Introduction:

Description of the dataset:

The Street View House Numbers (SVHN) Dataset consists of images of house numbers in Google Street View images. Each image is classified as 1 of 10 digits from 0 to 9. There are 73257 digits for training and 26032 digits for testing. Although the original images with bounding boxes around individual characters is available, the images that will be used for this project are cropped images of single characters resized to a resolution of 32-by-32 pixels. The dimension of each images is 32X32X3.

Description of the deep learning network and training algorithm:

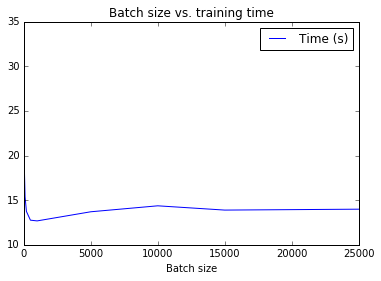
A convolutional neural network will consist of an initial convolutional layer with 20 5-by-5 kernals and a stride of 1 followed by a max pooling layer with a window size of 2-by-2 and stride of 2 followed by a rectified linear unit activation function. The second convolutional layer will consist of 50 5-by-5 kernals and a stride of 1 followed by a max pooling layer with a window size of 2-by-2 and a stride of 2 followed by a rectified linear unit activation function. Then there is a 25% 2-dimensional dropout layer. The output of these layers will be 5-by-5-by-50 array for each image, which will be flattened into a vector of length 1250. The next layer is a fully connected dense layer consisting of 500 neurons and the rectified linear unit activation function followed by a second fully connected layer of 10 neurons and the log softmax activation function.

Experimental Setup:

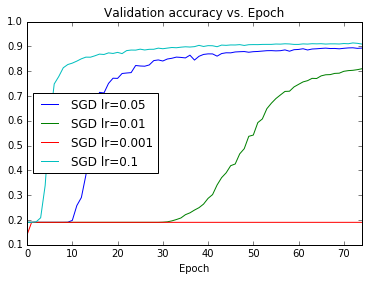
The framework that will be used to implement CNN networks is the Pytorch module in Python. Each network will be trained on the 73,257 training images. Then each trained model will be evaluated on the 26,032 test images. The primary metric will be overall accuracy and accuracy for each digit. Several optimizers will be used, such as stochastic gradient descent and Adam, in addition to several learning rates. 7000 samples from the training set will be reserved for validation and the remaining 66257 for training. Minibatches will be used to decrease the computation time. To determine the optimal number of mini-batches, the computation time for a few epochs based on different mini-batch sizes and 2 convoluation layers will be compared, and the value with the lowest computation time will eb selected. In order to determine the best optimization algorithm and learning rate, several possible ones will be implemented and the accuracy in the validation set will be compared over the number of epochs to determine which is best based on a network with 2 convolution layers. After selecting these parameters, overfitting will be detected by comparing the accuracy on the validation or test set to the accuracy on the training set. If the training accuracy is substantially higher, this indicates overfitting and that a model with fewer parameters is preferable.

Results:

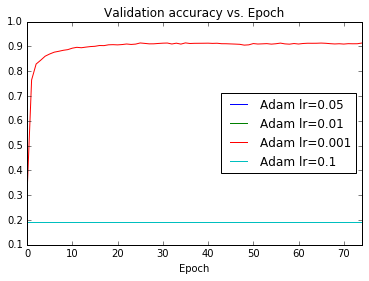
| *Batch size* | *Time (s)* |
| --- | --- |
| 16 | 32.1282 |
| 50 | 18.3828 |
| 100 | 15.3778 |
| 200 | 13.7388 |
| 500 | 12.7581 |
| 1000 | 12.6773 |
| 5000 | 13.6990 |
| 10000 | 14.3668 |
| 15000 | 13.8882 |
| 25000 | 13.9889 |



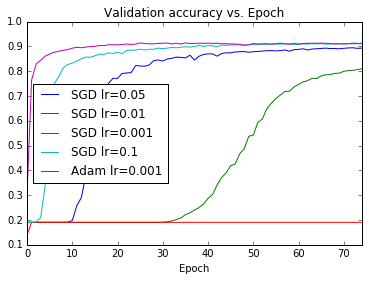
The computation time for various batch sizes is displayed in the above figure. The times are based on 2 epochs of training a convolution network with 2 convolution layers and 1 fully connected layer with 500 neurons. A mini-batch size of 1000 resulted in the smallest computation time, so that is the selected value.



The above figure plots the accuracy in the validation set for each epoch based on stochastic gradient descent optimization with various learning rates. The best learning rate is 0.1 which has the highest validation accuracy at every epoch. The next best is a learning rate of 0.05 followed by 0.01. The red curve is for a learning rate of 0.001 and there was no increase in accuracy unlike the other 3 curves.



The above figure plots the accuracy in the validation set for each epoch based on Adam optimization with various learning rates. The best learning rate is 0.001 which has the highest validation accuracy at every epoch. For all other learning rates there was no increase in accuracy.



The above figure plots the results of Adam with a learning rate of 0.001 with the results of stochastic gradient descent. Adam achieves a higher validation accuracy faster, but eventually SGD with a learning rate of 0.1 appears to converge. Based on these results, Adam with a learning rate of 0.001 is preferable.

The table below displays various measures for the convolutional neural network with 2 convolutional layers, a 2-dimensinal dropout layer, a fully connected dense layer consisting of 500 neurons and the rectified linear unit activation function followed by a second fully connected layer of 10 neurons and the log softmax activation function.

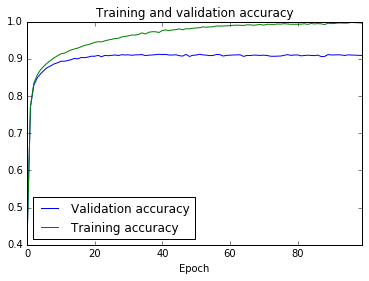
| *Digit* | *Precision* | *Accuracy* | *F1-score* | *Support* |
| --- | --- | --- | --- | --- |
| 0 | 0.89 | 0.91 | 0.90 | 1744 |
| 1 | 0.92 | 0.94 | 0.93 | 5099 |
| 2 | 0.94 | 0.92 | 0.93 | 4149 |
| 3 | 0.87 | 0.84 | 0.85 | 2882 |
| 4 | 0.88 | 0.93 | 0.90 | 2523 |
| 5 | 0.91 | 0.90 | 0.90 | 2384 |
| 6 | 0.88 | 0.88 | 0.88 | 1977 |
| 7 | 0.92 | 0.89 | 0.91 | 2019 |
| 8 | 0.90 | 0.83 | 0.86 | 1660 |
| 9 | 0.79 | 0.89 | 0.84 | 1595 |
| Avg/Total | 0.90 | 0.90 | 0.90 | 26032 |

The overall accuracy of the CNN model is 90%. Images of 1 were classified most accurately (94%) while images of 3 and 8 were classified least accurately at 84% and 83% respectively.

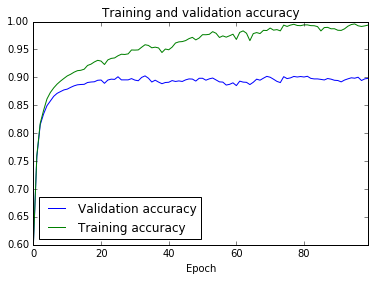
|  | *Predicted* | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Actual* | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* |
| 0 | 1590 | 22 | 10 | 19 | 9 | 7 | 38 | 5 | 11 | 33 |
| 1 | 43 | 4780 | 47 | 32 | 105 | 11 | 15 | 47 | 7 | 12 |
| 2 | 14 | 50 | 3823 | 70 | 58 | 15 | 17 | 53 | 21 | 28 |
| 3 | 18 | 95 | 43 | 2407 | 25 | 77 | 20 | 12 | 36 | 149 |
| 4 | 16 | 60 | 29 | 19 | 2334 | 6 | 10 | 15 | 11 | 23 |
| 5 | 7 | 15 | 19 | 84 | 25 | 2135 | 54 | 4 | 7 | 34 |
| 6 | 41 | 16 | 11 | 27 | 33 | 44 | 1747 | 5 | 31 | 22 |
| 7 | 5 | 118 | 41 | 31 | 16 | 3 | 5 | 1791 | 2 | 7 |
| 8 | 22 | 13 | 10 | 51 | 29 | 27 | 73 | 4 | 1373 | 58 |
| 9 | 28 | 18 | 40 | 21 | 19 | 18 | 11 | 3 | 20 | 1417 |

The cross-tabulation above shows how images of each digit were classified in the test set. The largest numbers are across the diagonal, indicating correct classification. For example, images of the number 3 were mostly classified as 3, but also misclassified as 9, 1, or 5, and the number 8 was misclassified as 9 or 3.

There is some evidence of overfitting in this model indicated by the plot below. The training accuracy increases to 1, but the validation accuracy increases to 0.9 and does not increase beyond that. Therefore, a model with fewer parameters might be preferable.



Removing the dropout layer results in a slight decrease in accuracy to 0.89. In addition, the training and validation accuracies fluctuate more as indicated by the plot below.



Summary and Conclusions

In conclusion a model with 2 convolution layers was trained and subsequently able to classify 90% of the test images. Performance might be improved if the number of kernals in each layer is modified, or the number of neurons in the fully connected layer is modified, or if additional layers are added or removed. In addition, the image data was not normalized prior to training the model, which might also affect the performance.

References

<http://ufldl.stanford.edu/housenumbers/>

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

<https://codetolight.wordpress.com/2017/11/30/getting-started-with-pytorch-for-deep-learning-part-3-5-pytorch-sequential/>

pytorch.org