Deep Learning Capstone Project
Build a Computer program to decode sequences of digits from natural images
P.S. Aravind

Project Overview:
Objective of the Capstone project is to build a computer program that can interpret number strings in real-world images. The computer trained model would decode sequences of digits from natural images, and print the number it sees in the image. Data for training the computer model would be Street View House Number (SVHN) - a large-scale dataset of house numbers in Google Street View images.

Problem Statement:
Recognizing sequence of digits in natural photographs is a hard problem. This project addresses this problem by implementing a solution for recognizing arbitrary multi-digit numbers from Street View imagery. Project implements a unified approach of localization, segmentation and recognition and integrates these 3 steps by using a deep convolutional neural network that operates directly on the image pixels. By employing various architectures of neural network, best performance is achieved by very deep layers. This approach has a **94.6%** accuracy in recognizing complete street numbers on publicly available SVHN dataset.

Strategy for solving the problem would be to train a model by reading a huge converted data from SVHN dataset. Raw SVHN dataset consists of both data and corresponding label, this raw data was converted into training and validation dataset to determine the predicted accuracy. Several optimization and deep neural network architecture was designed to decrease overfitting and obtain accuracy of over 90%.

Metrics:
Street number transcription is a special kind of sequence recognition problem, ie. given an image, the task is to identify the number in the image. The number to be identified is a sequence of digits $s = s_1, s_2, .. s_n$. When determining the accuracy of a digit transcriber, we compute the proportion of the input images for which the length n of the sequence and every element s_i of the sequence is predicted correctly. The system returns a confidence value in the form of probability of the most likely prediction being correct. For this project the sequences are of bounded length of five digits.

Data Exploration and Visualization:
The SVHN dataset is a dataset of 235K street numbers, along with bounding boxes for individual digits, giving about 600K digits total. The dataset was processed in the following way - the process starts by finding the small rectangular bounding box that will contain individual character bounding boxes. Then this bounding box is expanded by 30% in both the x and y direction, then the image is cropped to that bounding box and the image is resized by cropping it to 32 x 32 pixels. Also the image was grayscaled from 3 channels of color pixels to single channel to reduce the size of the image stored in memory.

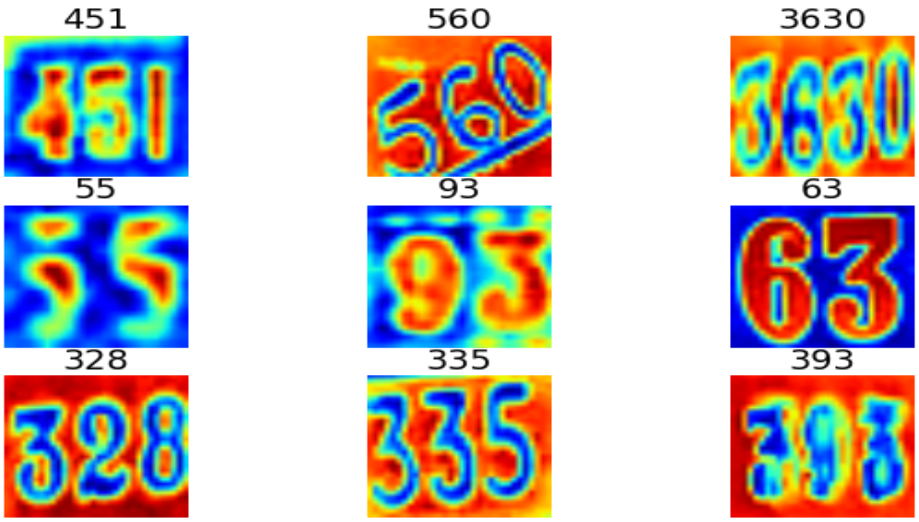
The images were standardized using scikit learn preprocessing function StandardScaler, this function standardizes the image by removing the mean and scaling to unit variance. Standardization of dataset is a common requirement for deep learning estimators, the model

might behave badly if the individual feature do not more or less look like standard normally distributed data.

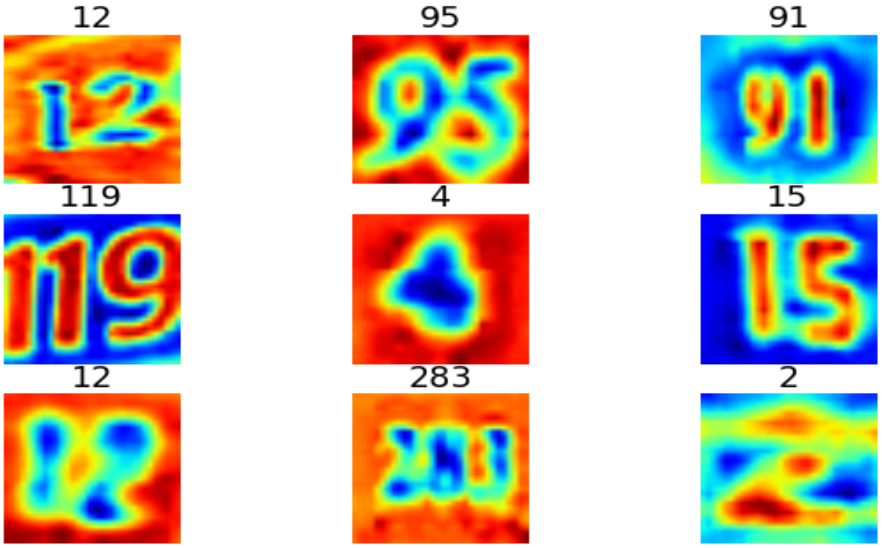
Grayscalling the image enabled processing 188K images for training, 47K images for validation and 13K images for testing in a home computer having 12Gb memory.

Following image samples show differing number of characters in the image with considerable scale variability, also digits in the images have several shifted versions for each training example.

Image samples for **Testing:**



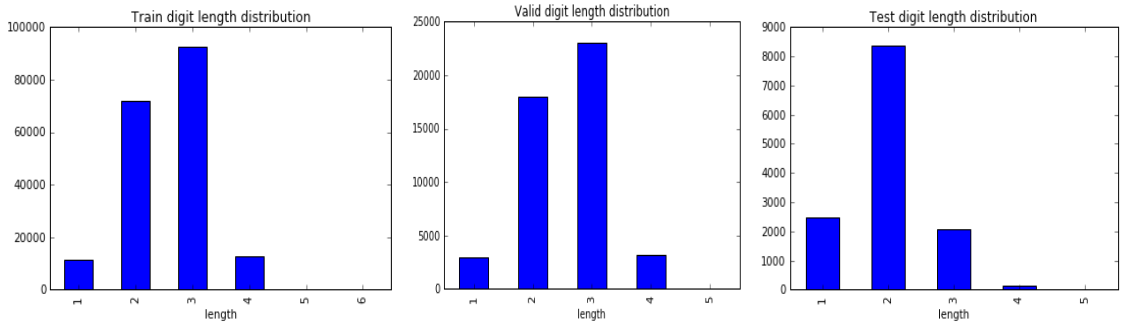
Testing:



Following table shows the distributions of digit lengths for datasets, data shows that most street view numbers have 2 or 3 digits and the number of digits drops considerably after 3 digits and

there are no house numbers with 6 digits in the test data. so our assumption to restrict our model to read only 5 digits would be supported by the test dataset digit length.

Length	Training count	Validation count	Test count
1	11,601	2,921	2,483
2	71,881	17,975	8,356
3	92,469	23,011	2,081
4	12,548	3,224	146
5	104	20	2
6	1	0	0
Total	**188,604**	**47,151**	**13,068**



Algorithms and Techniques:
This project follows the steps detailed in the published baseline model by using a deep convolutional neural network that operates directly on the image pixels. The model used in this project is configured with multiple hidden layers, best configuration for the project had 14 layers, having a deeper architectures resulted in obtaining better accuracy.

As detailed in the above cited paper about model architecture, convolutional neural networks are neural network with sets of neurons having tied parameters. Like most neural networks, they contain several filtering layers with each layer applying an affine transformation to the vector input followed by an element wise non-linearity. In the case of convolutional networks, the affine transformation can be implemented as a discrete convolution rather than a fully general matrix multiplication. This makes convolutional networks computationally efficient, allowing them to scale to large images. If the image is shifted by one pixel to the right, then the output of the convolution is also shifted one pixel to the right; the two representations vary equally with translation.

Image-based convolutional networks uses a pooling layer which summarizes the activations of many adjacent filters with a single response. Such pooling layers summarizes the activations of groups of units with a function such as their maximum, mean, or L2 norm. These pooling layers help the network be robust to small translations of the input.

Data Preprocessing:

Following preprocessing steps were performed on the input datasets.

1. Parse the input matlab file and extract the image, label and bounding boxes for each digit.
2. Crop the original images to maximum bounding box for all the digits and providing a buffer of 30% on height and width.
3. Grayscale the image and crop the image to 32 x 32 pixel, this reduces the 3 channel image to single channel.
4. Apply standardization using scikit learn preprocessing function.
5. Convert the labels to length of the digits along with 5 separate digits each having 11 outputs, 0 - 9 and 10 for blank.
6. Create testing and validation datasets using sklearn.
7. Split and compress the dataset to be used for modelling.

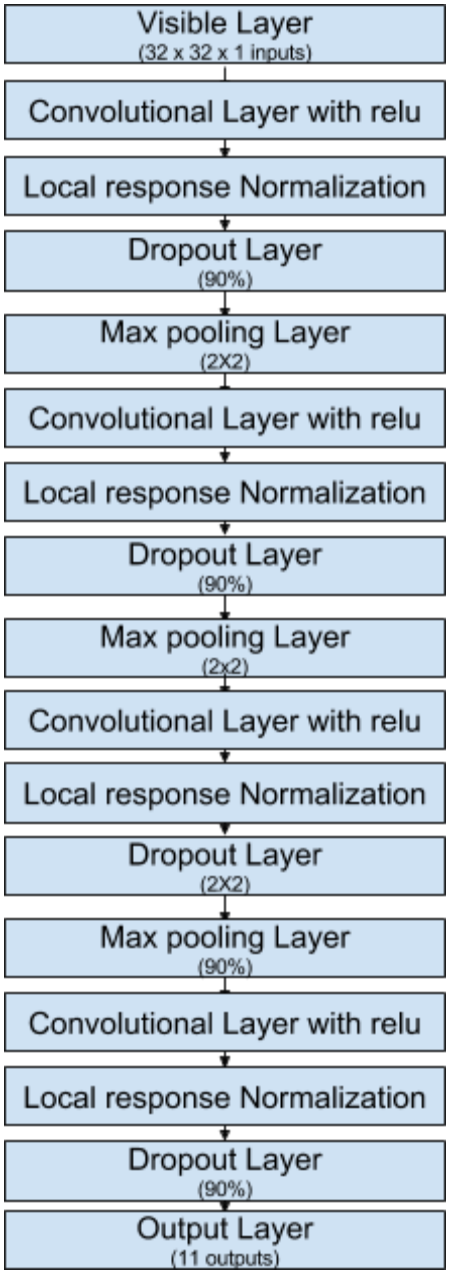
Implementation:

Below summarizes the network architecture.

1. The visible layer is a two dimensional pixels with the dimensions [32 x 32 x 1], the channel value is gray scaled.
2. The first hidden layer is a convolutional layer that performs the following:
    a. Flattens the filter to a 2-D matrix with shape [5 * 5 * 1, 16]
    b. Extracts image patches from the input tensor to form a virtual tensor of shape [256, 28, 28, 5 * 5 * 1]
    c. For each patch, right-multiplies the filter matrix and the image patch vector
    d. rectifier activation function is used.
3. Next local response normalization is applied, which converts a 4-D input tensor treated as a 3-D array of 1-D vectors, and each vector is normalized independently Within a given vector, each component is divided by the weighted, squared sum of inputs within radius of 5 pixels.
4. The next layer is a regularization layer using dropout, it is configured to randomly include 90% of neurons in the layer in order to reduce overfitting.
5. Next max pooling is performed on the input, with stride of one and are zero padded so that the output is the same size as the input. Pooling takes the maximum value over 2x2 blocks.
6. Then the layer is repeated with convolutional layer with rectifier activation function.
    a. Flattens the filter to a 2-D matrix with shape [5 * 5 * 16, 32]
    b. Extracts image patches from the input tensor to form a virtual tensor of shape [256, 24, 24, 5 * 5 * 16]
7. Local response normalization
8. Dropout set to 90%.
9. Maxpool layer with the size of 2 x 2.
10. Convolutional layer with rectifier activation function.
    a. Flattens the filter to a 2-D matrix with shape [5 * 5 * 32, 64]

    b. Extracts image patches from the input tensor to form a virtual tensor of shape [256, 20, 20, 5 * 5 * 32]
11. Local response normalization
12. Dropout set to 90%
13. Maxpool layer with the size of 2 x 2.
14. Convolutional layer with rectifier activation function.
    a. Flattens the filter to a 2-D matrix with shape [5 * 5 * 64, 128]
    b. Extracts image patches from the input tensor to form a virtual tensor of shape [256, 24, 24, 5 * 5 * 64]
15. Local response normalization
16. Dropout set to 90%
17. Finally, the output has 11 neurons for the [0 - 9] classes and 1 blank, a softmax activation function to output probability-like predictions for each class.

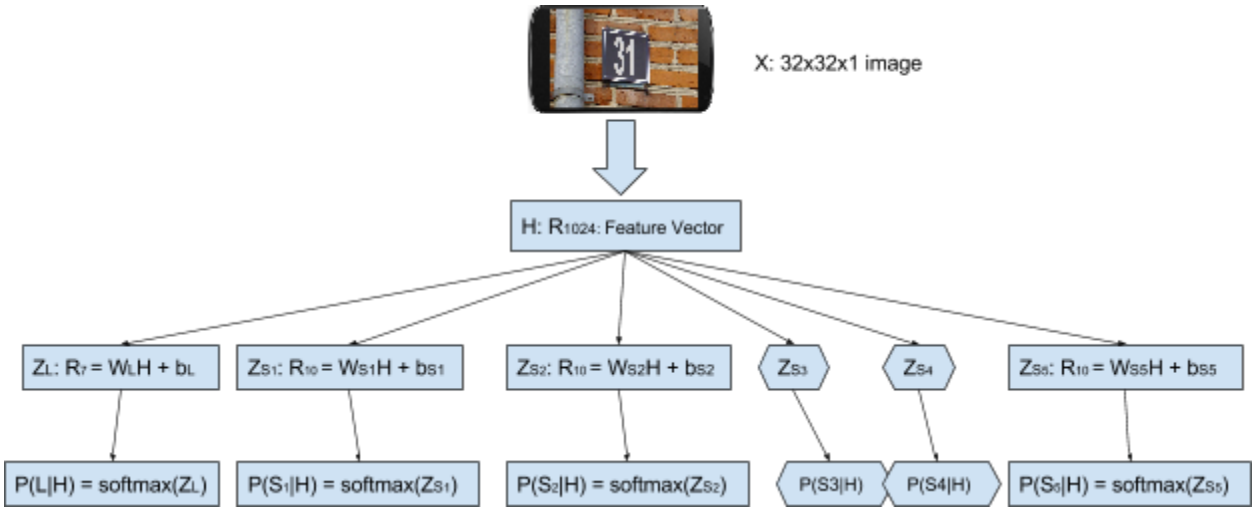
A depiction of the network structure is provided below.



Implementation:

Below diagram shows the computational graph used to transcribe the house numbers. The diagram shows how the parameters of P(S|X), where X is the input image and S is the sequence of numbers depicted by the image. Goal of the project is to learn a model of P(S|X) by maximizing log P(S|X) on the training set. First the set of features H from X is extracted using a convolutional network with a fully connected final layer. The networks learns its own means of representing spatial locations in H for the 5 digits along with the length of digits. Six separate softmax classifiers are then connected to this feature vector H, i.e., each softmax classifier forms a response by making an affine transformation of H and normalizing this response with the softmax function. First classifier provides the distribution over the sequence

length P(L|H), while the others provide the distribution over each of the members of the sequence, $P(S_1|H)$, …, $P(S_5|H)$.



To train the model, P(S|X) has to be maximized on the training set using stochastic gradient descent. Each of the softmax models for L and each S_i is used as a digit classifier softmax model for which that digit is not present.

Refinement:
After experimenting with various deep neural network layers, final deep layer as detailed above was designed with various layers using Tensor Flow. Following parameters namely: number of steps, initial learning rate, batch size and dropout rate(keep_prob) were implemented. This enabled passing different parameters to evaluate which parameters provided best result.

1. **Batch size**: Various batch size from 32 to 512 was experimented.
2. **Initial learning rate**: rate from .001 to .1 was experimented for exponential decay to the learning rate that was applied to Adagrad optimizer.
3. **keep_prob**: probability that each element is kept in that layer.
4. **Number of steps**: Number of iterations over the training set.

Batch Size	keep_prob	starter_learning_rate	steps	Test Accuracy
64	.5	.001	5000	**55.8%**
64	.5	.05	5000	**56.1%**
64	.85	.05	5000	**90.0%**
32	.90	.05	5000	**88.1%**
64	.90	.05	5000	**90.5%**
128	.90	.05	5000	**55.9%**
256	.90	.05	5000	**92.0%**
512	.90	.05	5000	**91.7%**

256	**.90**	**.05**	**50000**	**94.6%**

Model Evaluation and Validation:

The model was evaluated by plotting the loss for the training set. Below chart from tensor board shows the loss for each parameter set. As the graph shows the optimal parameters that provided the lowest loss is for Batch size:256, keep_prob: .9, starting learning rate:.05. For this set of parameter the model was run for 50,000 iterations and the test accuracy as 94.6%.

Justification:

According to the published baseline model the best model obtained a sequence transcription accuracy of 96.03%. After several iterations and experimentation with different layers, batch size, and starter learning rate, the best I was able to achieve was 94.6%. The published paper states that the performance of human operators is around 98% accuracy, so comparing to this, the accuracy of 94.6% is close enough but improvement could certainly be made and the model could be fine tuned and run on a machine that utilizes GPU's.

Conclusion

Free-Form Visualization:
After the the final model was evaluated, the model was tested on a random set of images from the testing dataset. Following image shows the predicted label for the random image along with the actual label. Results from the model shows that deep neural network is able to recognize multi-digit numbers from natural photographs.

Above result shows that for this sample of 9 images, 8 images was predicted correctly.

Reflection:
Neural networks have shown that it could learn to perform complicated tasks of recognizing ordered sequence of digits. This method could be applied to other problems, such as general text transcription or speech recognition.
After reading the “Reading Digits in Natural Images with Unsupervised Feature Learning”, I was trying to follow the steps detailed in the paper, along with processing the image with 3 channels and image size of 54 pixel, but after spending over $125 on G2 instance in AWS, I shifted back to using my Home Computer since I could experiment with different parameters. I had to restrict the image size to 32 pixels and gray scale the image due to memory restrictions. I also tried to follow the architecture they discussed in the paper, eight convolutional hidden layers, one locally connected hidden layer and two densely connected hidden layers, I was not able to design this architecture, I feel I could have gotten accuracy of 96.03% if I was able to devise an architecture similar to theirs. The paper had a team of 5 people working on to device a Neural Network, this project would have been more easier and less time consuming if I was able to work as a team with a group of fellow students, MOOC does not provide this type of environment to find like minded students to work on a project.

Following image shows the predicted values for randomly selected house digits from the internet, results show good performance, but I had to eliminate couple of images since the prediction was completely off the real value. Next step would be to port this model to a Android or IPhone app to recognize numbers from real time picture, since devising a model took a long time, converting the model as an app would be taken up when I have time.

Improvement:

Project could be improved in following ways:

1. Recreate the image processing and architecture as detailed in the original paper, this involves creating 54 x 54 pixel image from 64 x 64 pixel image along with 3 channels instead of gray scale image.
2. Port the model as a Mobile application to showcase how Neural Network model could be applied in real world pictures.

Reference:

1. Udacity forum:
    a. https://discussions.udacity.com/t/how-to-predict-bounding-boxes/188727
    b. https://discussions.udacity.com/t/image-recognition/182983
    c. https://discussions.udacity.com/t/svhn-capstone-for-deep-learning/188590
    d. https://discussions.udacity.com/t/deep-learning-capstone-project-environment-limitations/175871
    e. https://discussions.udacity.com/t/anyone-working-on-deep-learning-capstone-project/178422
    f. https://discussions.udacity.com/t/how-to-deal-with-mat-files/160657
    g. https://discussions.udacity.com/t/what-loss-function-to-use-for-multi-digit-svhn-training/176897

2. Tensorflow tutorials
3. SVHN Documentation
4. Reading Digits in Natural Images with Unsupervised Feature Learning