Digit classification using Convolutional Neural Networks

Saurabh Mathur

November 9, 2016

Abstract

This project deals with the problem of handwritten digit classification.

The key idea is to show that carefully designed neural networks can perform complex tasks like image recognition without the need of data preprocessing or feature engineering.

The problem statement is as follows -

Given an image, identify the digit that it represents.

1 Introduction

Machine Learning is the application of statistical methods to make predictions about the future using data from the past. It works by finding patterns in data and using the same patterns to make predictions.

Supervised Learning is the most common form of machine learning. Since handwritten digit data was available, I was able to apply Supervised Learning to the problem. I've described the dataset and supervised learning in detail in sections 2 and 3 respectively.

I applied 3 approaches to solving this problem - softmax regression, multilayered neural network, and convolutional neural network.

- Softmax regression is the simplest kind of neural network and consists of 2 layers of neurons for input and output respectively. I've described its structure in detail in section 4.

- A multilayered neural network consists of hidden layers of neurons between the input and the output layers. The multilayered neural networks can theoretically estimate any continuous function. I've described their structure and working in section 5.

- A convolutional neural network is a specialized form of a multilayered neural network that works really well on images. I've described the structure and working along with some famous architectures in section 6.

For the training of these models, I used the ADAM Stochastic Gradient Descent algorithm which combines the advantages of AdaGrad and RMSProp [4].

Supervised Learning models become more likely to overfit the data as their complexity increases. In deep neural networks, conventional ways to tackle this problem fail as they are slow at large scale. Geoffrey E. Hinton et al. proposed a method called dropout to introduce regularization in deep neural networks [3]. I used dropout to prevent overfitting in the convolutional neural network. I've discussed it in detail in section 7.

2 The dataset

I used the Mixed National Institute of Standards and Technology standard benchmarking dataset (MNIST) for training digit classification models. It consists of 52,500 labeled black and white images of handwritten digits 0 to 9. Each 28×28 image is a one-hot encoded vector of length 784. Figure 1 shows some of the images from the dataset.

The current state of the art accuracy on this dataset is 99.77 %. Dan C. Ciresan, Ueli Meier and Jürgen Schmidhuber used a committee of 35 convolutional neural networks to achieve this result. The overall output was the average of the individual output of each of the networks[1]

Figure 1: MNIST - Some of the images, classwise

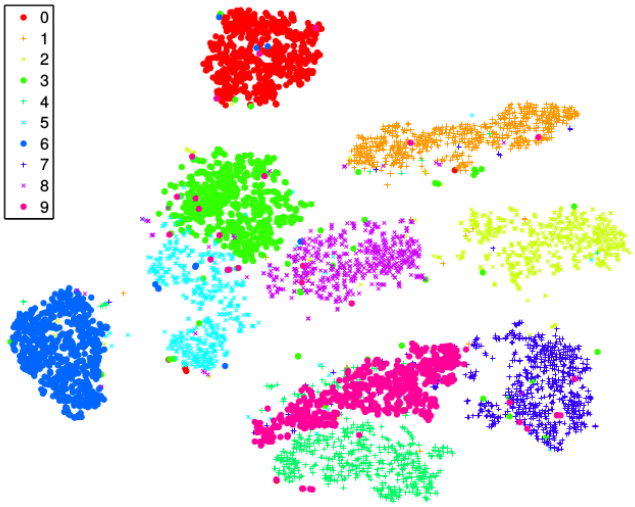

Figure 2: MNIST - Visualization of the data using t-SNE[8]

3 Supervised Learning

Supervised Learning consists of two phases - training and testing.

- In the training phase, we use a set of pre-labeled data to establish a mathematical relation between the data and labels.
- We use this mathematical relation, also called a model in the testing phase to predict labels for unlabeled data.

Most supervised learning algorithms perform poorly on raw data. We need to feed them pre-selected features which are highly correlated with the target variable. So, we need an extra step before training the model - feature selection.

In the feature selection step, we rely on domain experts and engineers to select and derive features from the original dataset.

4 Softmax Regression

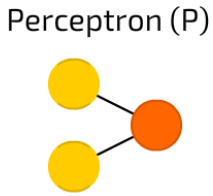

Figure 3: Softmax regression - structure [9]

The softmax regression or perceptron is the simplest kind of neural network. As shown in Figure 3, it consists of 2 layers - one for input and another of output . The model consists of a linear function composed with a softmax function -

$$\hat{y}(W, b; X) = softmax(W \cdot X + b)$$

The softmax function converts the output of the linear function to probabilities -

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{n} e^{z_i}}$$

The softmax regression model that I built for this project has $28 \times 28 = 784$ input neurons and 10 output neurons. I fed the data to the optimizer in 262 batches of 200 image-label pairs each for 25 epochs with a learning rate of 0.01. The accuracy on the dataset was 74.5 %

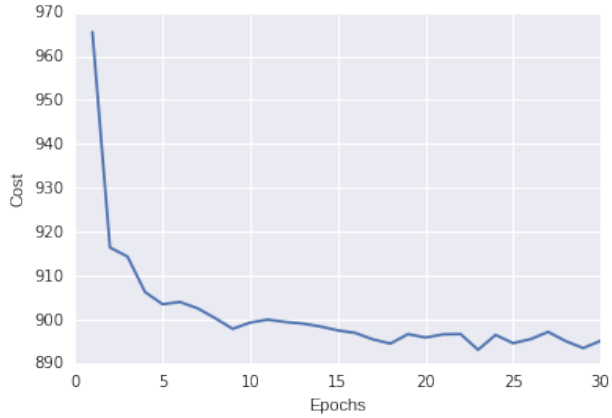

Figure 4: Softmax regression -Variation of training cost in each iteration

Figure 5: Softmax regression - Accuracy on training data and Validation data

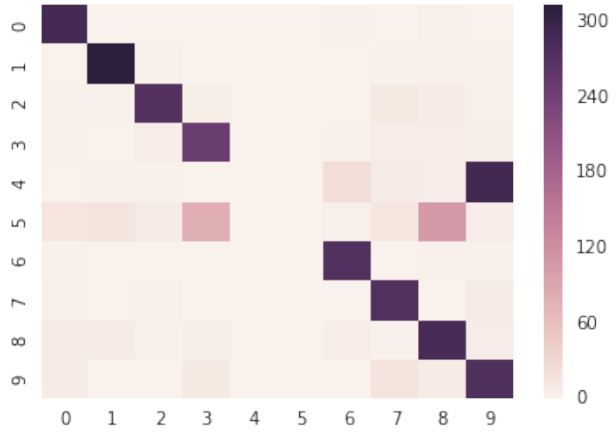

Figure 6: Softmax regression - Confusion matrix

5 Multilayer Neural Network

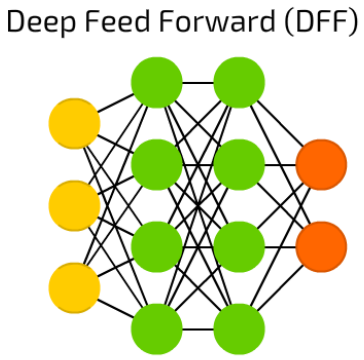

Figure 7: Multilayer Network - structure [9]

A Multilayer Feed Forward Neural Network consists of one or more hidden layers between the input and output layers as shown in figure 7.

Each layer of such a network uses a linear function followed by a non-linear activation function. Commonly used activation functions include softmax, hyperbolic tangent, the logistic sigmoid function and the Rectified Linear unit.

We can train Multilayer networks using the backpropagation algorithm. It consists of two passes over the network - a forward pass and a backward pass.

- Forward pass involves computing the output of each layer using the output of the previous layer as input and thus, finding the output of the network by applying a non-linear function on the last layer.
- Backward pass involves comparing the output of the network with the actual value or the ground truth. We propagated the error to each layer in the reverse order, updating weights at each unit. using the derivative of the error term at that unit with respect to the output of that unit.

Deep Neural Networks try to learn a better representation of the dataset, removing the need for manual feature selection discussed in section 3.

The Neural Network that I built for this project consists of 5 layers - two for input and output respectively, two hidden layers having 256 and 128 neurons respectively and a dropout layer between the hidden layers.

I fed the data to the optimizer in 262 batches of 200 image-label pairs each for 10 epochs with a learning rate of 0.001. The accuracy on the test dataset was 93.4 %.

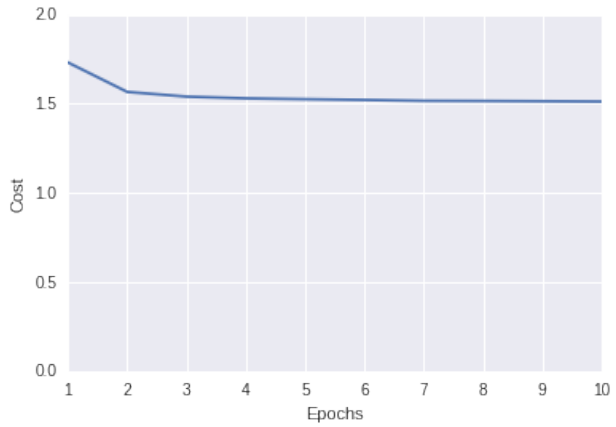

Figure 8: Multi-Layer Network -Variation of training cost in each iteration

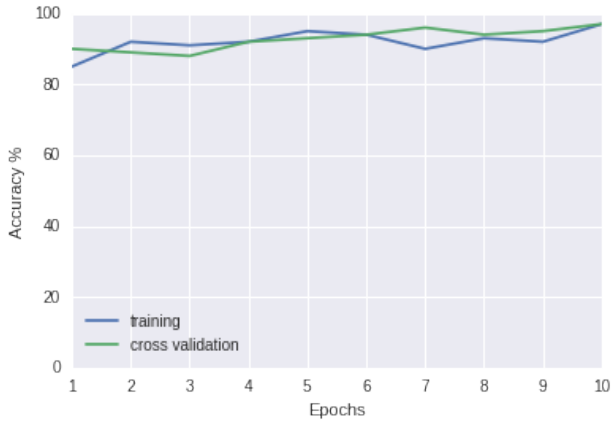

Figure 9: Multi-Layer Network - Accuracy on training data and Validation data

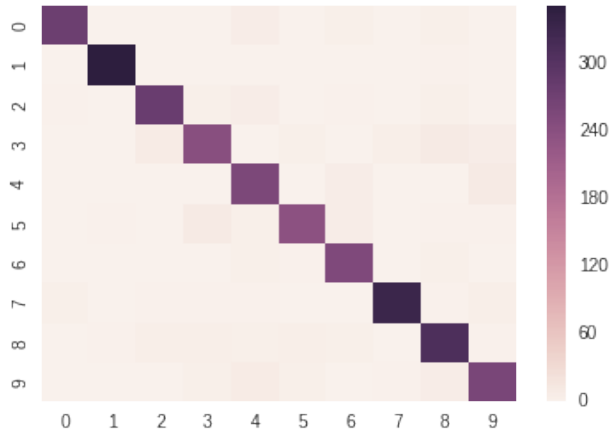

Figure 10: Multi-Layer Network - Confusion matrix

6 Convolutional Neural Network

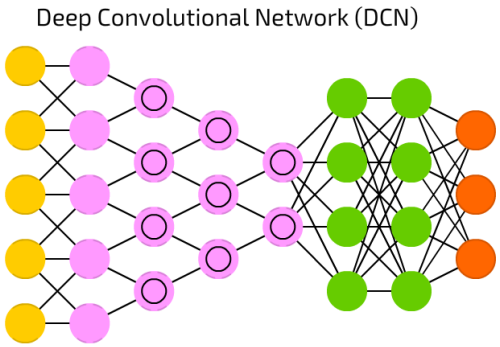

Figure 11: Convolutional Network - structure [9]

Just like Deep Neural Networks learn features automatically, Convolutional Neural Networks or ConvNets derive features from image data. They exploit the fact that in natural signals such as images, larger features are composed of smaller ones.

While ConvNets were used for image classification with great success as early as the 1990s [2], ConvNets were not as popular in the mainstream computer vision and machine learning communities until the ImageNet competition in 2012. Deep convolutional networks achieved spectacular results on a data set of about a million images from the web spread across 1,000 different classes, almost halving the error rates of the best-competing approaches[5, 6]. ConvNets are now the dominant approach for almost all recognition and detection tasks, even approaching human performance on some tasks.

Since then, the one ConvNet architecture that stands out from the rest is the GoogLeNet. It consists of stacked up inception modules. Each inception module has layers in parallel.[7]

An interesting demonstration by Kevin Xu et al. combines ConvNets with recurrent net modules for the generation of image captions[10].

According to Yann LeCun Yoshua Bengio and Geoffrey Hinton, there are four key ideas behind ConvNets: local connections, shared weights, pooling and the use of many layers [6].

- Convolution layers implement local connections.

- Max pooling layers implement Pooling.
- dropout implements shared weights.

In a Convolution Layer, units are arranged as feature maps. Each of the outputs of the previous layer is connected to patches in this feature map by a set of weights called a filter bank. All units in a layer share to same filter bank.

This allows detection of groups of patterns in images and makes their detection independent of location.

The role of the pooling layer is to merge similar features into one. The key idea of a max-pooling operation is to split the image into uniform patches and take the maximum value of each patch as representative of that patch.

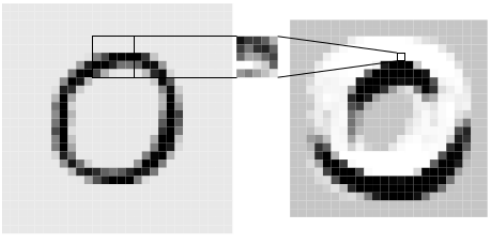

Figure 12: Input image (left), weight vector (center) and resulting feature map (right) [2]

The ConvNet that I built for this project consists of 4 layers - a convolutional layer followed by ReLu, a pooling layer, dropout, and a fully connected output layer.

I fed the data to the optimizer in 262 batches of 200 image-label pairs each, for 10 epochs with a learning rate of 0.001. The accuracy on the test dataset was 97.3 %.

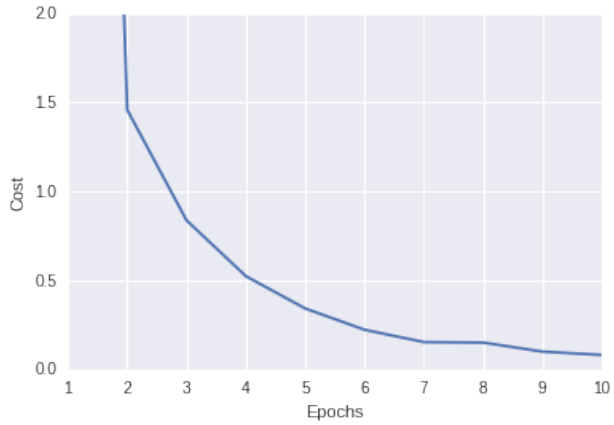

Figure 13: Convolutional Network -Variation of training cost in each iteration

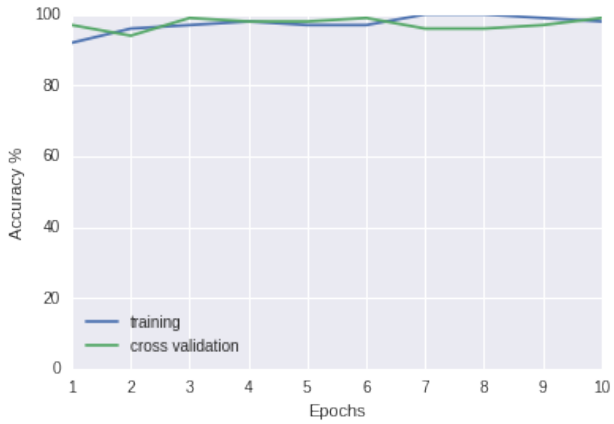

Figure 14: Convolutional Network - Accuracy on training data and Validation data

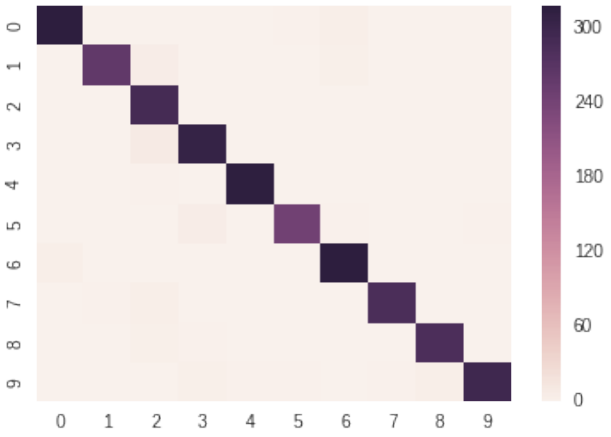

Figure 15: Convolutional Network - Confusion matrix

7 Regularization - Dropout

Deep neural networks overfit their training data. Hence, they have poor accuracy on testing data that was never seen during training.

Conventional methods to solve this problem, also know as regularization include weight penalty based methods (L1 and L2) and averaging over multiple models. They are slow and consume too much memory to be practical on a large scale. Also, we may not always have enough data to build multiple models to average over.

Geoffrey Hinton et al. proposed a simple yet effective way called dropout [3] to solve the problem of overfitting. The key idea is - when training a neural network, randomly drop-out units. Dropping out a unit here means temporarily hiding the unit along with its input and output connections ie. setting its output to 0.

They show that dropout is same as to averaging over 2^n thinner networks which share their weights. Thus, when making predictions, we can get a reasonable approximation of the actual average by multiplying the outputs of dropout layers with the probability of retaining them.

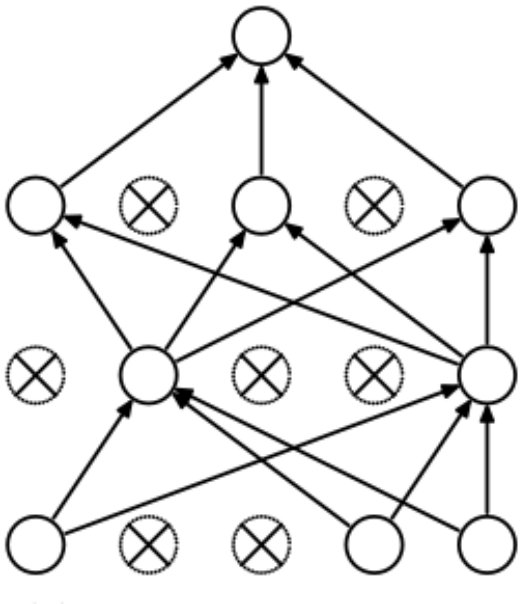

Figure 16: Training a Neural Network with dropout [3]

8 Results and Conclusions

I observed that the 5-layer neural network model consumes 10 times more time to train than the softmax regression model. Also, the Convolutional neural network consumes 6-7 times more time to train than the 5-layer neural network.

Training Time -

$$Softmax > MultilayerNetwork > ConvNet$$

The softmax model had a cross-validation accuracy of 74.5 % after training for 25 epochs. The 5-layer neural network model had a cross-validation accuracy of 93.4 % while the ConvNet had an accuracy of 97.3 % each after training for 10 epochs,

Accuracy -

$$ConvNet > MultilayerNetwork > Softmax$$

References

[1] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[2] Y. L. Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Advances in neural information processing systems 2. chapter Handwritten Digit Recognition with a Back-propagation Network, pages 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[6] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436444, 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[8] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9: 25792605, Nov 2008.

[9] F. V. Veen. The neural network zoo. http://www.asimovinstitute.org/neural-network-zoo/.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.