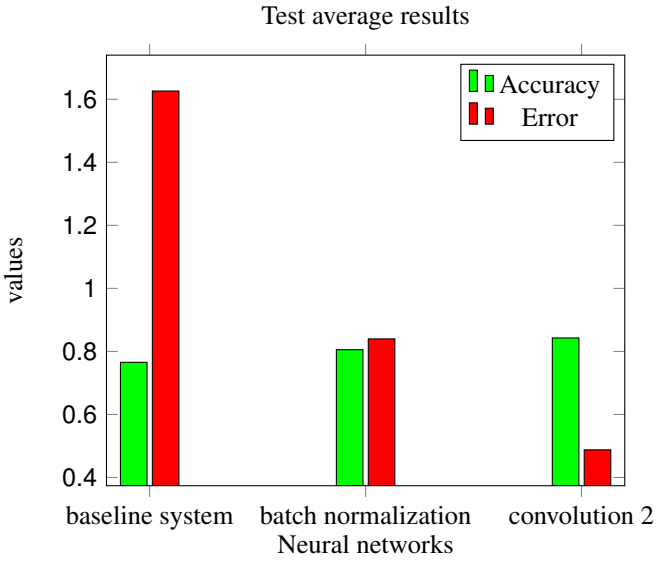
6.1. Best models

It is important that test examples should not be used in any way to make choices about the model so that I decided to choose models based on the validation accuracy. In a nutshell, this paper built main four models with two deep neural networks and two convolutional networks as follows:
model 1: one deep neural network with Adam learning rule with the lowest validation accuracy at 0.774
model 2: model 1 adding batch normalization comes the second in terms of classification accuracy in validation sets
model 3: model with one convolution reaches 0.835
model 4: network containing two convolutions obtain the highest validation accuracy with 0.852



6.2. Test experiments

Clearly, the deep architecture with batch normalization is regarded as the best deep network without convolutional layers and deeper convolutional network beats another convolutional network with only one convolution and max pooling. Thus, test results on selected best models are displayed in the following histogram. In particular, model 1 is shown as well because considering the impact of adding batch normalization which is so comparable with deeper convolutional network. The same size of validation set 15800 test data were applied to test final models actual behaviors and obtain test average values in the experiment as all settings are consistent with selected models. We can be seen from the following histogram as red represents errors and green represents accuracies. Just like we expected, this histogram shows that the model 1 achieve the lowest average test accuracy but the highest test error with nearly twice than error of model 2 not to mention that the batch-normalized network enjoys higher test accuracy. However, the biggest test accuracy was achieved by deeper convolutional network with around 0.85 as well as very low average test error(0.48). In short, test results of all of them are close to validation accuracy with only 0.1 difference. As a consequence, the final models chose by validation sets regardless of deep neural or convolutional network agree well with the test sets due to little generalization error.



7. Conclusions

In conclusion, convolutional network with two convolutions achieve the best performance in comparison with not only the shallower convolutional model but also introduced deep neural networks explained in this paper which can be verified as well on test sets. Plus, adding batch normalization does help model to perform better in my experiments with higher(more than 0.2) validation accuracy and obvious lower errors. Performance of Adam learning rule with adaptive learning rates beats that of similar learning rule RMSProp due to added bias correction terms and outcomes of default SGD learning rule in the earlier baseline system.

Actually, there are some changes with regards to initial baseline system because default SGD learning rule has been changed to Adam when Adam goes better with deep neural network on the basis of EMNIST (Extended MNIST) Balanced dataset as most literatures displayed. It further becomes the new baseline system in latter experiments described as model 1 in section 6. Also, Merely adding Batch Normalization to this new model yields a substantial improvement which leads to close excellent performance with the best model confirmed in this paper. In fact, this report concentrates on the comparison between the baseline, batch-normalized networks and convolutional networks rather than achieving the state of the art performance on EMNIST (which the described architecture does not though I try to get better behavior).

However, the precise effect of Batch Normalization on the convolutional model remains an area of further study. At the same time, dropout in this report fails to help train networks in improved performance though it does mitigate overfitting trend. Its percentage of dropout data may be changed instead of 50% in experiments in this paper. Besides, hold-out test sets may fail to reflect enough generalization because of similar pattern with training sets. So other test sets could be considered so as to get more practical generalization error.

References

Bishop, Christopher. *Machine Learning and Pattern Recognition*. Springer-Verlag New York, 2014.

Bottou, Leon. *Stochastic Gradient Descent Tricks networks*. Springer, 2012.

et al, LeCun. *Neural Networks: Tricks of the trade*. Springer, 1998b.

Ioffe, Sergey and Szegedy., Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

John Duchi, Elad Hazan, Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. 2011.

Nielsen, Michael. *Neural network and Deep Learning*. MIT Press, 2017. http://neuralnetworksanddeeplearning.com.

Vincent Dumoulin, Francesco Visin. A guide to convolution arithmetic for deep learning. 2016.