Capstone Project

Machine Learning Engineer Nanodegree

October 21st, 2016

I. Definition

Project Overview

Objective of our project is to create a software application that can interpret numbers from real images and transcribe it immediately.

Application of above project entails OCR (Optical character recognition) on real world images having numbers in unknown fonts, colors and background.

We particulary focus on SVHN dataset (http://ufldl.stanford.edu/housenumbers/). SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images. Identification of numbers on images of this dataset would apply to geo tagging to help finding actual location of particular address.

Problem Statement

In this project, we want to identify arbitrary multi-character text in unconstrained natural photographs using machine learning techniques.

To solve this problem, we create models by training on SVHN dataset. Our model is deep neural network with numbers of hidden convolutional layers (CNN) on image pixels. First we need to preprocess and crop images to remove noise and pass it as training dataset. We use tensorflow library with python interface to create and train our neural network while avoiding overfitting. Our model correctly identifies number of digits and output digits corresponding to number in images. Actual digits corresponding to images are provided with dataset as labels which we used to train and check accuracy of our model.

Metrics

We choose percentage of images correctly transcribed as our accuracy metric i.e.

*Accuracy = (100*Number of images correctly interpreted)/(total number of images)*

Above metric signifies correctness of our model by being proportional to number of images correctly identified.

We might also choose our metric at more granular level by taking into account number of digits correctly identified, but by choosing all digits of an image as a unit we make our metric more robust and accurate.

II. Analysis

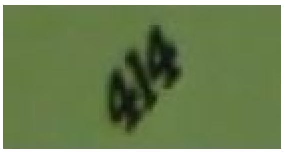
Data Exploration

BASIC STATISTICS OF DATA

Below table lists properties of training and testing images

	TRAINING	TESTING
NUMBER OF IMAGES	33402	13068
MINIMUM WIDTH	25	31
MAXIMUM WIDTH	876	1083
AVERAGE WIDTH	128.28	172.58
MINIMUM HEIGHT	12	13
MAXIMUM HEIGHT	501	516
AVERAGE HEIGHT	57.21	71.56
MINIMUM AREA	300	403
MAXIMUM AREA	438876	478686
AVERAGE AREA	10103.64	18461.23

SAMPLE IMAGES

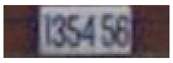


Below are distribution of number of digits in images.

NUMBER OF DIGITS IN IMAGE	**COUNT (TRAINING IMAGES)**	**COUNT (TESTING IMAGES)**
1	5137	2483
2	18130	8356
3	8691	2081
4	1434	146
5	9	2

As we see in above tables there are wide variation in image sizes, colors and fonts. We need to standardize the images.

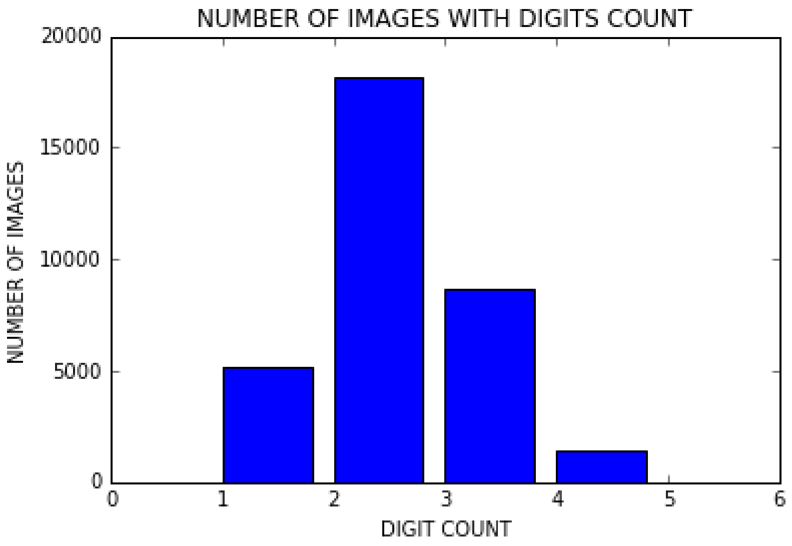
- Below is an outlier image with 6 digits as we limits our model to only 5 digits.



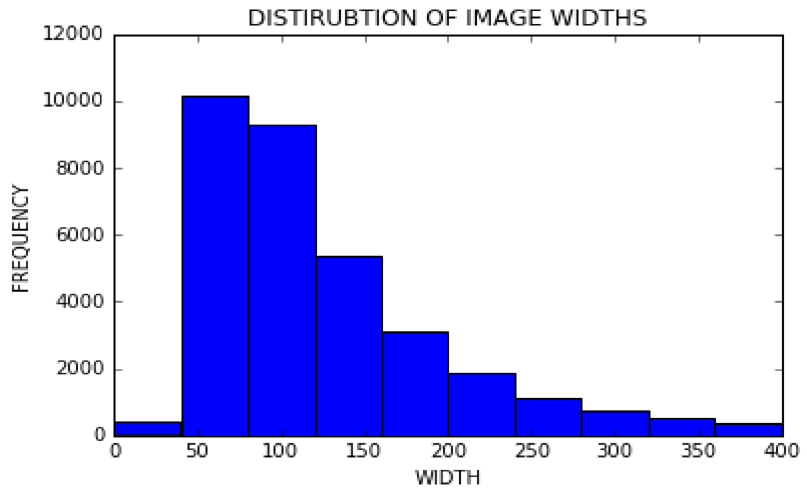
- Next is an image with maximum area.

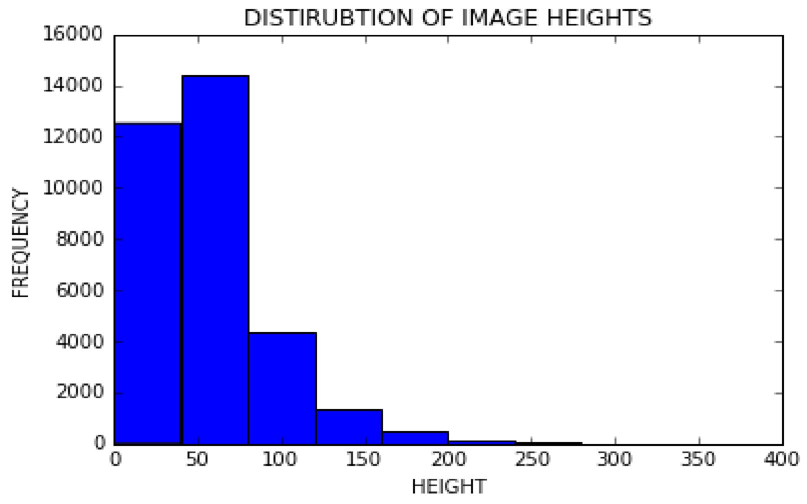
Exploratory Visualization

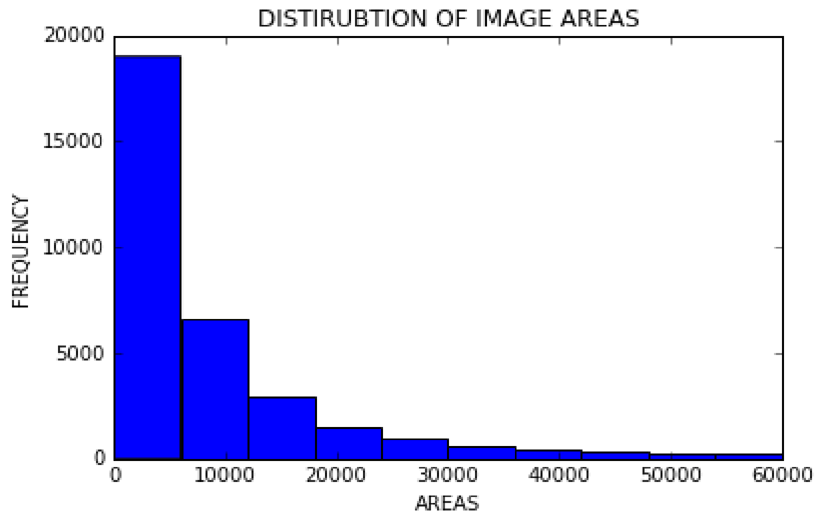
Below is histogram of number of digits in training images. We see that most of image has multiple digits compelling us to use deeper convolutional neural networks.



Below 3 histogram plots specify distribution of the dimensions of images viz. widths, heights and areas. We see wide variance in distribution compelling us to standardize size of images before feeding it to model training.







Algorithms and Techniques

We use deep convolutional neural networks (CNN) as our model taking cue from recent successes in image recognition through CNN. We use multiple convolutional layers with RELU and maxpool units followed by a fully connected layer.

We used 4 layers neural network as explained below:

1) LAYER 1 is convolutional with size [5, 5, 1, 16]. It specify that convolution is applied on 5*5 patches of image having single depth and output with depth of 16. It is followed by RELU unit to induce non-linear activation function and max_pool unit of size 2*2 to downsample the image
2) LAYER 2 is convolutional with size [5, 5, 16, 32]. It specify that convolution is applied on 5*5 patches of image having depth of 16 and output with depth of 32. It is followed by RELU unit to induce non-linear activation function and max_pool unit of size 2*2 to downsample the image.
3) LAYER 3 is convolutional with size [5, 5, 32, 64]. It specify that convolution is applied on 5*5 patches of image having depth of 32 and output with depth of 64. It is followed by RELU unit.
4) LAYER 4 is a fully connected layer of 5 neurons of size (64*11) having 64 input and 11 outputs for each possible digit (0-9) and other (10) for digit not present at that position.

We train above network by finding its parameters using an optimizer where we inputs data as a batch size of 64 and using cost to minimize as cross entropy of model output and actual digits.

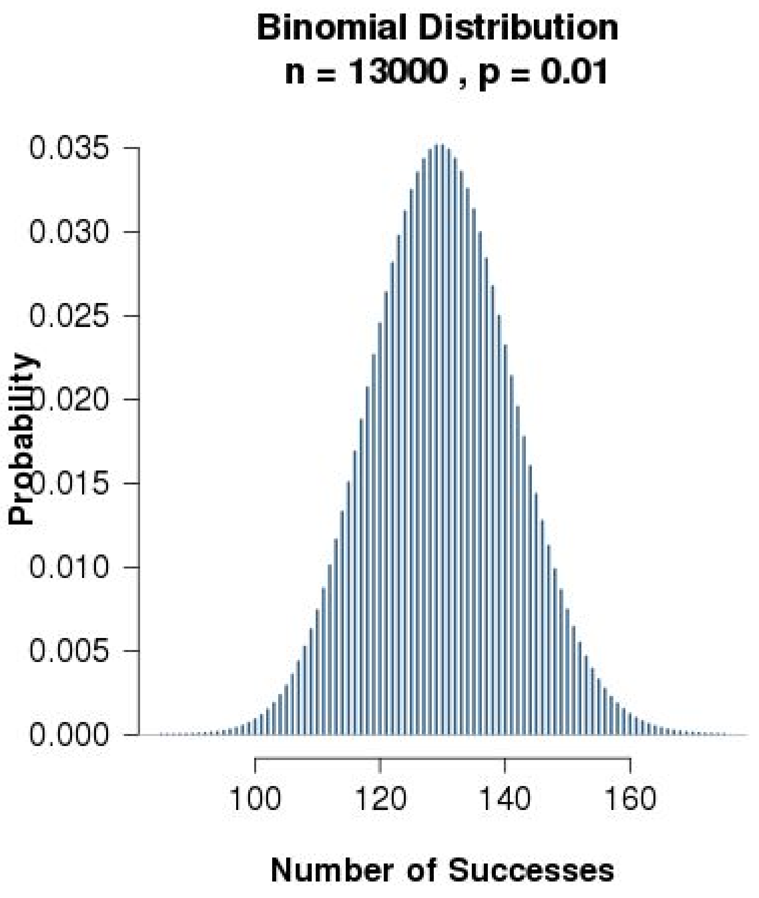
To avoid overfitting, we also used drop-out while training.

Benchmark

We make benchmark such that our model should be significantly better than random guess.

Probability of guessing correctly 1 digit out of 10 possibility is 0.1 so probability of correctly guessing 2 digits are 0.1 * 0.1 =0.01

As we found that most of our numbers have 2 digits, we consider the binomial distribution of correctly identifying images as below. We see that it is very difficult to guess correct image due to low probability, so a model better than random guess would be very good if it is even able to identify 1000 images (10%) of images in test set correctly.



III. Methodology

Data Preprocessing

As found in analysis step, there are wide variations in dimensions and other characteristics of images, we preprocess all images by following steps.

- Finding bounding box of all digits
    - We are given boundary rectangles of all digits in images separately. We find combined boundary rectangle of all digits using spatial union of all these boundaries and separate image in this boundary to remove all other noise from image.
- Rescale image to 32*32 size
    - As there are wide variation in dimension of images we standardize the image to 32*32 so as to have a standardized size.
- Convert to gray
    - To simplify model of our digit identifier, we convert our image to grayscale

Check below examples handled by preprocessing

ORIGINAL IMAGE

PROCESSED IMAGE

ORIGINAL IMAGE

PROCESSED IMAGE

We also created labels corresponding to numbers to validate and train model as expected digits.

E.g. above “48” image would be represented in model as
array([4, 8, 10, 10, 10]).

Here absent digit is represented as 10.

Implementation

We use tensorflow framework to implement and train our neural network.

We use tensorflow placeholders for our input dataset and output labels. We use tensorflow variables for neural network parameters like weight and biases.

For implementing convolution layer we use conv2, max pooling by maxpool and rectified non linear activation by relu components of tensorflow.

We use nn.sparse_softmax_cross_entropy_with_logits utility to find cost function used by optimizer present from train module in tensorflow. We used sparse softmax function as we need to compare particular digit output out of 11 possibilities.

We run the optimizer with number of steps specified by NUM_EPOCHS as number of complete iterations on training data.

There were number of repetitions in our code as we needed to implement similar kind of layers and outputs for number of digits.

Refinement

As we use number of epochs as parameter to decide on number of iterations on training set. We tried with different epochs and got accuracies as below

NUMBER OF EPOCHS	ACCURACY
1	43%
2	53%
4	59%
6	62%
8	63%
10	63%

We see that our optimizer is converging on 6 iteration, which we choose as final parameter in our model.

We also tried with couple of optimizers. With GradientDescent Optimizer, we got only 52% accuracy, but with MomentumOptimizer we achieved 63% accuracy so we used MomentumOptimizer in our model.

IV. Results

Model Evaluation and Validation

We validated our model on an independent testing set. By validating on independent data set, we made sure that we are avoiding the case where our model has overfitted the data while training.

Accuracy of our final model on testing dataset is 62%

We closed on number of epochs as 6 with optimization algorithm as Momentum optimizer and drop-out rate of 0.95.

We also confirmed output of our model on some samples of testing set to confirm validity of our accuracy function as below which predicts correct labels.

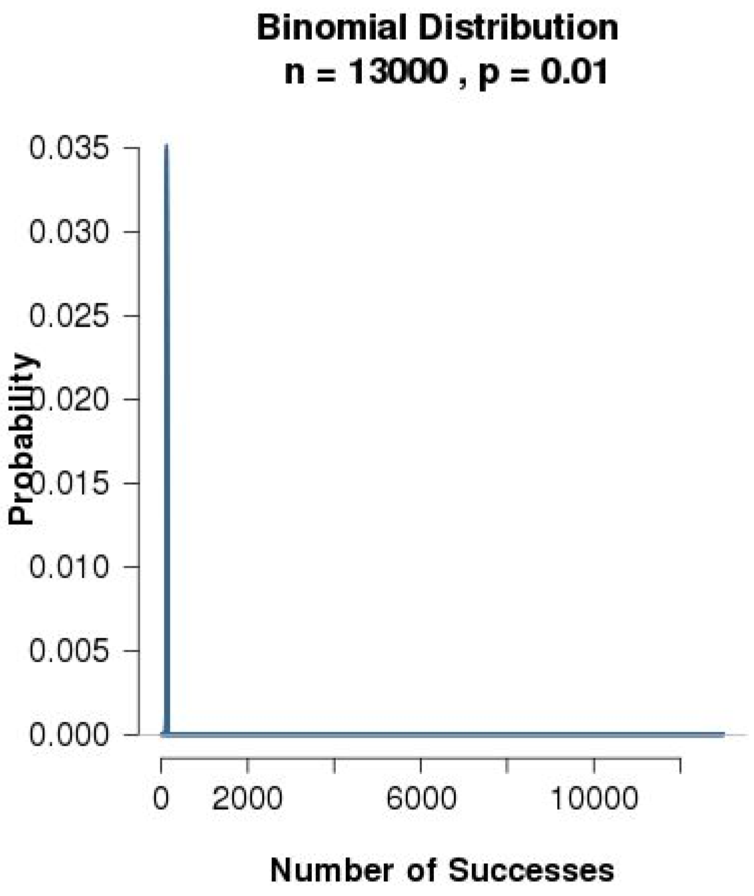
```
# check some image randomly for sanity checking
with graph.as_default()
   print(np.argmax(test_prediction, 2).T[11])
   print(np.argmax(test_prediction, 2).T[21])
[ 2  0 10 10 10]
[ 1  6 10 10 10]
```

To check robustness of our model, we tried with addition of 0.1 in pixels values of processed image and were able to find same accuracy.

Justification

We checked in benchmarking section that even 10% accuracy should be highly significant for model than random guessing due to low probability and multiple digits. As we have got 63% accuracy validation our model justifies the benchmark.

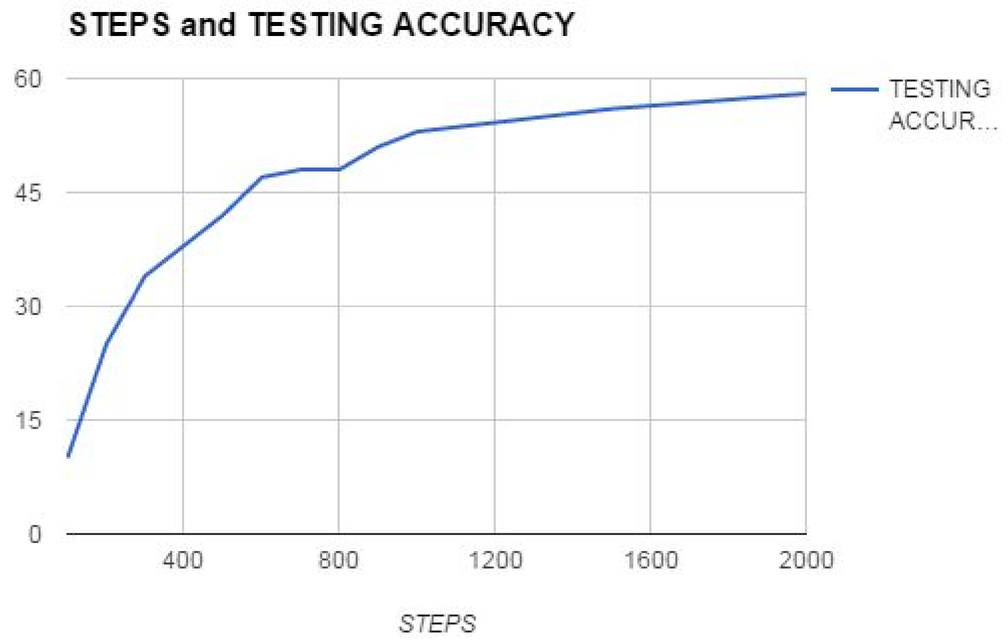
We can see that as we have correctly identified 8174 out of 13068 test images which is significantly above critical threshold in below distribution when considering random guess



V. Conclusion

Free-Form Visualization

Below plot depicts accuracy of model with number of steps. We see that with number of steps, our accuracy keep increasing till it converges to around 60%. Initial rate of increase in accuracy is high due to high learning rate.



Reflection

For creating application of number identification from images, we create a model using machine learning techniques on training it with SVHN dataset.

First we preprocessed the images to remove noise and standardize it before passing into model by following steps.
1) we find bounding box containing all digits in image so that we could remove all extra noise from image.
2) we crop the image into 32*32 size to standardize it
3) we convert the images to grayscale

As a model, we use deep neural network with four layers constituting 3 convolutional layers with RELU and max_pool units and fourth fully connected layer with 5 logit outputs corresponding to target digits in image. We trained the model using optimizer with cost as cross-entropy between target digits and model output.

Difficult aspects of our solutions were:-

- Removing noise and standardization of images before passing it to model
- Finding the correct hyper parameters of our model.

Interesting aspects of our solutions were:-

- Working on real and large dataset
- Our solution is general and could be applied on any image having digits and technique could itself be tried on other vision problems.

Improvement

We could try deeper convolutional layers so that we might be able to get better accuracy while incurring more computational cost. This could bring our model's accuracy closer to human level accuracy achieved by Google engineers.

We could also use local contrast normalization (LCN) technique but as reason for its use was not clear, we avoided it. We might try to use it after understanding concepts behind this technique.

We could also try making directly usable application of model like a camera APP in android.