Capstone Project: Deep Learning

Street View House Numbers Recognition

Ryan Gao

Nov. 24 2016

I. Definition

1. Overview

Street View House Numbers is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. The dataset contains over 600,000 labeled digits cropped from street level photos. The ability to automatically transcribe those address number from a geo-located patch of pixels and associate the transcribed number with a known street address helps pinpoint the location of the building it represents.

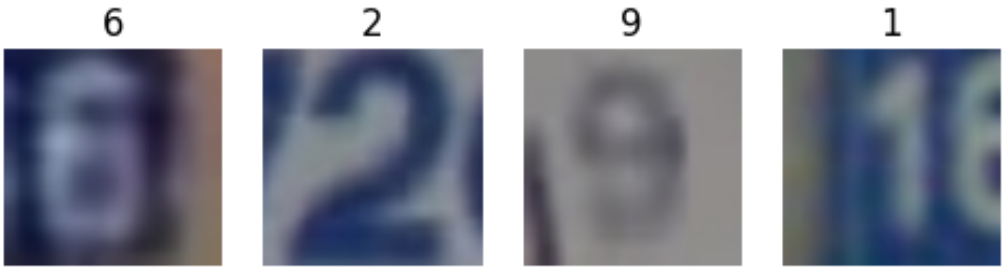
The problem of recognizing characters in images has been extensively studied in the last few decades. For instance, the MNIST digit dataset has been thoroughly addressed with algorithms approaching perfect performance. However, the MNIST dataset is venerable. With rapid advances in mobile phone cameras and computation capabilities, the more difficult problem of recognizing and understanding scene test is receiving increased attention.

Traditional approaches to recognizing arbitrary multi-digits numbers typically seperate out the localization, segmentation, and recognition steps. In this project, I built a 5-layer convolutional neural network, and focus on recognizing multiple digits simultaneously without segmentation. The model achieves 84.8% accuracy on character-level recognition and 74.5% accuracy on sequence level recognition in original images. With little adjustment, the model achieves 92.1% accuracy on cropped digits recognition. Thus this mode can be applied to individual character recognition or sequence recognition with limited length such as car license plates recognition.

2. Problem Statement

There are two types of dataset are provided. One is MNIST-like individual digits. The other is original images with character level bounding boxes. In this project, I mainly focus on recognizing the sequence on the original images which the model is built for.

Figure 1a. samples of individual digits



Street number transcription is a special kind of sequence recognition. It is obvious that digit recognition is much simpler than fully general object classification. There are only 10 classes we need consider. So the problem will be how do we recognize the digits as a sequence. Given an image, the task is to identify the number in the image. The number to be identified is a sequence of digits, $s = s_1, s_2, \ldots, s_n$.

Figure 1b. sample of one original image

Our basic approach is to train a probabilistic model of each digit then combine them to a sequence. Let S represents the sequence in the given image X and s_i is the ith digit in the sequence. Each of the digit variable s_i has 10 possible values. This means it is feasible to represent each of them with a softmax classifier that receives as input features extracted from X by a convolutional neural network.

At test time, we predict

$$S = (s_1, \ldots, s_5) = (argmax P(S_1 = s_1|X), \ldots, argmax P(S_5 = s_5|X))$$

The argmax for each character is computed independently.

3. Metrics

The labels in this project are both digits and there are no irreverent class in the images. So accuracy would be the sufficient metric for this project. When determining the accuracy of the digit transcriber, there two forms of metrics I use in this project.

Firstly, due to the characteristics of the model I built, which recognize the digits separately then combine them to a sequence, it's important to know how well of each digit is recognized. So I need calculate the single character recognition accuracy, which is defined as `def accuracy_single()` .

$$character\ accuracy = \frac{\sum single\ digit\ recognize\ correctly}{\#\ total\ digits\ in\ the\ dataset} \times 100\%$$

Secondly, as a sequence, one mistaken recognition may make a huge difference in understanding them in some application. Thus, I 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. This accuracy function is defined as `def accuracy_multi()`

$$Let\ I_i = indicator\ of\ the\ ith\ image\ recognized\ correctly$$

$$sequence\ accuracy = \frac{\sum_1^n I_i}{n} \times 100\%$$

II. Analysis

1. Data Exploration

The SVHN dataset is a dataset of about 200K street numbers, along with bounding boxes for individual digits. The dataset is in two format, one is of original images, the other is MNIST-like 32-by-32 images centered around a single character. Since our goal is to recognize the multiple digits simultaneously. So the following dataset all refers to the original ones.

Figure 2. images with bounding box

The original-formatted dataset is divided into three subsets: train set, extra set and test set. Each set contains the original images in png format with 3 color channels, together with a digitStruct.mat file. The digitStruct.mat file contains a struct called digitStruct with the same length as the number of original images. Each element in digitStruct has the following fields: name which is a string containing the filename of the corresponding image. bbox which is a struct array that contains the position, size and label of each digit bounding box in the image. Eg: digitStruct(300).bbox(2).height gives height of the 2nd digit bounding box in the 300th image.

Table 1. Train and test dataset

	Number of images	**Number of digits**
train.tar	33402	73257
test.tar	13068	26032

The size of images is show in Figure 3. The size of images in the train set is not uniformly distributed nor is the test set. The width of the image in the dataset ranges from 25 to 1083. Also, we can see that images in the test set are a bit larger than that in the train set. However, we don't need remove those outliers, since all the images will be cropped and resize to 32×32 in the preprocessing stage according to bounding box given the dataset.

Figure 3. sizes of train and test dataset

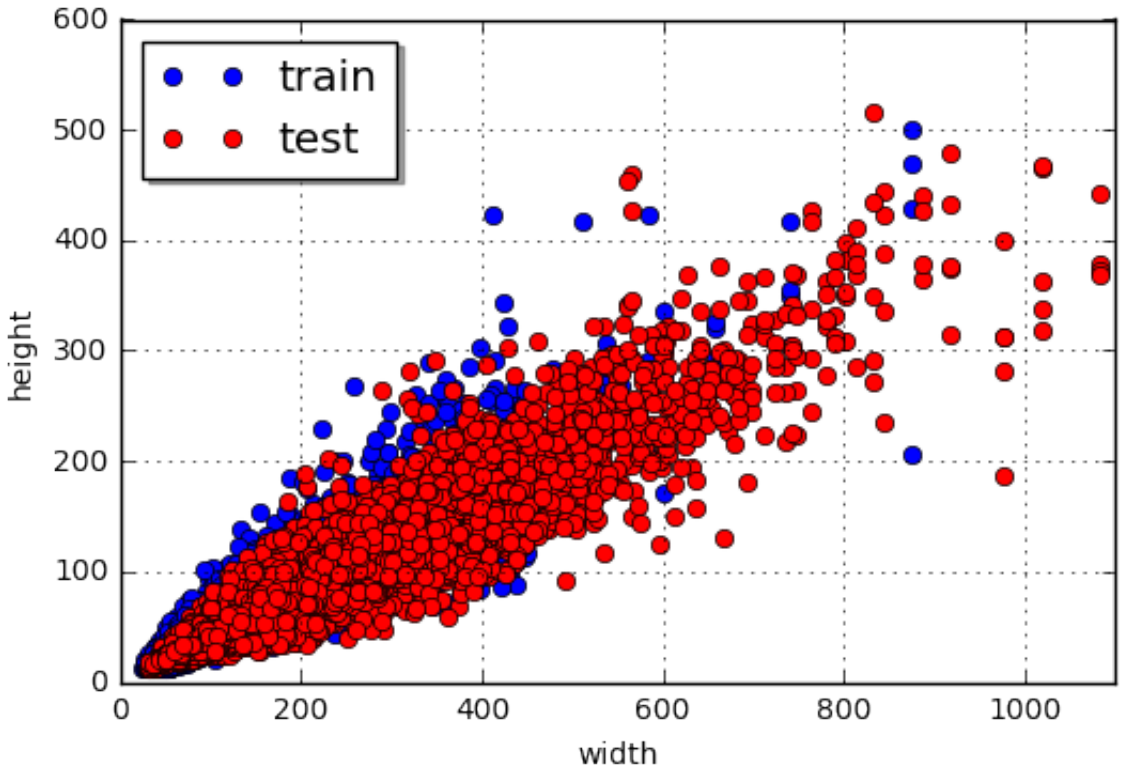


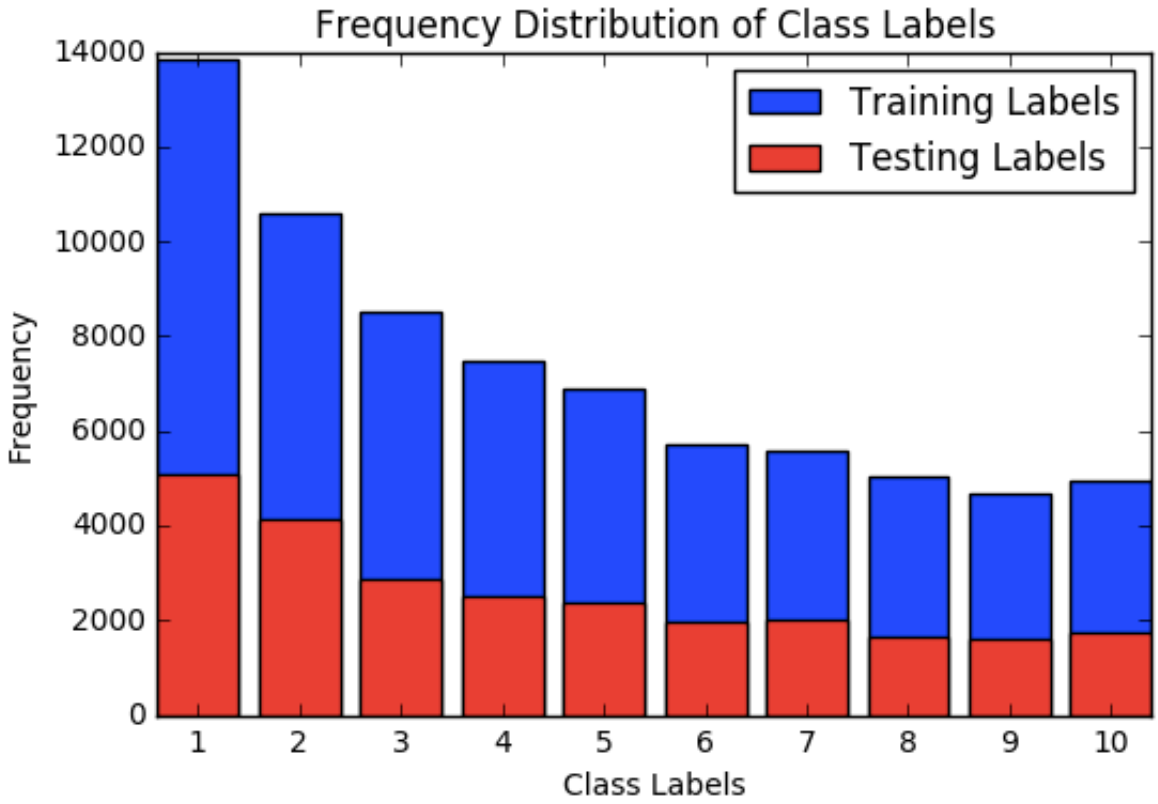
Table 2. sizes of images in train and test dataset

Size (pixels)	mean	median	max	min
train height	57.2	47	501	12
train width	128.2	104	876	25
test height	71.5	53	516	13
test width	172.5	132	1083	31

Another special property of the street number transcription problem is that the sequences are of bounded length. The length ranges from 1 to 6, Actually there is only one image containing more than 5 digits in the train and test dataset. So we remove it from the dataset.

Figure 4. image with 6 digits

Figure 5. Distribution of digits in images



As shown in Figure 5. The distribution of each digit is imbalanced. The frequency of (1,2,3) is apparently higher than other digits. However, the test dataset preserves same imbalances.

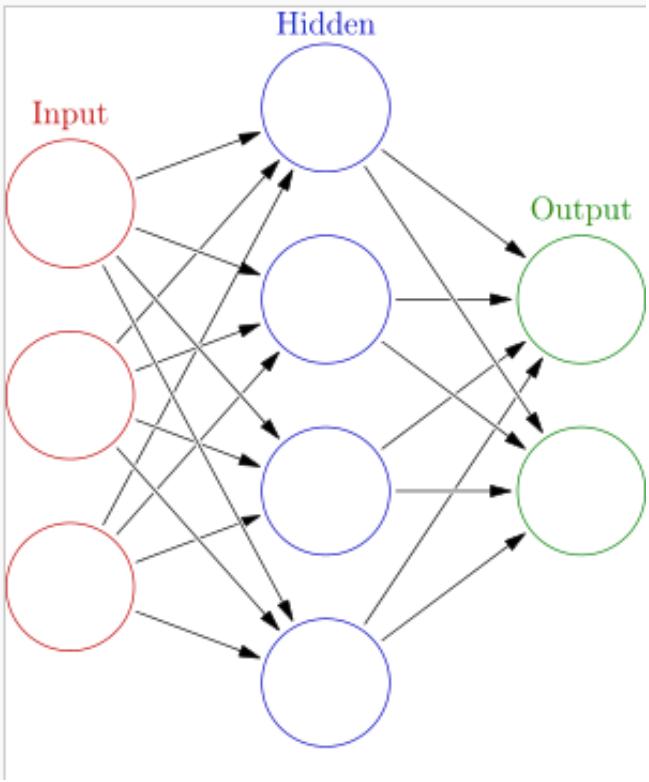
2. Algorithms and Techniques

The general literature on image understanding is vast. Object classification and detection has been driven by many large scale visual recognition challenge such as ImageNet. Various model and methods have been used in both research and industrial work. Here I focus on reviewing the techniques that either have been the major component of the model or help reduce training time and improve accuracy.

Convolution Neural Network

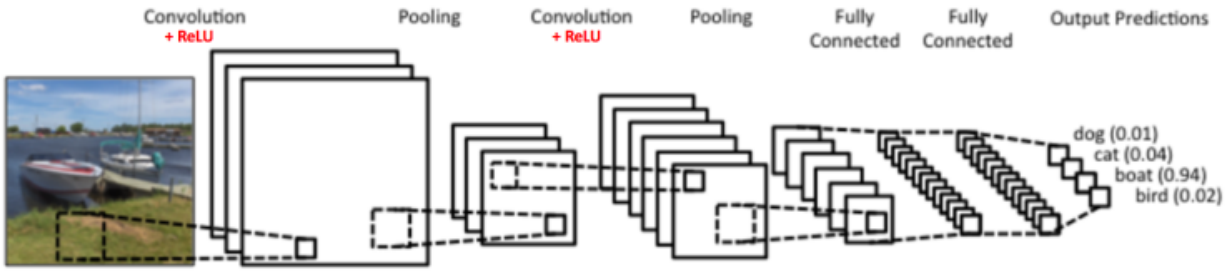
Artificial Neural Network(ANN) is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs, which typically organized in layers. In a fully connected layer each neuron is connected to every neuron in the previous layer, and each connection has its own weight. This makes no assumptions about the features in the data. It's also very expensive in terms of memory (weights) and computation (connections).

Figure 6. Neural Networks



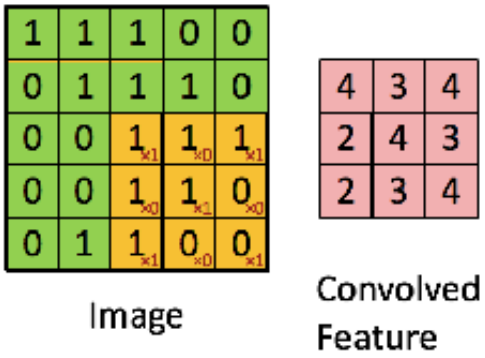
Convolutional Neural Network(CNN or ConvNets) is one type of the ANNs, and has proven very effective in areas such as image recognition and classification. In a convolutional layer each neuron is only connected to a few nearby neurons in the previous layer, and the same set of weights is used for every neuron. This connection pattern only makes sense for cases where the data can be interpreted as spatial with the features to be extracted being spatially local and equally likely to occur at any input position.

Figure 7. A simple ConvNet



For images, CNNs have repetitive blocks of neurons that are applied across space, these blocks of neurons can be interpreted as 2D convolutional kernels, repeatedly applied over each patch of the image. In one convolution layer, as shown in Figure 8, the yellow $\mathbf{3} \times \mathbf{3}$ matrix is called 'filter' or 'kernel' and the matrix formed by sliding the filter over the image and computing the dot product is called the 'Convolved Feature' or 'Feature Map'. We compute element wise multiplication between the filter matrix and the image matrix and add the multiplication outputs to get the final integer which forms the output feature map.

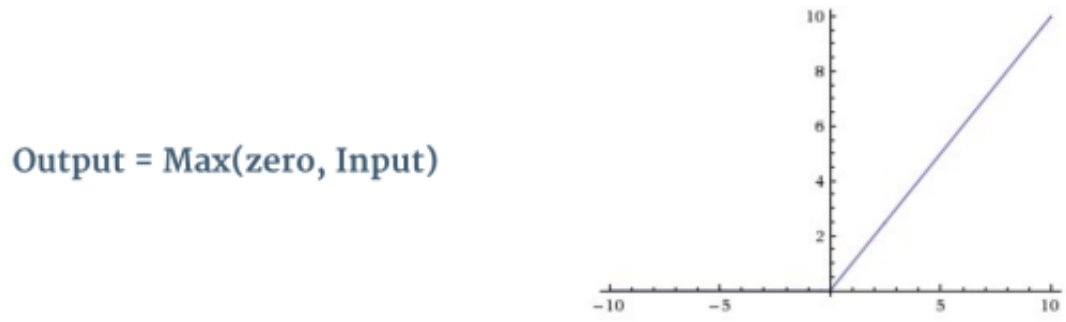
Figure 8. Convolution kernel and feature map



ReLu

ReLU is the abbreviation of Rectified Linear Units. This is a layer of neurons that applies the non-saturating activation function $f(x) = \max(0, x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other non linear functions such as $tanh$ or $sigmoid$ can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

Figure 9. ReLu function



Local Response Normalization

In neurobiology, there is a concept called “lateral inhibition”. This refers to the capacity of an excited neuron to subdue its neighbors. We basically want a significant peak so that we have a form of local maxima. Local Response Normalization (LRN) layer implements the lateral inhibition in the CNN layers.

Denoting by $a^i_{x,y}$ the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity, the response-normalized activity $b^i_{x,y}$ is given by the expression

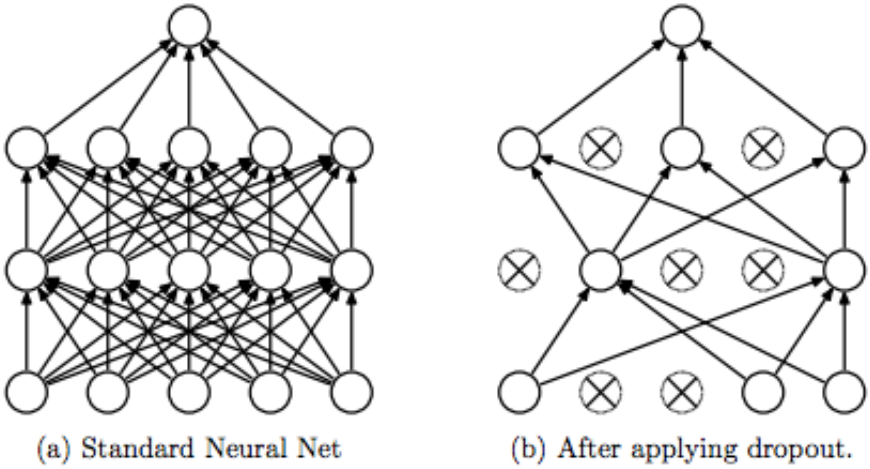
$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2)^\beta$$

where the sum runs over n "adjacent" kernel maps at the same spatial position, and N is the total number of kernels in the layer.

Response normalization reduces our single digit and sequence recognition error rates by 0.9% and 0.7%, respectively.

Dropout

Figure 9. dropout



"Dropout" consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in back-propagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

Without dropout, my network exhibits substantial overfitting. Applying dropout gives me roughly 5% accuracy improvement on character-level recognition and 7% improvement on sequence-level recognition.

Adam Optimization

Adam is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Adam combines the advantages of two popular methods: AdaGrad , which works well with sparse gradients, and RMSProp, which works well in on-line and non-stationary settings. Some of Adam’s advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyper-parameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.

3. Benchmarks

Human performance on sequence-level recognition reaches 98%. And the state-of-art of sequence transcription accuracy is 96.03%. The model is built by Goodefellow $et\ al$. Their model also achieves a character-level accuracy of 97.84%. However, the parameters of their model is so large that it took them 6 days to train the model.

Apparently, the MacBook I use couldn't bear such computational task and the number of parameters must be reduced due to the 8G RAM. The threshold I set for this project is to reach 80% -85% accuracy on sequence transcription and 90% accuracy on character-level recognition on which level people may find it useful when transcribing the digits automatically.

III. Methodology

1. Preprocess

1. Dataset composition

Since there are given no information about how the sampling of these images was done. I compose the validation set from the randomized training samples, yielding 6000 samples.

Table 3. train, test and validation dataset

	sample count
train_dataset	27401
test_dataset	13068
valid_dataset	6000

Figure 5. Distribution of sequence length in each dataset

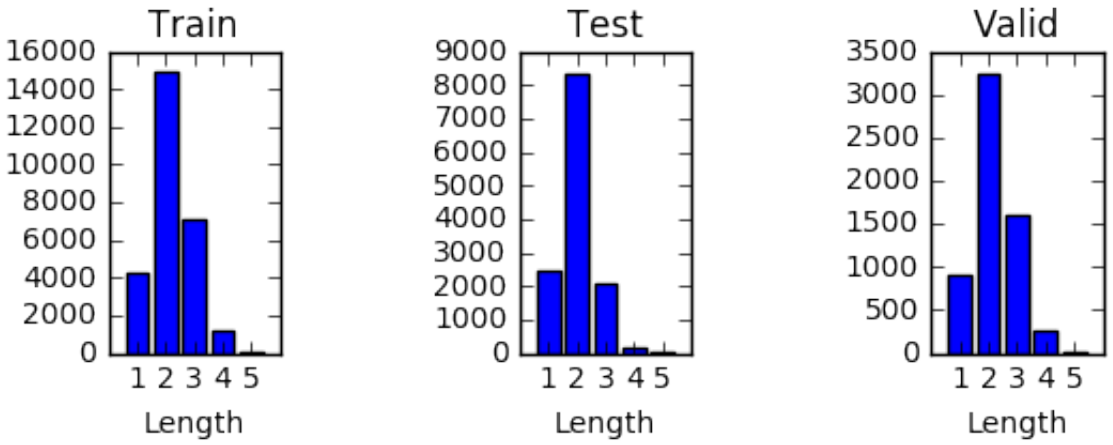
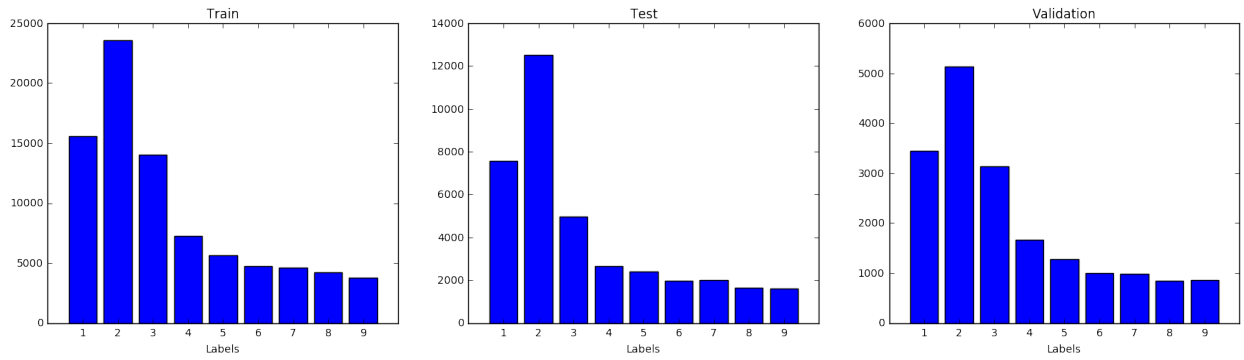


Figure 6. Distribution of digits count in each dataset



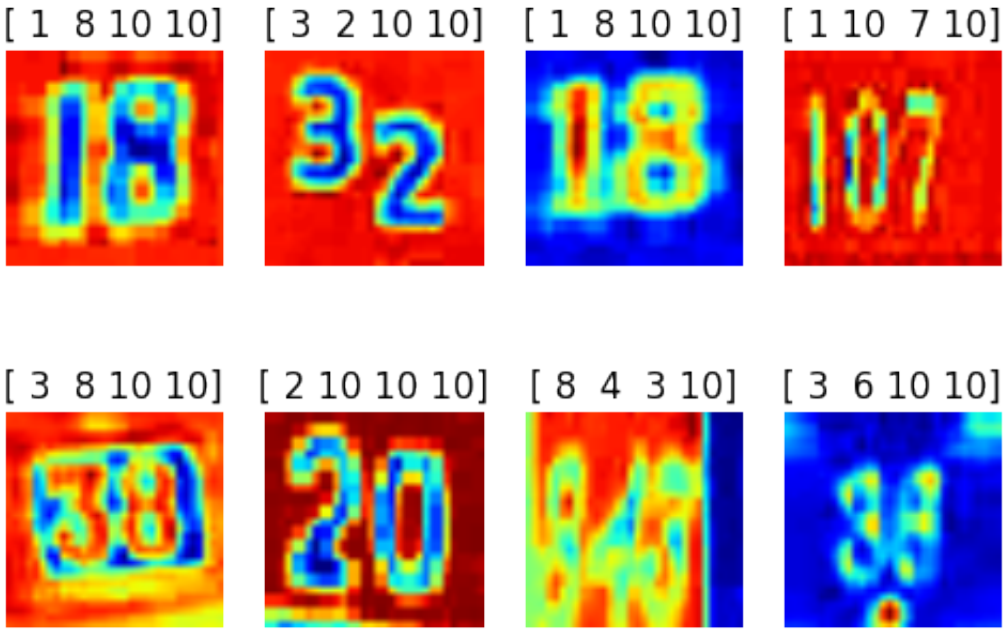
2. Reshape & Normalization

I preprocess the dataset in two aspects.

First, I find the small rectangular bounding box boundaries by reading the digitStruct.mat. I then crop the original images to remove redundant information. Therefore, the size of images is reduced. After cropping, I resize the images to 32×32 for computational convenience.

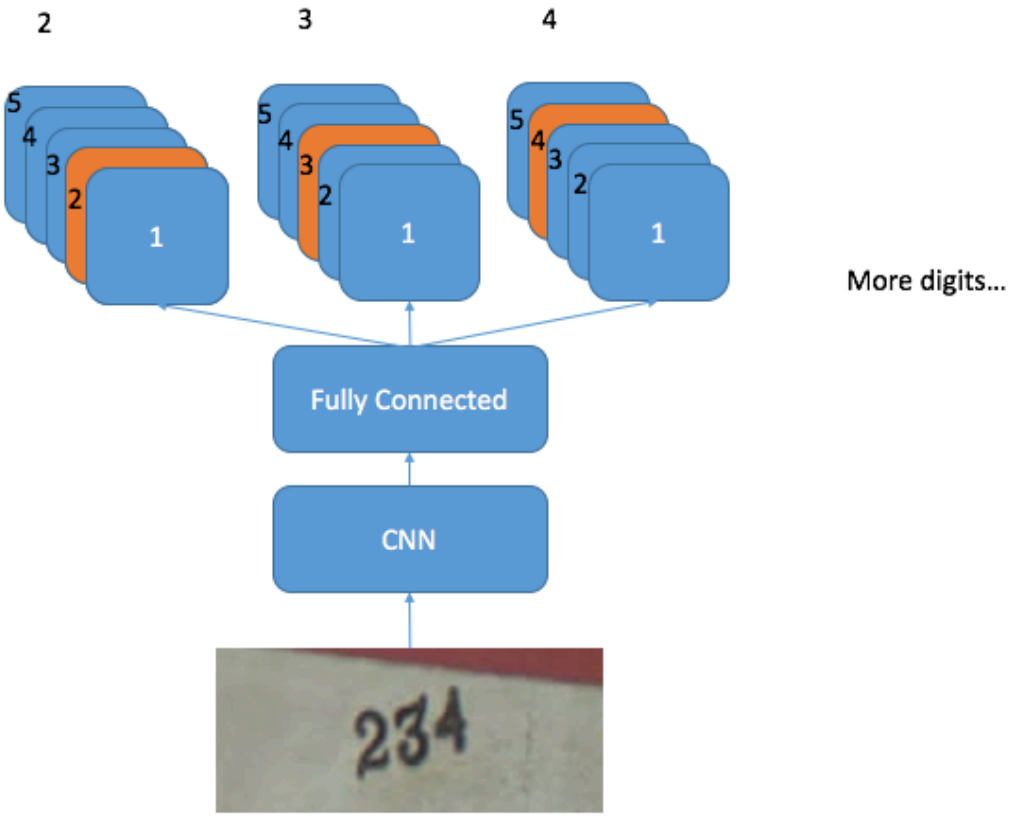
Second, unlike the MNIST dataset, the SVHN dataset comes from the real world image, which has 3 color channels. Our goal is to recognize the digits, so the color doesn't matter. I first convert the images to greyscale, then subtract the mean of each image and divide by its standard deviation. The image information is stored into numpy ndarray.

Figure 10. images after normalization (10 stands for none)



2. Implementation

Figure 11. Architecture



The base ConvNet architecture is composed of three repeatedly stacked feature stages and one fully connected hidden layer, and five locally connected layers for recognizing each digit. Each convolution stage contains a convolution module, followed by a rectified module and a max-pooling module.

All convolution kernels are of size 5×5 . The first convolution layer produces 16 features convolution filters, while the second and third convolution layers output 32 and 64 features. All convolution layers use zero padding and zero stride on the input so that the representation size remains the same. All the max pooling window size is 2×2 with a stride 2×2 . The fully connected layer contains 1024 nodes each.

I trained with dropout applied to the fully connected hidden layer. Adam optimizer is used for stochastic optimization. Regularization constant, learning rate were both tuned to 0.001.

Each the input training batch contains 64 images in size of 32×32 with 1 color channel. It normally takes 20,000 steps for training the model.

3. Refinement

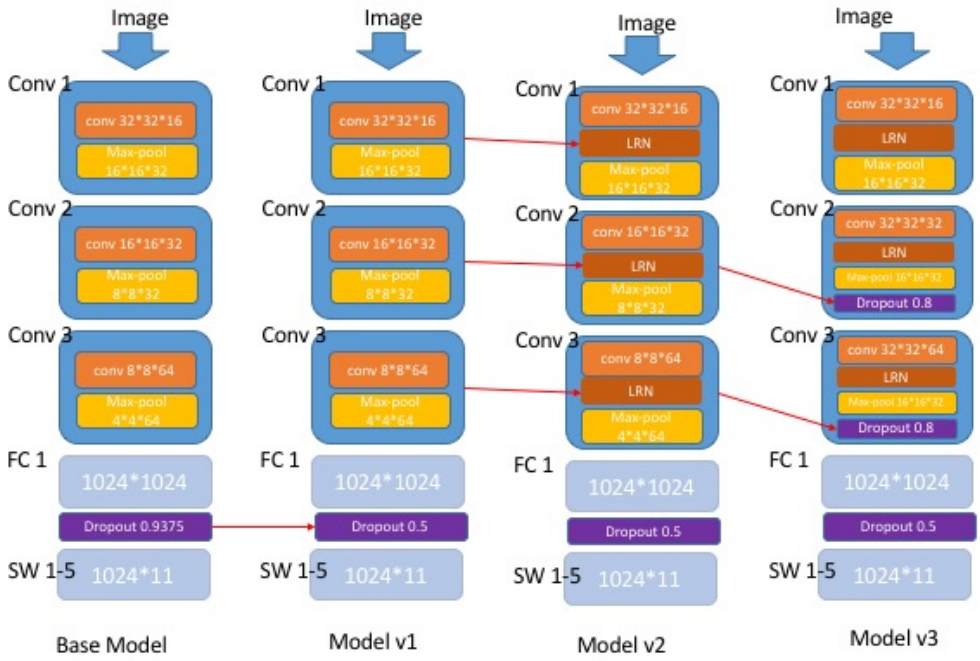
There are many techniques for refinement and many hyper-parameters need to be tuned. To do a grid search on deep neural network will cost huge amount of effort and time. The refinement approach I adopted is observing the loss and accuracy on validation set and then decide which approach may improve the accuracy on test dataset.

The base model contains 3 ConvNet each followed by a max-pooling layer and 1 fully connected layer followed by 5 locally connected layers. Dropout is only applied on fully connected layer with value of 0.9375.

I mainly adjust the model in 3 ways:

1. Add/remove one layer
2. Increase/decrease the number of hidden units
3. Change the probability in dropout

Figure 12. Architect evolution



IV. Results

1. Model Evaluation and Validation

Table 4. Model Accuracy

Model	Accuracy_single	Accuracy_multi
Base model	77.8%	63.4%
Model v1	82.3%	71.5%
Model v2	82.2%	72.2%
Model v3	84.8%	75.9%
Model v3 (on single digit dataset)	92.1%	None

On comparing these four models, we can notice that whenever we change the dropout value or add a dropout layer, there will be a leaping growth on final accuracy. Figure 13a shows the loss of train predictions at each epoch of model v2 and v3, and Figure 13b shows the validation error corresponding to those loss. The loss of Model v2 is always smaller than that of Model v3 in the same epoch showing that v2 performs better than v3 on training dataset. However, the validation error of Model v2 is greater than that of Model v3, which indicates that Model v3 is better generalized. Reducing the dropout rate or adding dropout layers helps avoid overfitting

on the training dataset and makes the Model v3 more robust.

Figure 13a. Train prediction loss of Model v2 and v3

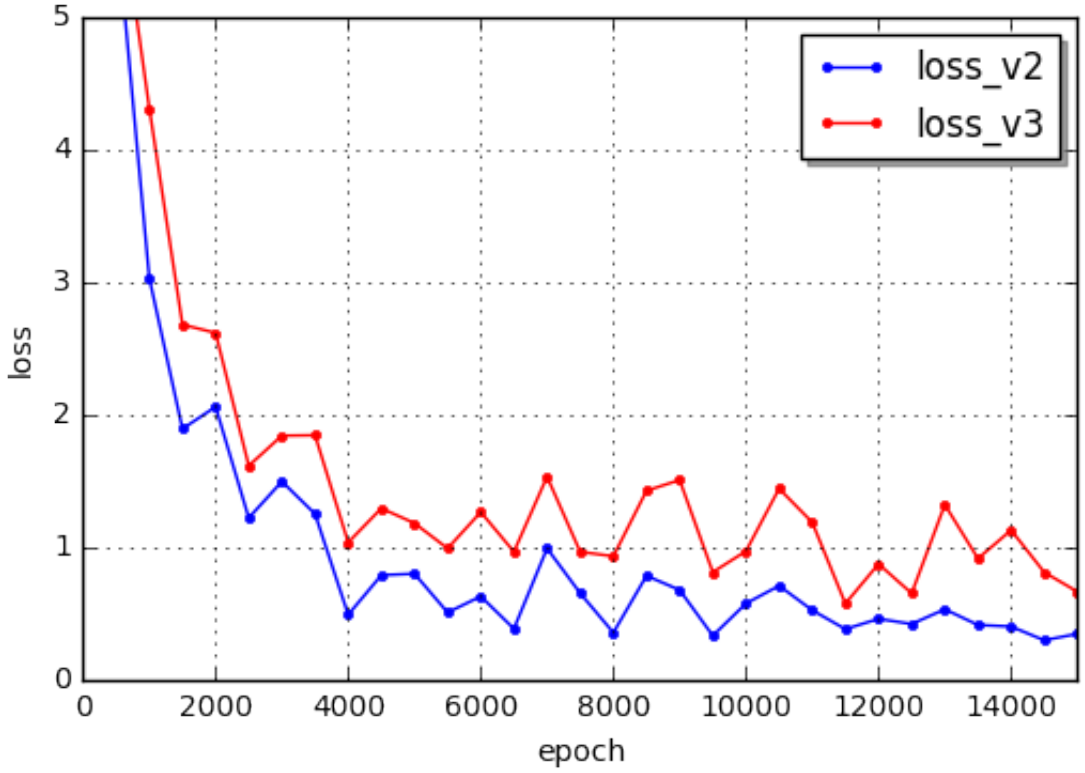
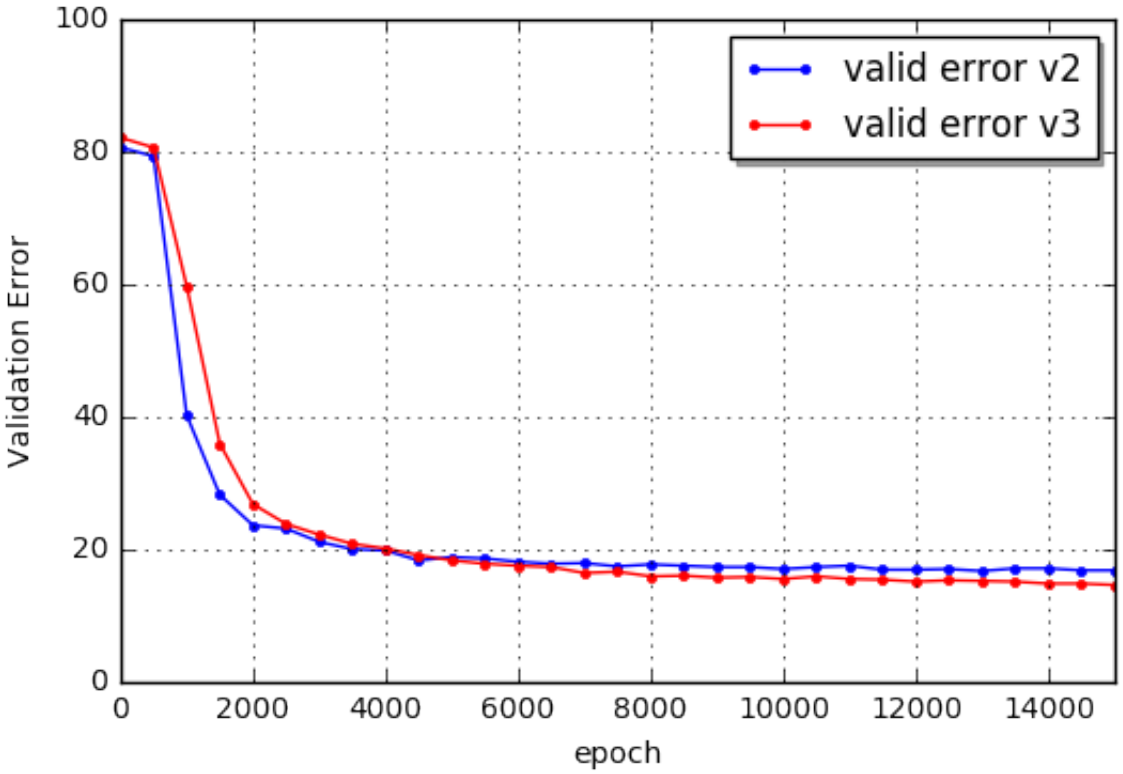


Figure 13b. Validation error of Model v2 and v3



After several turns of refinement, Model v3 achieves 84.8% accuracy on individual digits recognition and 75.9% accuracy on sequence recognition on the original images. As shown in Figure 12 , this model contains three ConvNets stage and 2 fully connected layers. Each ConvNet stage contains a local response normalization layer, a max-pooling layer. And dropout layers are added to all layers except the input layer. Specific parameters are listed in Table .

Table 5a. Model v3 parameters

Layer	Filter	Stride	Depth	Output
conv_1	5×5	1×1	16	$32 \times 32 \times 16$
max_pool	2×2	2×2	16	$16 \times 16 \times 16$
conv_2	5×5	1×1	32	$16 \times 16 \times 32$
max_pool	2×2	2×2	32	$8 \times 8 \times 32$
conv_3	5×5	1×1	64	$8 \times 8 \times 32$
max_pool	2×2	2×2	64	$4 \times 4 \times 64$
Fully connect				1024×1024
Output				1024×11

Table 5b. Model v3 parameters

Other parameters	Value
Dropout(on ConvNet)	0.8
Dropout(on FC)	0.5
learning rate	0.001

In addition, I reduced the number of hidden units in Model v3 then applied it on the less difficult individual digit dataset. It achieves 92.1% accuracy on recognizing single digits.

2. Justification

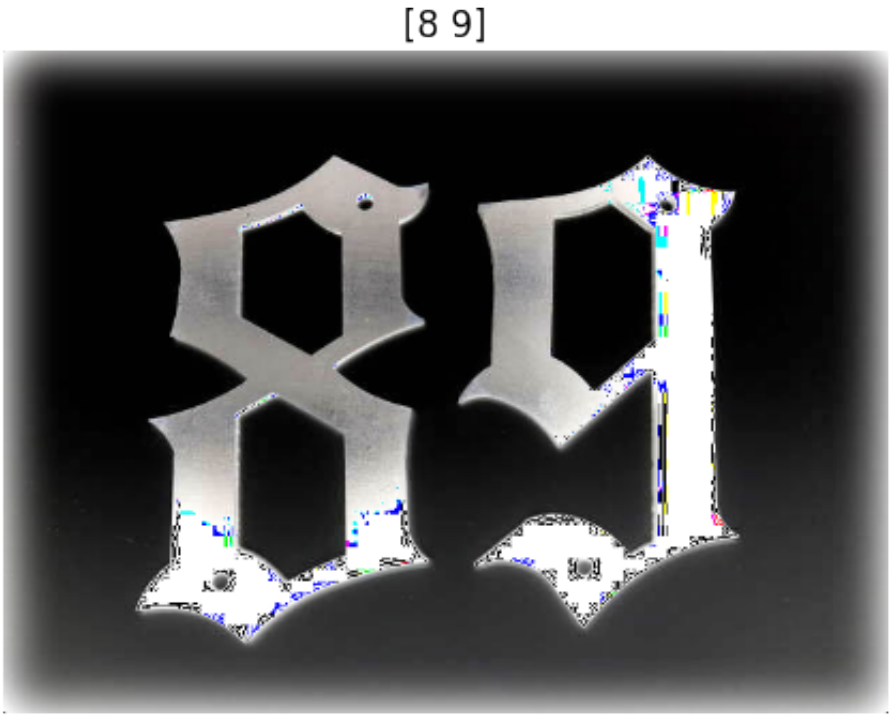
The best model I built so far reaches 75.9%, while the human's performance on this dataset reaches 98% accuracy. Comparing to threshold this is not accurate enough for transcribing sequence digits in real world. However, when using the MNIST-like dataset, I reached 92.1% accuracy on recognizing individual digits. I think this implementation is an adequate solution to recognizing single digit in real world.

V. Conclusion

1. Reflection

In this project, we first crop the images according to the given boundary information then normalize them to mitigate the variation of each image. Then the 5-layer convolution neural networks do the segmentation and localization of ordered sequences of object simultaneously. It taking the preprocessed images as input outputs the possible values of each digits. Those predicted digits form up the sequence in the image. Similar procedures could be implemented in real world applications for recognizing digits sequence.

Figure 14. correctly classified image



However, one drawback of this model is that our approach is trained fully supervised only, and we use the digits' boundaries in the preprocess stage to remove redundant information. Without these boundaries and preprocess, the street view numbers recognition accuracy may decrease a lot. As shown in Figure 15, when dealing with ill-forms of street numbers, it's difficult for the model predicting correctly.

Figure 15. misclassified image

[1]

Yet, one interesting but also difficult part of this project is tuning hyper-parameters. With a little adjustment, the accuracy of the model change significantly. Finding the best combination of hypermeter definitely require many experiments and deep understanding the convolutional neural networks.

2. Improvement

Due to the Mac's computational limitation, the training process is too long and I couldn't add more hidden layers or units. For further accuracy improvement, we can add more hidden layers by using faster processors of GPUs.

On the other hand, in the preprocess step I reshape the image to $\mathbf{32} \times \mathbf{32}$ format. However, there are many images of higher resolution in the dataset . And in real world, the digital camera contains more and more pixels. Thus, a training model that could process larger image size need implementing when we want to build an end-to-end transcriber in application.

References

1. SVHN dataset: http://ufldl.stanford.edu/housenumbers
2. 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*. (PDF)
3. Pierre Sermanet, Soumith Chintala, Yann LeCun *Convolutional Neural Networks Applied to House Numbers Digit Classification*
4. Diederik P. Kingma,Jimmy Lei Ba *ADAM- A METHOD FOR STOCHASTIC OPTIMIZATION*
5. Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks*
6. G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. *Improving neural net-works by preventing co-adaptation of feature detectors.*

7. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky,Ilya Sutskever, Ruslan Salakhutdinov *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*
8. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton *ImageNet Classification with Deep Convolutional Neural Networks*
9. Matthew Earl *Number plate recognition with Tensorflow*
10. https://en.wikipedia.org/wiki/Convolutional_neural_network
11. A Basic Introduction To Neural Networks
12. https://www.reddit.com/r/MachineLearning/comments/3yy7ko/what_is_the_difference_between_a_fullyconnected/
13. Feature extraction using convolution