Convolutional Neural Network for Street View House Numbers

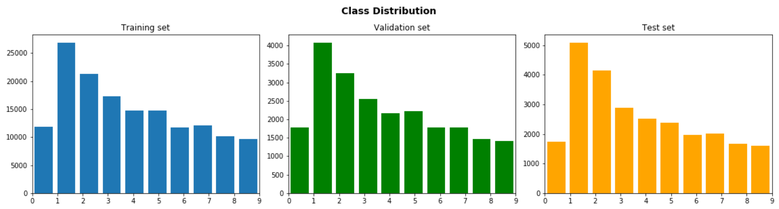
Muthu Chidambaram & Melinda Ding

Problem Statement

The goal of our project was to correctly detect a series of numbers in a given image of house numbers by training a convolutional neural network with multiple layers.

Data Collection

- The data set we are using was obtained from Google Street View images and was preprocessed and labelled by a research group at Stanford [3].
- The images are centered around a single character but still contain distractors at the sides (such as other digits).
- All digits have been resized to 32x32 pixels.
- The data comes labelled with 10 classes, 1 for each digit. We split the data into 150733 samples for training, 22524 samples for validation, and 26032 images for testing.
- The distribution of the classes in each set is shown below:



Procedure & Methods

We first converted all the images to grayscale because color doesn't matter for this problem, and to decrease compute time. This was done using skimage's rgb2gray function. We then normalized all the images.

We then defining the following simple network (from Sharma's notebook[2]) as our base model:

- Convolutional layer: 32 (5x5) filters, relu activation
- Max-pool: (2x2) pooling size, stride=2
- Convolutional layer: 64 (5x5) filters, relu activation
- Fully-connected: 1000 nodes, relu activation
- Fully-connected: 10 nodes, softmax activation
- Adam optimizer, batch size of 128, and 10 epochs

This model achieved ~90% accuracy.

We then researched and experimented with different ways to improve our model including:

- Adding more data
- Transfer learning with the VGG16 network
- Varying the depths and density of the convolutional layers[1]
- Varying filter size and stride
- Different loss optimizers (SGD) and learning rates

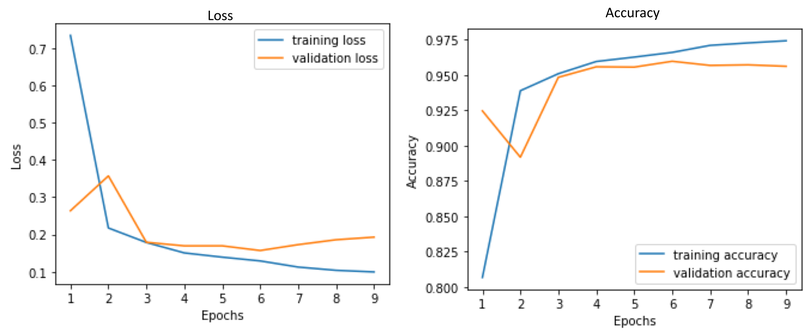
Model

Architecture

Layer	Filter/Pooling Window Size	Activation	Stride	Depth	Number of Parameters
Input					
Convolutional	5x5	Relu	1x1	48	1248
Max-pool	2x2		2x2	48	
Convolutional*	5x5	Relu	1x1	128	153728
Max-pool	2x2		1x1	128	
Convolutional*	5x5	Relu	1x1	256	819456
Max-pool	2x2		1x1	256	
Convolutional*	5x5	Relu	1x1	256	1638656
Max-pool	2x2		2x2	256	
Fully-connected		Relu		1024	9438208
Dropout		Probability = .50			
Fully-connected		Relu		512	524800
Fully-connected		Softmax		10	5130

- Convolutional* = convolutional layer followed by a batch normalization and a dropout layer with probability 0.1
- Total number of parameters: 9,700,202
- Total number of trainable parameters: 9,698,922
- Batch size of 128
- Max 25 epochs with early stopping and patience of 3

Evaluation



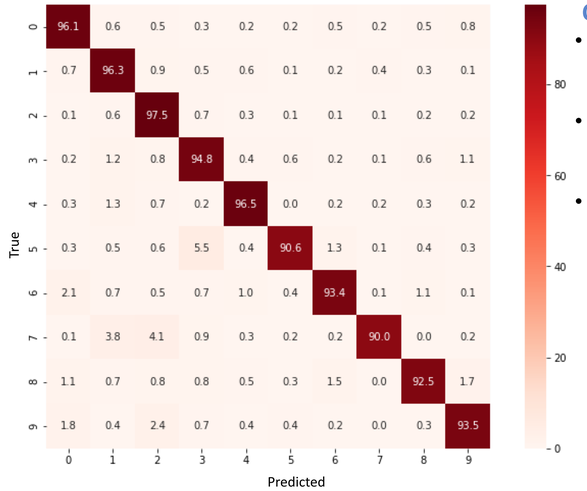
Loss
From the loss graph, we see that the model is possibly slightly overfit to the data; the validation loss curve comes down to touch the training loss curve, and then trends back up and away.

Both
Both graphs describe the performance of the model over time. The training and validation loss curves converge around 3 epochs and remain pretty tight (meaning good performance) until 6 epochs.

Accuracy
The accuracy graph further illustrates the overfitting present. Right after 6 epochs, the validation accuracy begins to decline while the training accuracy continues to improve.

Results

Test Loss	0.21379435018403886
Test Accuracy	0.9473724646588814



Confusion Matrix

- The model was best at predicting 2 (96.3% accuracy).
- The model was worst at predicting 7 (90% accuracy).
- Delineating between 7 and 1 was the most difficult task for the model, with a bias toward picking 1. This is probably because there were more 1s than any other number in the dataset.

Conclusion

Findings

- We were able to increase test accuracy by ~4%.
- Shallower networks worked better for the smaller size of our dataset.
- Denser networks with 2 fully connected layers at the end worked best.
- Adam optimizer worked better than the SGD optimizer.
- Adding more data helped improve accuracy the most. This is because the original training dataset had only 70,000 images. We added 100,000 more images from the extra dataset.

Future Work

- Implement sliding detection to read in and label non centered images. This functionality would also extend to probabilistic number detection in arbitrary images.
- Further tuning the model to prevent overfitting via adjustment of batch normalization, dropout rates, and learning rates.
- Create variations (such as rotations, stretches, salt & pepper flakes) in the training data set to increase size and difficulty.
- Apply basis transformations on the images to increase the number of channels in the image.

References

[1] Goodfellow, Ian J. et al. (2013). "Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks". In: arXiv:1312.6082 [cs]. arXiv: 1312.6082. url: http://arxiv.org/abs/1312.6082.
[2] Sharma, Aditya, SVHN-CNN, March 2018, https://github.com/aditya9211/SVHN-CNN
[3] 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. (http://ufldl.stanford.edu/housenumbers/)