Deep Learning Teaching Kit
Lab 1, Sample Solution 1

1 Backpropagation

1.1 Question 1

Logistic regression:

$$P(y=1|x_{in}) = x_{out} = \sigma(x_{in}) = \frac{1}{1+e^{-x_{in}}}$$

If the error backpropogated to x_{out} is $\frac{\partial E}{\partial x_{out}}$, write the expression for $\frac{\partial E}{\partial x_{in}}$ in terms of $\frac{\partial E}{\partial x_{out}}$.

Answer:

$$\begin{aligned}\frac{\partial E}{\partial x_{in}} &= \frac{\partial E}{\partial x_{out}} \cdot \frac{\partial x_{out}}{\partial x_{in}} \\ &= \frac{\partial E}{\partial x_{out}} \cdot \frac{\partial [\sigma(x_{in})]}{\partial x_{in}} \\ &= \frac{\partial E}{\partial x_{out}} \cdot [1-\sigma(x_{in})] \cdot \sigma(x_{in}) \\ &= \frac{\partial E}{\partial x_{out}} \cdot [1-x_{out}] \cdot x_{out}\end{aligned}$$

1.2 Question 2

Multinomial logistic regression:

$$P(y=i|X_{in}) = (X_{out})_i = \frac{e^{-\beta(X_{in})_i}}{\sum_k e^{-\beta(X_{in})_k}}$$

What is the expression for $\frac{\partial (X_{out})_i}{\partial (X_{in})_j}$?

Answer:

Set $-\beta(X_{in})_i = Z_i$, then $P(y=i|X_{in}) = (X_{out})_i = \frac{e^{Z_i}}{\sum_k e^{Z_k}}$.
First consider $\frac{\partial (X_{out})_i}{\partial Z_j}$.

If $i = j$, then:

$$\begin{aligned}\frac{\partial (X_{out})_i}{\partial Z_j} = \frac{\partial (\frac{e^{Z_i}}{\sum_k e^{Z_k}})}{\partial Z_i} &= \frac{e^{Z_i}(\sum_k e^{Z_k}) - e^{Z_i}(e^{Z_i})}{(\sum_k e^{Z_k})^2} \\ &= \frac{e^{Z_i}}{(\sum_k e^{Z_k})} - (\frac{e^{Z_i}}{\sum_k e^{Z_k}})^2 \\ &= (X_{out})_i - (X_{out})_i^2 \\ &= (X_{out})_i[1 - (X_{out})_i]\end{aligned}$$

If $i \neq j$, then:

$$\begin{aligned}\frac{\partial (X_{out})_i}{\partial Z_j} = \frac{\partial (\frac{e^{Z_i}}{\sum_k e^{Z_k}})}{\partial Z_j} &= e^{Z_i} \cdot \frac{\partial (\frac{1}{\sum_k e^{Z_k}})}{\partial Z_j} \\ &= e^{Z_i} \cdot \frac{-e^{Z_j}}{(\sum_k e^{Z_k})^2} \\ &= -\frac{e^{Z_i}}{\sum_k e^{Z_k}} \cdot \frac{e^{Z_j}}{\sum_k e^{Z_k}} \\ &= -(X_{out})_i \cdot (X_{out})_j\end{aligned}$$

Since $\frac{\partial (X_{out})_i}{\partial (X_{in})_j} = \frac{\partial (X_{out})_i}{\partial (Z_j)} \cdot \frac{\partial (Z_j)}{\partial (X_{in})_j}$, then:

If $i = j$:

$$\frac{\partial (X_{out})_i}{\partial (X_{in})_j} = \frac{\partial (X_{out})_i}{\partial (X_{in})_i} = (X_{out})_i[1 - (X_{out})_i] \cdot (-\beta)$$

If $i \neq j$:

$$\frac{\partial (X_{out})_i}{\partial (X_{in})_j} = -(X_{out})_i \cdot (X_{out})_j \cdot (-\beta) = \beta \cdot (X_{out})_i \cdot (X_{out})_j$$

1.3 Question 3

According to the DAG,

$$y = max(\sigma(x_1), \sigma(x_2)) + min(\sigma(x_1), \sigma(x_2))$$

Suppose the error passed down to y is $\frac{\partial E}{\partial y}$. Write the values for $\frac{\partial E}{\partial x_1}$ and $\frac{\partial E}{\partial x_2}$ in terms of $\frac{\partial E}{\partial y}$.

Answer:

$$\begin{aligned}\frac{\partial E}{\partial x_1} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial x_1} \\ &= \frac{\partial E}{\partial y} \cdot \frac{\partial [\sigma(x_1) + \sigma(x_2)]}{\partial x_1} \\ &= \frac{\partial E}{\partial y} \cdot [1 - \sigma(x_1)] \cdot \sigma(x_1)\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial x_2} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial x_2} \\ &= \frac{\partial E}{\partial y} \cdot \frac{\partial [\sigma(x_1) + \sigma(x_2)]}{\partial x_2} \\ &= \frac{\partial E}{\partial y} \cdot [1 - \sigma(x_2)] \cdot \sigma(x_2)\end{aligned}$$

2 MNIST Handwritten Digit Recognition

2.1 Data

The MNIST dataset of handwritten digits has a training set of 60000 samples and a testing set of 10000 samples. We split the training dataset 70% training and 30% validation by random sampling:

- Training: 42000
- Validation: 18000
- Testing: 10000

2.2 Model Structure

The ConvNet architecture is based on Sermanet's work [1], which is composed of two stacked feature stages. Each stage contains a convolution module, followed by a pooling module and a normalization module. And the final stage is a 2-layer fullyconnected neural network to do the classification.

Our final model consists of 2 convolution layers with 128 hidden units, spatial lp-norm pooling whose size is 3 by 3 and sliding is 2 by 2, convolution filter whose size is 5 by 5, and followed by a 2-layer fully-connected classification layers with 256 hidden nodes. Each layer, including convolution and classification, has a dropout layer with 0.5 factor as regularization. On top of that, we use *tanh* as activation, SGD as optimizer, and normalized log-likelihood as loss function.

2.3 Training

We implemented the model on Torch. As shown before, we split the training data into training and validation and we controlled the generalization of the model by referring its performances on validation data. We train the model with the pure stochastic way with batch size equal to one.

2.4 Experiment Results

In order to save time, we ran experiments on a small dataset which contains 10% of full data. We explored different model structure (more layers, more kernels, different filter sizes, different pooling sizes, adding dropout layers, etc.) as well as different settings (different non-linearities, different optimizers, different batch sizes, etc.). And results are shown as the following.

2.4.1 Kernel Size

The great effects from the change of hidden units can be shown in Figure 1. As we keep increasing the hidden units within each of the layers, we obtain faster convergence for the training set. The validation set is also taking advantage of the larger number of hidden units.

2.4.2 Number of Layers

We also investigated the effects of adding more convolution layers. From figure 2 we can find that more layers boost the prediction accuracy. The interesting point here is that when we use only one convolution layer, it has faster convergence on training error, but the model can not be generalized to the validation set and thus gets a higher validation error. It's obvious that it overfits the training data in the non-deep features space.

2.4.3 Batch Size

As we increase the batch size during training, both training and validation error also increase significantly, which is shown in Figure 3. The reason is probably that higher batch size needs more epochs for convergence.

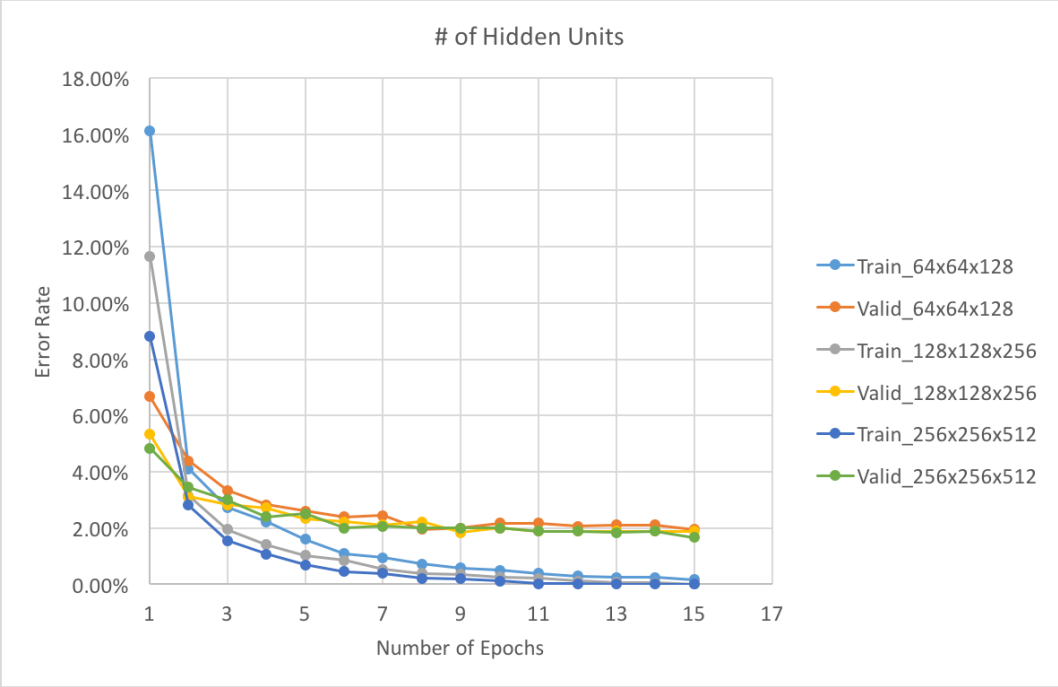

Figure 1: Different Hidden Units

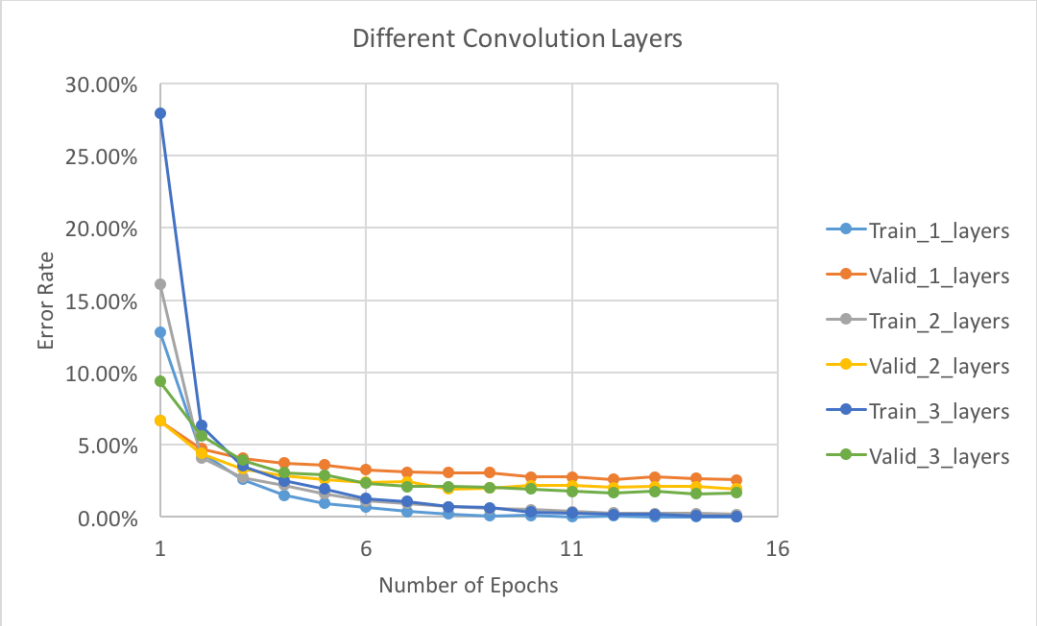

Figure 2: Different Convolution Layers

2.4.4 Optimizers

Figure 4 shows the result of optimizers. The conjugate gradient method converges faster for the training data than SGD does. However for validation set, it can decrease more but is not as stable as SGD's error curve. In general, the CG optimizer performs slightly better than SGD.

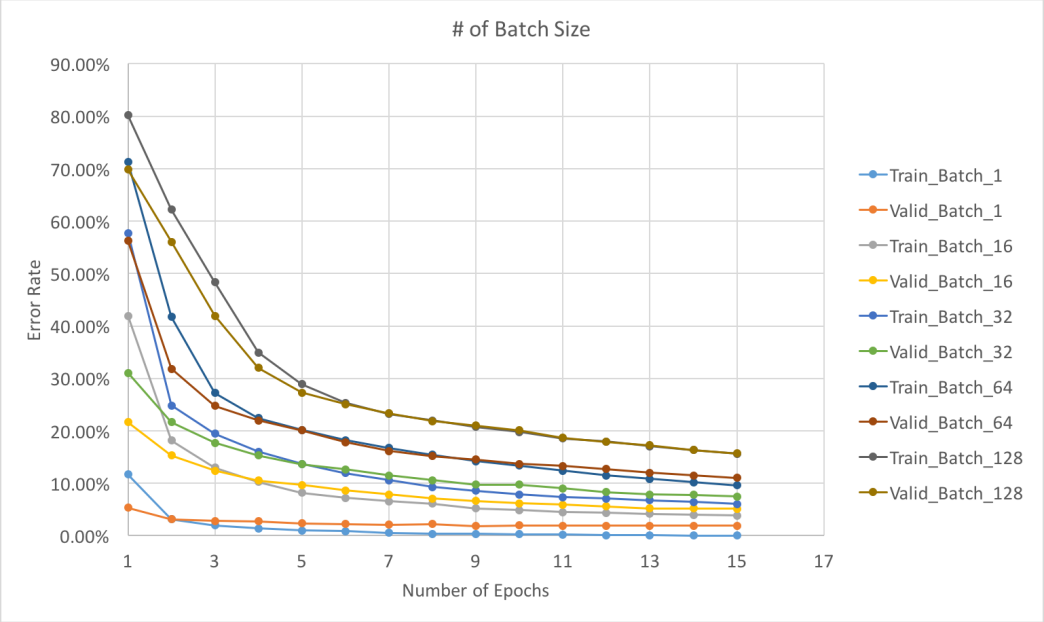

Figure 3: Different Batch Size

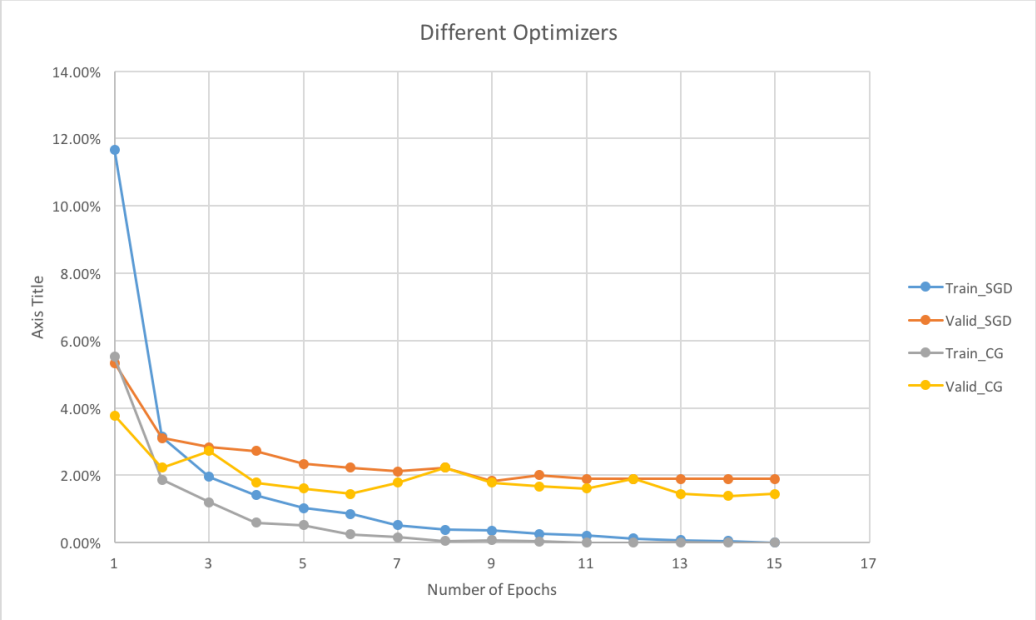

Figure 4: Different Optimizers

2.4.5 Non-linearities

We also investigated the influence of different activation function. We tried to replace the $tanh$ non-linearity by $ReLU$ and $Sigmoid$ as figure 5 shows. The result of $Sigmoid$ is worse than the other two, which takes a long time to converge to around 8% error. The other two are comparable. But $tanh$ got a more stable and slightly lower error rate.

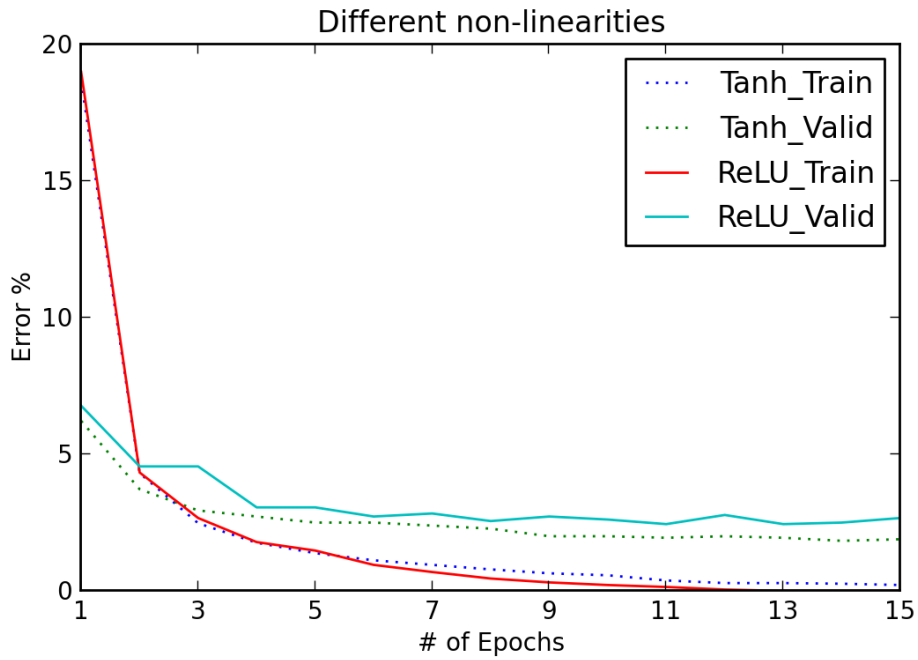

Figure 5: Different non-linearities

2.4.6 Different Pooling Sizes

We explored different pooling sizes. The default setting is $2 * 2 * 2 * 2$. In addition, we considered $3 * 3 * 3 * 3$ and $3 * 3 * 2 * 2$. The result is shown in figure 6 . The pooling size of 3 with a step 2 returns the best result.

2.4.7 Different Filter Sizes

We also explored different filter sizes, 3, 5 and 7. The result is shown in figure 7. Filter size 7 returns the lowest validation error.

2.4.8 Dropout

We also discussed dropout [2] as a way to avoid overfitting on training set and improve the generalization of the model. We added dropout (0.5) to the first CNN layer, the second CNN layer and the fully-connected layer respectively as well as all three layers. The results of dropout are shown in figure 8 and figure 9. As the results show, when adding dropout to all three layers, it will improve the generalization of the model.

2.5 Conclusion

The results display that increasing the layers and hidden units is generally useful. The pooling size, kernel size, optimizer, activation and so on, should be carefully chosen. Not every setting converges the model. On the other hand, some of them converge but tend to over-fit, so beware of using the validation set to anticipate the performance on testing data. Our final gets 99.73% accuracy on training set and 99.54% on testing set.

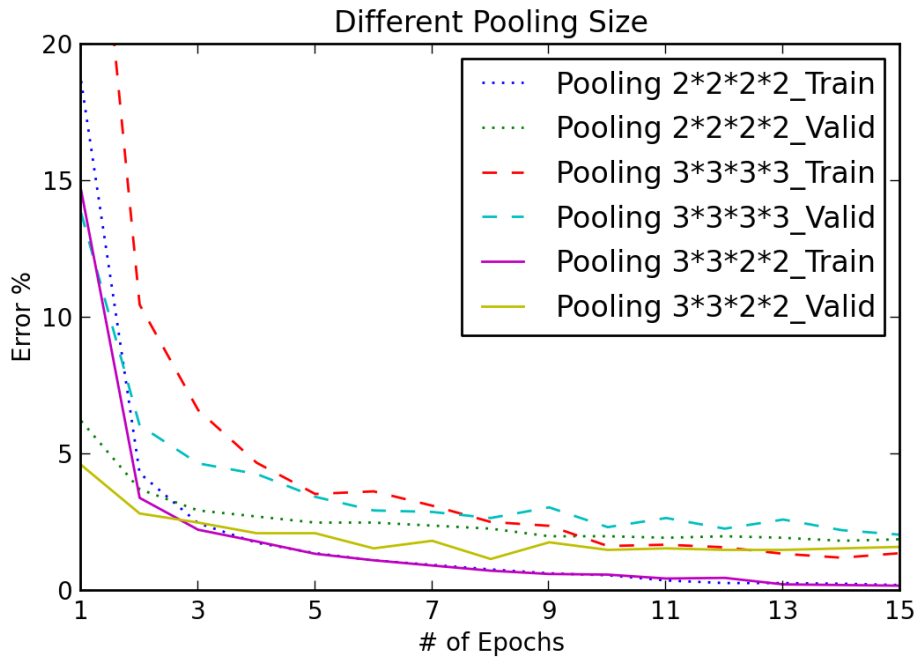

Figure 6: Different Pooling Sizes

References

[1] Pierre Sermanet, Sandhya Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3288–3291. IEEE, 2012.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

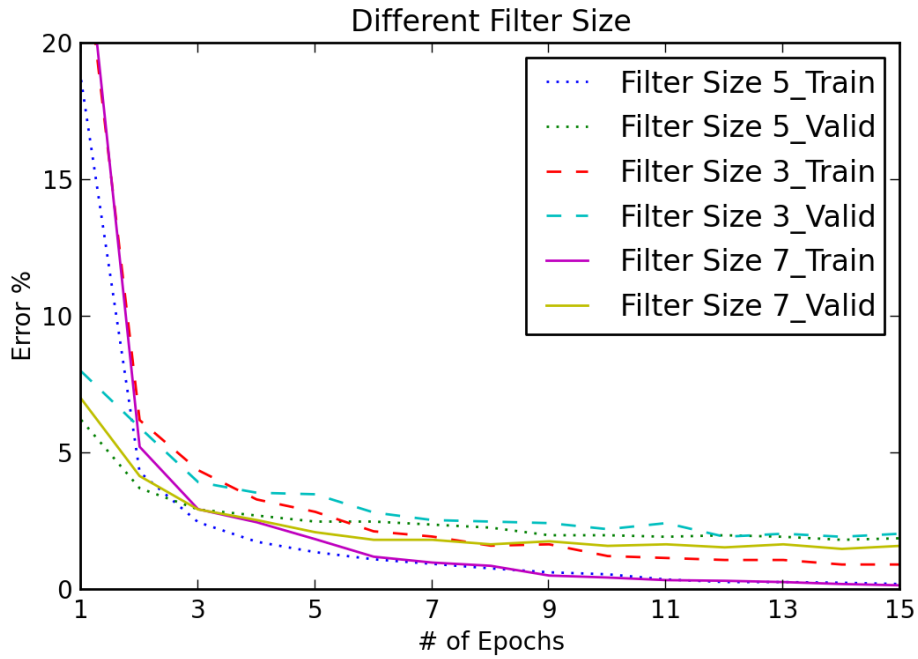

Figure 7: Different Filter Sizes

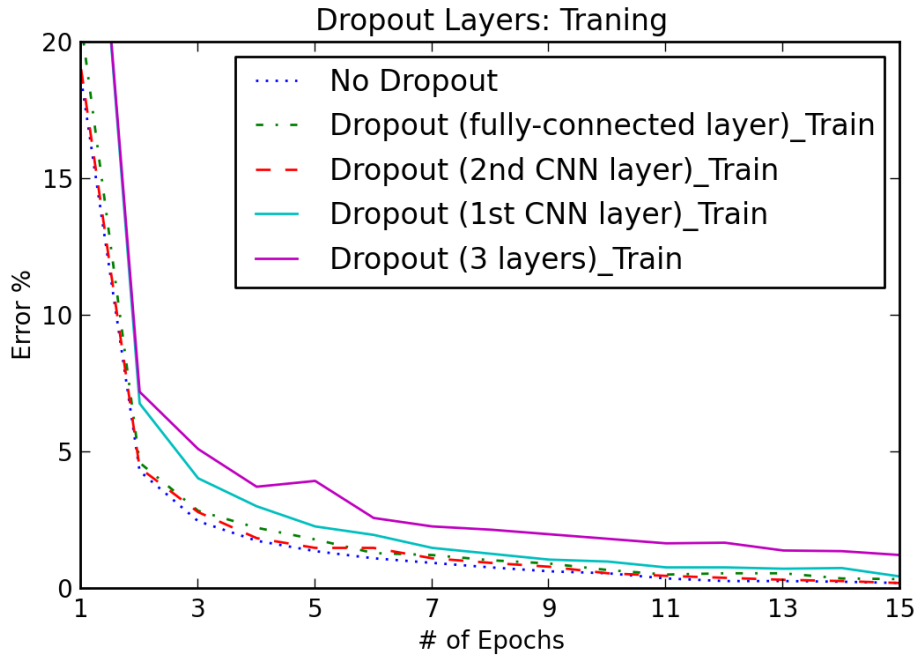

Figure 8: Dropout: Training

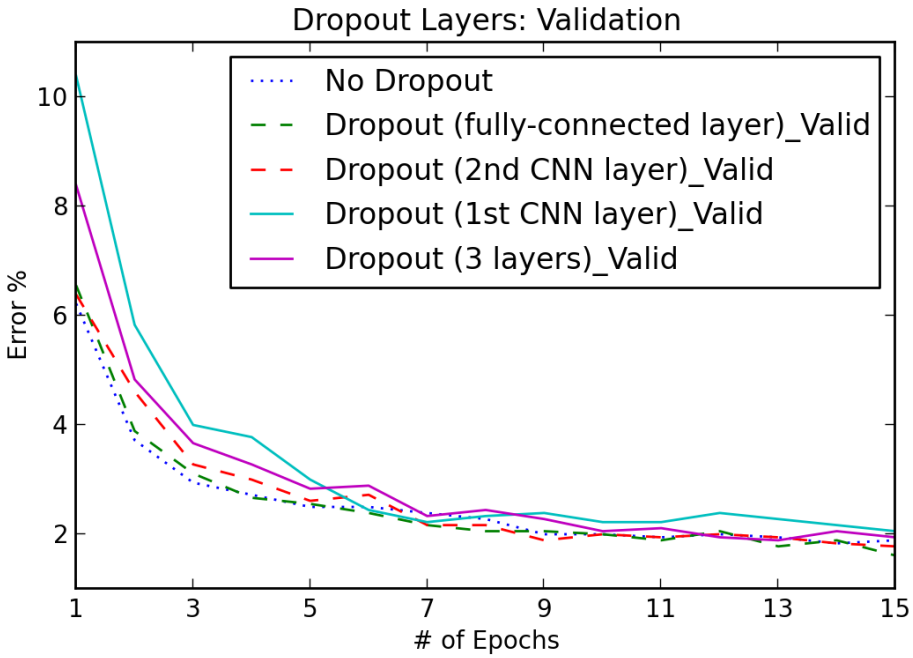

Figure 9: Dropout: Validation