EECS 542 (Computer Vision) - HW1
Convolutional Neural Networks

Yash Bhalgat (Uniqname: yashsb)

September 24, 2017

1 Part 1

Codes for the following layers written and checked:
Convolution, Pooling, Linear, Leaky_ReLU, Batch Normalization, softmax and loss_ crossentropy.

Batch normalization backprop equations taken from [1].

**Results and plots shown together in Part 2

2 Part 2

In this part, I trained and tested the network over the MNIST dataset. Apart from a basic baseline model, the effects of using leaky ReLU and batch-normalization are also reflected.

2.1 Baseline model for the CNN

2.1.1 Baseline Model

Input size -> [28,28,1]	
Layer 1 output size -> [24,24,8]	...{Conv layer}
Layer 2 output size -> [11,11,8]	...{Max Pooling layer}
Layer 3 output size -> [11,11,8]	...{ReLU activation}
Layer 4 output size -> 968	...{Flatten}
Layer 5 output size -> 10	...{Linear layer}
Layer 6 output size -> 10	...{Softmax layer}

Decisions and Reasoning:

1. A filter size of 3x3 to 7x7 is appropriate to extract the features. After some tuning, 5x5 turned out to be the best filter size.
2. To keep the higher order features intact, I chose a pooling grid of size 4x4 with a stride of 2 (so overlapping windows).
3. I could use multiple (two) fully connected layers. But there was no significant increase by adding a linear layer. Hence the linear layer size was chosen to map the features from 968-dim to 10-dim
4. Softmax layer is used at the end to output a set of probabilities for each class to choose the maximum as the prediction

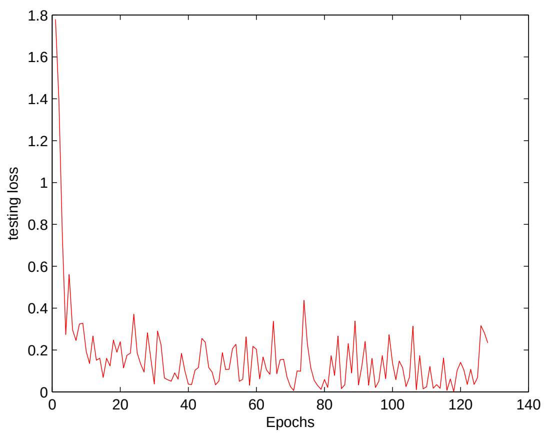
2.1.2 Hyper parameters of the network

In this assignment, the test set is used as the validation set. Hence, all the hyper parameters are tuned over the test set. The best performance is obtained by choosing the following params:

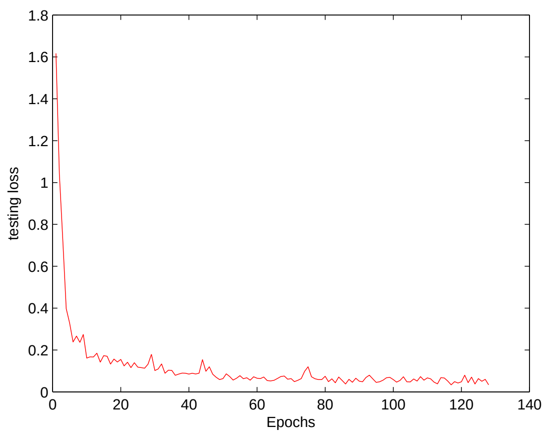
```
batch_size -> 100
lr (learning rate) -> 0.2
wd (weight decay) -> 0.0004
```

2.1.3 Baseline model: Explanation and results

Choosing a smaller batch size implies larger time for each epoch. So, a batch size smaller than 100 did not seem appropriate. Also, a batch size like 1000 gave a 2% dip in the accuracy. While training, each batch was randomly chosen from the training set and passed through the network. The following figures show the performance of the network in terms of training losses, testing losses and accuracy.



(a)



(b)

Figure 1: Training and testing loss across epochs

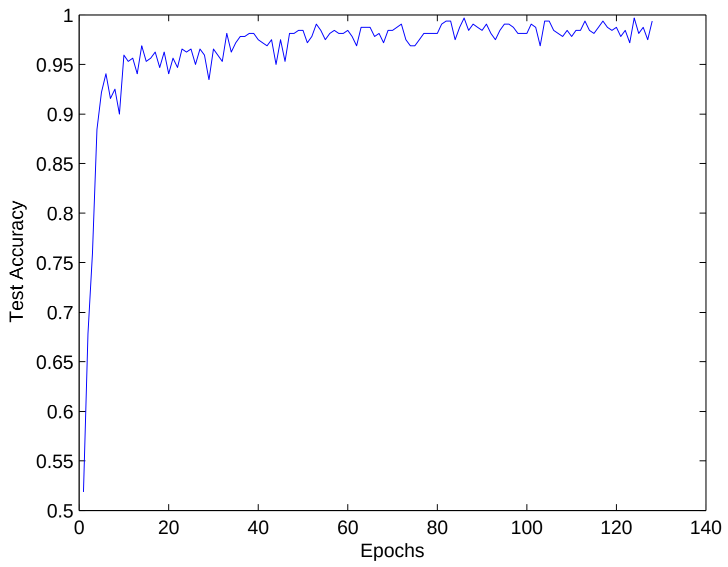


Figure 2: Testing accuracy for the **Baseline Model** across epochs

The maximum accuracy is achieved at 128 epochs (10 iters per epoch).The model reached a maximum test-accuracy of **99.38%**, which greater than the required 96% accuracy in the assignment.

3 Effects of Batch-normalization layer and Leaky-ReLU

According to [1], the batch normalization significantly reduces training time and gives a jump in accuracy especially in very deep networks. So, to test the effect of the BN layer, I ran the problem through a new network - just by adding a BN layer after the convolution layer.

New network with BN layer:
Input size -> [28,28,1]

Layer 1 output size -> [24,24,8]	...{Conv layer}
Layer 2 output size -> [11,11,8]	...{Max Pooling layer}
Layer 3 output size -> [11,11,8]	...{Batch Norm layer}
Layer 3 output size -> [11,11,8]	...{ReLU activation}
Layer 4 output size -> 968	...{Flatten}
Layer 5 output size -> 10	...{Linear layer}
Layer 6 output size -> 10	...{Softmax layer}

As another part of testing, I replaced all the ReLU activations by leaky-ReLU functions. After running through the network while keeping the hyper parameters same, following results were obtained:

	Baseline	Batch-Norm	leaky-ReLU
Accuracy after 1280 iters (%)	**99.38**	97.45	98.82
Train loss after 1280 iters	0.2339	0.3368	**0.2247**
Test loss after 1280 iters	**0.0337**	0.0984	0.0891
# of iters to reach 96% accuracy	350	280	**220**

Table 1: Results and comparison of different methods for feature extraction

It can be observed that Batch normalization doesn't give any advantage, mainly because the network is not very deep and we already have a high accuracy. And the leaky-ReLU is helpful only in reducing the of training epochs required to reach a destination accuracy.

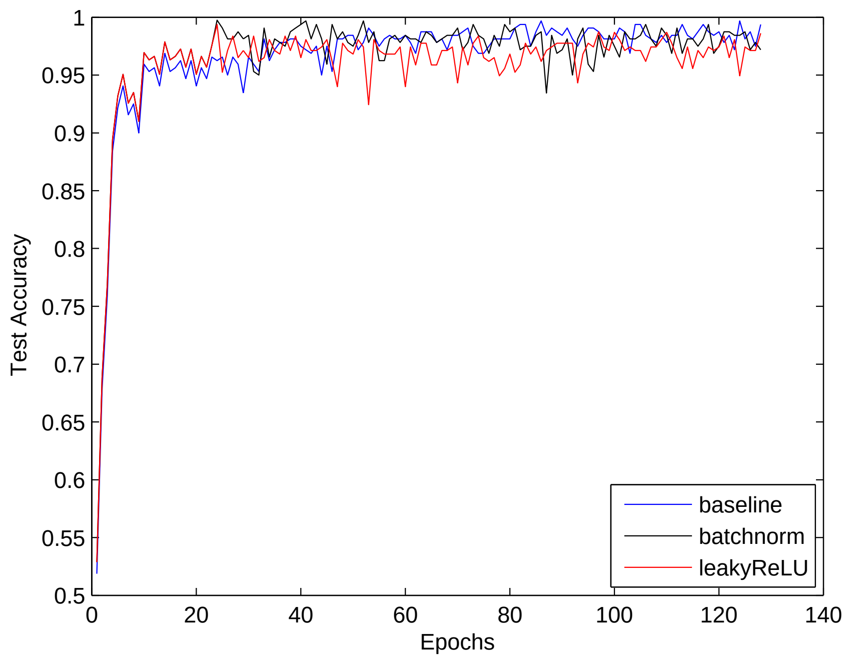


Figure 3: Comparison of test accuracy through different models

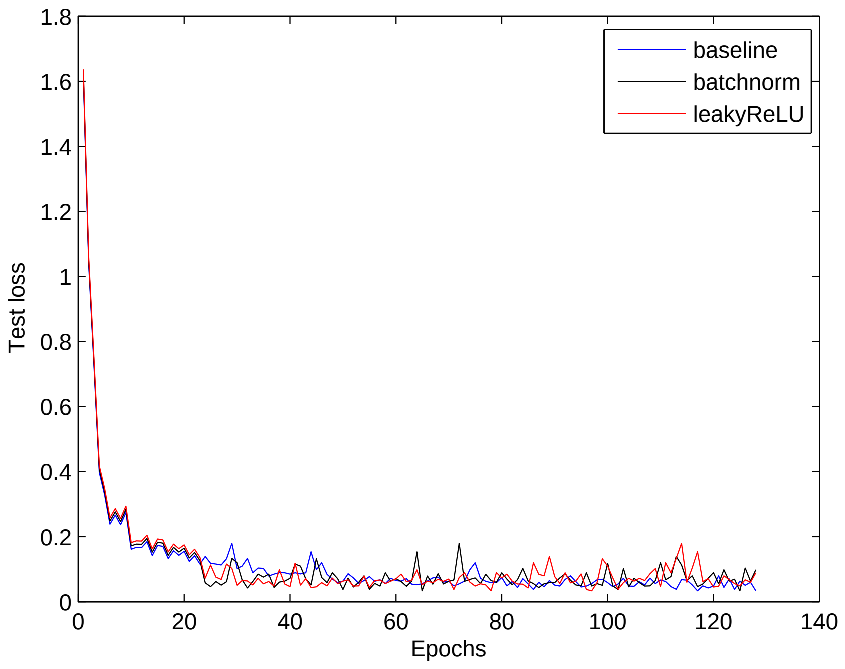


Figure 4: Comparison of testing/validation loss through different models

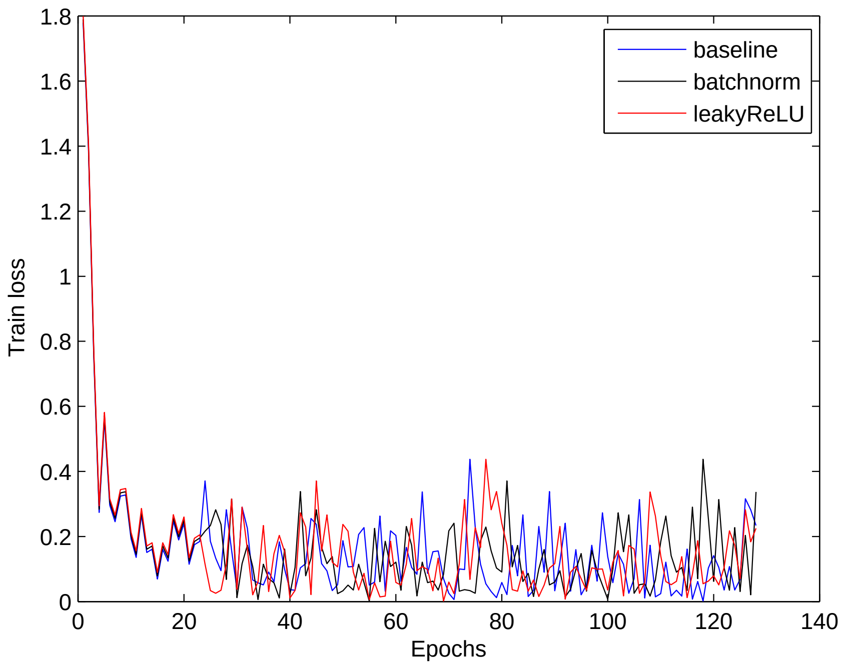


Figure 5: Comparison of training loss through different models

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.