Multi Digit Recognition - Convolution Neural networks

Neural networks defines a computational approach, where large collections of neural units are structured to model the computational structure of brain as understood by humans. The neural nets has been in literature since 1940s, recent advances in computing power bring them to forefront. Computer vision is one of fields which has gained tremendously from the neural networks. Digit recognition from handwritten or scanned document has been extensively studied & various solutions exist based on custom feature recognition for any particular digit. Detecting the digits from real world images is a difficult problem as digits can have various scale, rotational or spatial transformations.

Convolution Neural networks (CNN) is extension of neural networks as it adds spatial transformation layer to the neural networks. It roughly models the human visual cortex, where set of neurons respond to restricted region of space. This results in model which is invariant to translation, scale & rotation of the input image.

Problem Statement:

In this project I will train a Convolution neural network in tensorflow to classify the digits in the natural form, I plan to use the public street view house number dataset available at http://ufldl.stanford.edu/housenumbers/ [2]. The model I develops build on the one shared by google team [Goodfellow] .

Tasks involved in the project includes:

- Grab the public dataset & generate the labeled dataset both for individual numbers & sequence of numbers.
- I first plan to repeat the task on mnist type handwritten numbers from public data. Generate the synthetic dataset for sequence of digits using the mnist dataset. This dataset is easy to learn by classifier as it has less extent of various scale, noise & spatial transformations.
- Develop the tensor-flow model & experiment hyper-parameters of the model with synthetic mnist database.
- Use the tensor-flow model from above task, with further enhancements if needed.

Metrics

One metric of interest would be the number of characters classified correctly in the given test set, which would also be used as measure of performance of the individual digit classifier.

Other metrics of evaluation would be the number of digit sequences classified correctly, given as ratio of digit sequences classified correctly to total number digit sequences in the dataset.

Data Exploration & Generation

Mnist data ::

Mnist data is public database of handwritten digits available at(http://yann.lecun.com/exdb/mnist/). I generate a synthetic database of sequence of digits with length ranging from 1-5 . Each mnist digit is 28x28 pixels, 5 digit sequence is generated by concatenating five of these digits to generate image of size 28x140 pixels. The blank space in any digit sequence is represented by uniform distribution of 28x28 centered at 0 with label of -1.

Distribution of digits used at each of the five locations is balanced as shown in fig 2. We have uniform distribution of numbers at each digit location.

Fig 1 : Digit sequence generated from mnist database

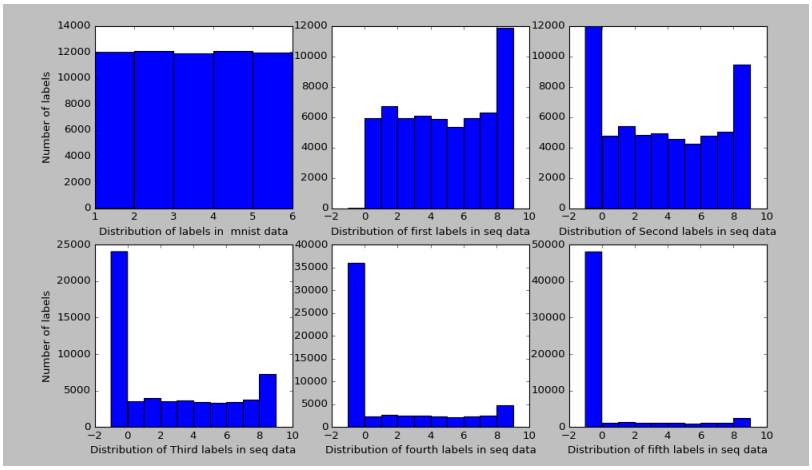

Fig 2 : Distribution of digits in the sequence

SVHN data:

SVHN is a real-world image dataset, it's more complex as compared to mnist data. It provides real world capture of street view house numbers. with bounding box marked around each digit. I generate two sets of data one for individual digits & other sequence of digits. Individual digits are from cropped data provided in SVHN cropped database. Sequence of digits is cropped from SVHN data taking the smallest bounding box enclosing all the digits in the image & extrapolating by 10%. The pixel are further normalized & cropped with bounding box of 32x32 pixels.

Fig 4- SVHN digit sequence database

Fig 5 – SVHN sequence of digits database

Fig 6 – Smallest bounding box sequence of digits

Fig 7 – Cropped sequence scale to 64x64

Label data for training is also prepared from SVHN labels, with -1 used to signify empty digit. Each sequence of digits is encoded as 6 digit vector. Zeroth vector element specify the length of the digit sequence with subsequent giving value of actual digits.

For example, Sequence of 3 digits 721 is encoded as label as vector of length 6 as follow [3,7,2,1,-1]. The CNN is expected to learn these six members of label vector, first being the length & subsequent digits. As further explained in the following methodology, special digit of -1 is used to stop backward propagation of error for the empty digits in the sequence.

Algorithms

The model is developed based on Convolution Neural networks. CNN is a multi layer network like any other neural network the difference being the convolution layers in the first few stages, which transform the 3-D volume of input space through a differentiable function. A typical CNN is made of stack of various layers as shown below with few set of Convolution layers to provide the depth of learning. Its from the number of convolution layers that network get the capability to learn various features of the problem space.

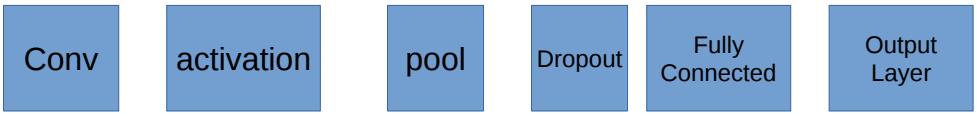

Fig 7 – Layers in typical Convolution Neural Networks

- Convolution Layer - Performs 2-D filtering on input images, consist of set of learning filters with narrow receptive field. Each filter is convolved along width & depth of input space computing dot product to produce 2-dimensional output per filter.

- Activation – This is layer of neurons that apply activation function to transform the output to -1 to 1. This defines the thresholds at which output of neurons to be triggered for change in input activations.

- Pooling Layer - This layer is non-linear down sampling used between various convolution layers to summarize & progressively reduce the spatial size of the representation.

- Fully Connected – This layer is used at end of CNN, the high level reasoning is done by fully connected layer. It has complete connectivity to previous convolution layers & performs function of flattening the feature map from previous convolution layers.

- Output layer – This is final output layer as is used to assign probability estimate to each class using soft-max activations.

Methodology

The basic CNN structure used for experiments is sequence of 3 to 5 convolution layers with 5x5 kernel followed by relu activations and max-pool taking average over 2x2 window. I also experimented with including dropout layers after various convolution layers. As suggested earlier we don't want to propagate back the error for blank in any digit sequence. This is accomplished by cutting off the corresponding output layers with blank digit. This can be done by enhancing the loss function to exclude the blank digits from contributing to the loss calculation.

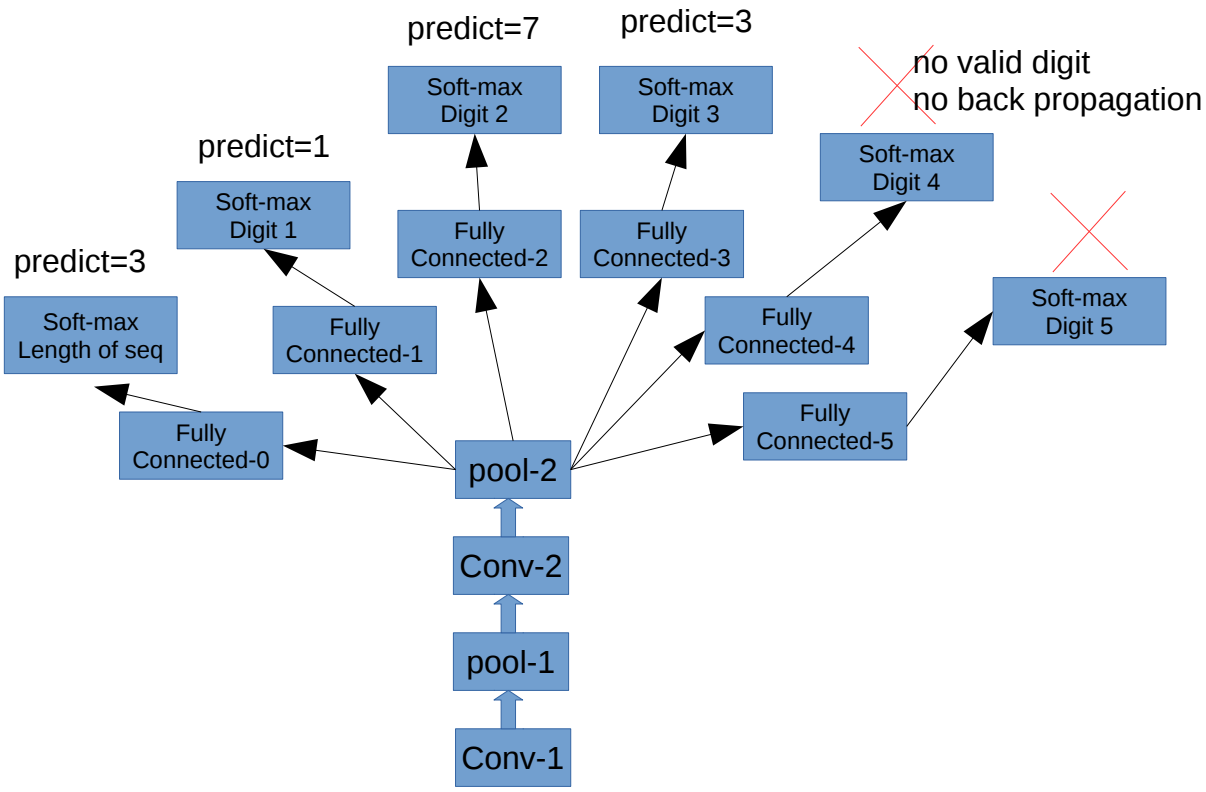

Fig 8 – Stop gradient back-propagation for empty digits

Hyperparameters

CNN are characterized by various parameters which can be broadly divided among network architecture & training parameters

Network Architecture

- Depth of Convolution network.
- Kernel Size, stride.
- Number & topology of fully connected layers.

Learning

- learning rate
- decay rate for learning
- regularization
- Dropout etc.

Experiments with MNIST type data.

I experiment with network hyper-parameters like kernel size, number of convolution layers, with & without fully connected layers. As far as training hyper-parameters experimented with learning rate & decay of learning rate.

I experimented with two & three layer convolution network with kernel depths of [16,32] & [16,32,48] with 5x5 kernel & various learning rates of 0.009, 0.05, 0.09 with staircase decay of 0.95 & batch size of 500. There is dropout layer at output of last convolution layer with 85% retention probability. All these networks were compared for the loss validation accuracy for single & sequence of digits.

The best results were achieved for network with three convolution layers [16,32,48] & one output layer per each digit & one classifier for length of sequence.

Convolution	Fully Connected	learning_rate	Validation accuracy after 40 epochs	
			Single digit	Sequence
[16,32,48],dropout	1	0.09	99.4%	97.8%
[16,32,48],dropout	1	0.009	92%	76%
[16,32,48],dropout	2	0.09	98.7%	94.2%
[16,32],dropout	1	0.09	98.9%	93.7%
[16,32],dropout	2	0.09	98.1%	89.3%

Table 1 : Accuracy mnist type sequence of digits classifier

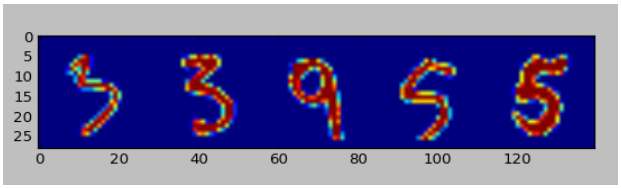

Fig 9 – misclassified as 53955-actual 33955

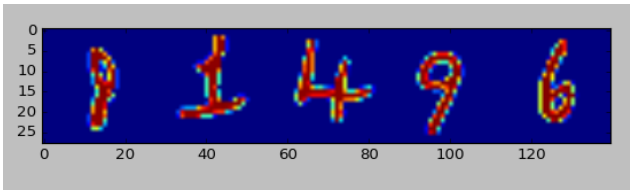

Fig 10 – misclassified as 11496 -actual 81496

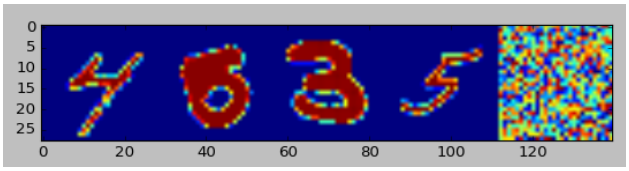

Fig 11 – misclassified as 4835-actual 4535

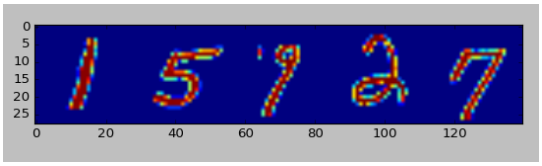

Fig 12 – misclassified as 15727 -actual 15927

Experiments with SVHN data.

Single digit classifier - As noted earlier SVHN is more complex dataset with lot of variations in input image. The training data of 65931 images digits of size 32x32, normalized with corresponding labels, validation set was based on 7326 images with same normalization as training images. Same as with mnist data the kernel size was 5x5, with stride of 2 per CNN layer, batch size was decided based on highest possible training set that fits in GPU memory for any tested CNN architecture. Also tried various learning rates with staircase decay of 0.95.

I experimented with three various configuration of convolution layers followed by single fully connected layer. The table gives asymptotic accuracy achieved by training for enough epochs, this ranges from few 40 to 150 epochs based on learning rates. Addition of dropout to CNN layers doesn't provide any advantage in accuracy scores, as the training data is diverse enough to need any regularization by using dropout. Best accuracy achieved was approx 92% for configuration as noted in table 2

Convolution layers	Fully Connected	learning_rate	Accuracy single digit classifier	
			Training	Validation
[32,64,128]	[128x10]	0.05	97.7	92.3
[48,64,128]	[128x10]	0.05	99.9	91.1

Table 2 : Accuracy SVHN single digit classifier

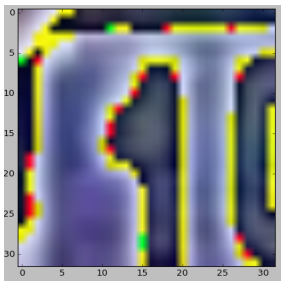

Fig 13 – misclassified as 4 - Label 1

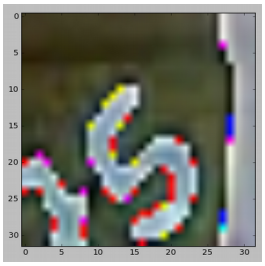

Fig 14 – misclassified as 6 – Label 5

Sequence of digits classifier – As noted earlier this data was obtained from public data provided at [1]. The data consist of 30061 training images & 3341 validation images. The predicted label is sequence of 6 digits, first being the digit count followed by the actual digits. I expect to achieve same degree of accuracy in predicting individual digits as with single digit classifier above. Accuracy of predicting the digit sequence depends on predicting the length of sequence & individual digits, probability of predicting correct sequence should be product of accuracy of individual digits.

As for single digit classifier I tried multiple architecture for convolution layers followed by one fully connected layer & one locally connected layer. The kernel size was 5x5 for each convolution layer followed by max pool of 2 for each convolution layer.

Best accuracy achieved was approx 94% for accuracy of individual digits in the sequence, it achieved accuracy of approx 78% for predicting the digit sequence.

Convolution-layers	Fully Connected	learning_rate (Decay_rate)	Accuracy sequence of digit classifier (Train/validation)	
			Digit	Sequence
[64,128,256,512]	[1024x512][512x10]	0.09(0.95)	(99.9%,94.3%)	(99.8%,77.6%)
[64,128,256,512]	[1024x512][512x10]	0.01(0.99)	(99.8%,93.8%)	(98.8%,75.6%)
[64,128,256,512]	[1024x512][512x10]	0.05(0.99)	(99.6%,93.9%)	(98.0%,75.9%)
[64,128,256,512]	[1024x512][512x10]	0.02(0.99)	(99.8%,93.6%)	(98.9%,75.1%)
[32,64,128,160]	[160x256][256x10]	0.09(0.95)	(99.8%,93.3%)	(99.2%,73.9%)
[64,64,64,128]	[128x10]	0.09(0.95)	(97.7%,93.6%)	(89.5%,74.8%)

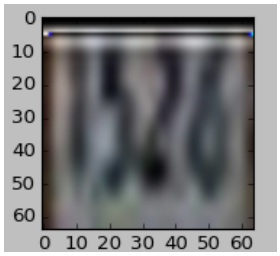

Fig 15
Label-1526 predict-1386

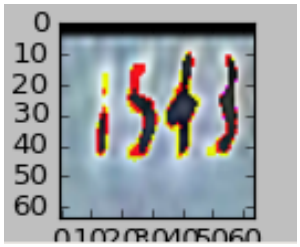

Fig 16
Label=1546 predicted=1523

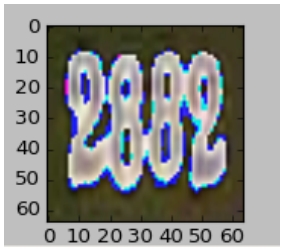

Fig 17
Label=2882 predicted=2888

Learnings & Summary

Convolution neural networks have found recently found applications in computer vision. This project was great learning experience in this domain. Simple framework of some CNN layers that performs the task of feature extraction from the input images followed by fully connected regression layers can perform quite complex computer vision tasks. The main challenge in the project has been experimenting with the trade-off between batch size & learning rate. Faster learning rate leads to high variance in learned parameters , CNN layers not generalizing enough the features in the training data. While lower learning rate leads to high bias with slow extraction of input features leading to low accuracy both for training & validation data.

Looking back I was able to achieve good accuracy of more than 90% for individual digits both for single digit & sequence of digits. However the accuracy for transcribing the overall sequence saturates at 80% for the validation data. I think I can improve on this by experimenting with deeper networks & appropriate learning rates.

Scripts & Code

mnist_classifier

Directory for scripts to download mnist data, create synthetic sequence of digits from mnist data. Dumped as three separate pickle files for training, validation & test datasets.

mnist_parse.py - script for creating synthetic data
mnist_model_seq.py - tensorflow model for mnist classifier
mnist_seq_classifier.py - mnist classifier for sequence of digits.

svhn_classifier

Directory for scripts to download svhn data, create separate npy files for training, validation & test datasets.
svnh_parse_single.py - script for creating 32x32 single digits from svhn data
svnh_parse_multi.py - script for creating 64x64 sequence of digits from svhn data
svnh_single_classifier.py -Classifier for single digits.
svnh_model_single.py - tensorflow model for single digit classifier.
svnh_seq_classifier_mask.py - Classifier for sequence of digit.
svnh_model_seq_mask.py - tensorflow model for sequence of digit classifier.

References

[1] Goodfellow et al. 2014, Multi-digit Number Recognition from Street View Imagery using Deep Convolution Neural Networks.

[2] The Street View House Numbers (SVHN) Dataset http://ufldl.stanford.edu/housenumbers/

[3] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning.

[4] CS231n: Convolutional Neural Networks for Visual Recognition. (http://cs231n.github.io/)