Final Project Report for CS 175, Spring 2018

Project Title:

Convolution Neural Network of a Plate Number Recognition Classifier using SVHN Data.

Project Number: 4

Student Name(s)
Huy Duc Vo, 17903562, hdvo1@uci.edu
Khoa Thanh Anh Nguyen, 36594518, khoatn5@uci.edu
Khoa Nguyen, 43498533, khoan8@uci.edu

General Instructions:

· Your report should be 5 to 7 pages long in PDF

· If you want to add more details (e.g., additional graphs, examples of your system's output, etc) beyond the 7-page limit, feel free to add an Appendix to your report for additional results

· **What the Team submits to Canvas (one student submits the items below on behalf of the team)**

o Your report entitled **FinalReport.pdf**

o A zip file called **project.zip** that contains the following in 1 directory called project/

§ A README file that contains a 1 line description of each file in project/

§ A Jupyter notebook (called **project.ipynb**) that can be run directly and that demonstrates your project. Your notebook can import a sample of the data that you used, import 1 or more models that you built, and generate examples of the types of predictions or simulations your model can make. The notebook should not take any longer than 1 minute to run in total (if you have models that require a lot of training time, train them offline and just upload the models and some sample data to illustrate them). Feel free to generate examples of your model(s) in action, e.g., for reviews you could generate examples of reviews where the models work well and reviews where the models work poorly.

§ Also save a .html version of your notebook called project.html, showing the outputs of all the cells in the notebook.

§ Upload any data files needed to run project.ipynb – keep your data sets to 5MB in total or or less.

§ Also include a subdirectory called src (within the zipped project/ directory) with all of the individual code (scripts, modules) for Python (or equivalent for other languages) that your team wrote or adapted– these don't need to be called by the project.ipynb notebook but need to be in the src/ directory

§ Note that we don't necessarily plan to run all your code, but may want to look and run parts of it.

o **Individual Contribution**

Each team member needs to submit a ½ page of additional text as an individual, titled "**IndividualContribution_ID.pdf**" that provides an honest assessment of which parts of the project you contributed to and which parts were worked on jointly. This should be written individually – you may wish to discuss the plan of what you will write with your project partner, but the page you write should be generated separately by each individual.

1. Introduction and Problem Statement

In this project, we try to classify the single digits and multiple digits in an image and then we try to use a command line python with a image so that we can classify any image that we wants to classify the number on it. Assuming we have a robot and the robot needs to find the a room so that robot will take a lot of picture of room numbers. So the picture will be runned by the command line that we setup. Then the image will be passed into the a single Convolutional Neural Network model for localizing and classifying a sequence or a single of digit in the real world image. Convolution neural networks are layers of sets of neurons with parameters that filter the input data vector followed by an elementwise non-linearity.

The challenge when we create this classification model, the digits on the image become more complicated when we account for environment factors like lighting, shadows, specularities, and occlusions. When we change the resolution, motion, and focus blurs, it also affect the outcome of the classification.

2. Related Work:

The main article that we use for this project is Recognition from Street View Imagery using Deep Convolutional Neural Networks. According to the guide, Goodfellow recommends us to use deep convolutional network directly on the pixel of the image via localization, segmentation, and recognition. Then we will train the model via Tensorflow and analyze the result. The paper also introduce a new kind of output layer and conditional model of sequences.

According to Goodfellow, the math behind is letting S represent the output sequence and X represent the input image. Our goal is then to learn a model of P(S | X) by maximizing log P(S | X) on the training set. To model S, we define S as a collection of N random variables S1, . . . , SN representing the elements of the sequence and an additional random variable L representing the length of the sequence. We assume that the identities of the separate digits are independent from each other, so that the probability of a specific sequence s = s1, . . . , sn is given by

$$P(\mathbf{S} = \mathbf{s}|X) = P(L = n \mid X)\Pi_{i=1}^{n} P(S_i = s_i \mid X).$$

To train the model, one can maximize log P(S | X) on the training set using a generic method like stochastic gradient descent. Each of the softmax models (the model for L and each Si) can use exactly the same backprop learning rule as when training an isolated softmax layer, except that a digit classifier softmax model backprops nothing on examples for which that digit is not present. At test time, we predict

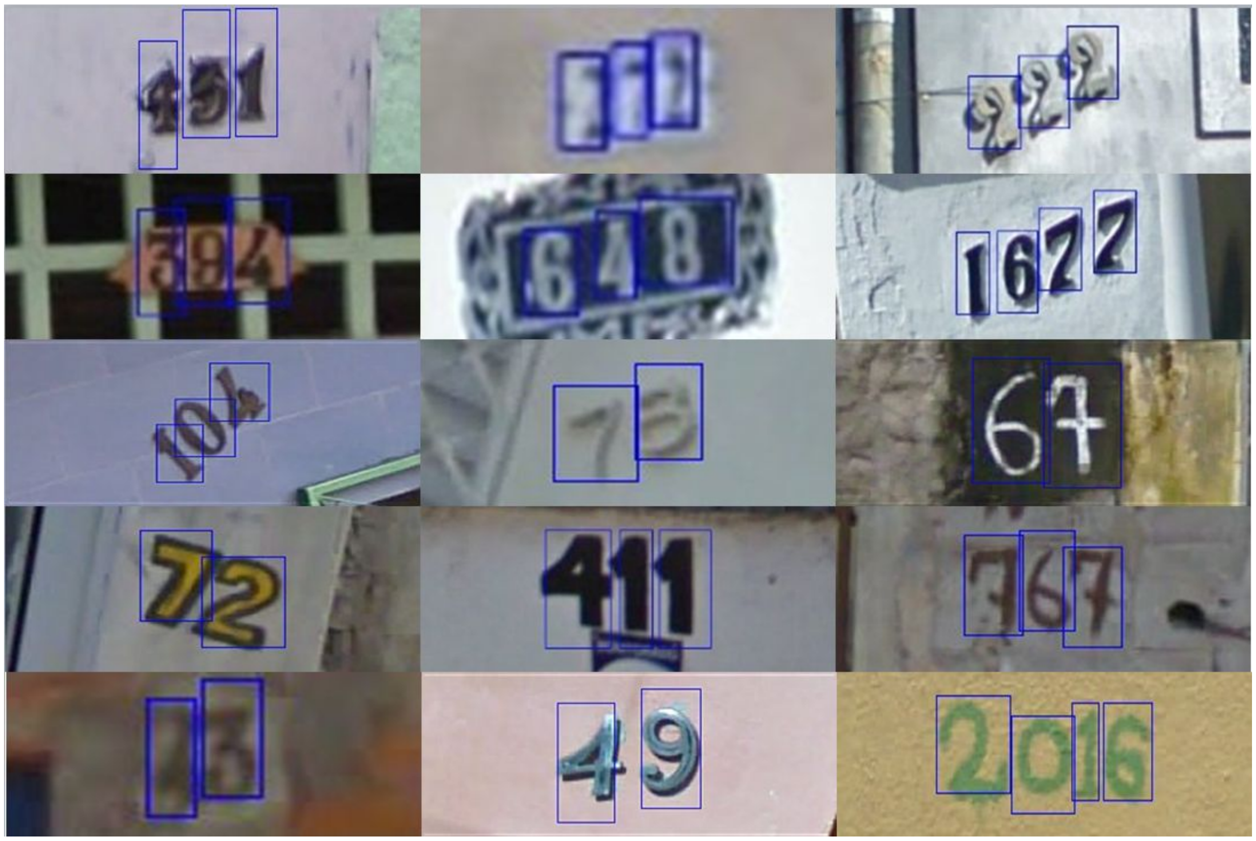
$$\mathbf{s} = (l, s_1, \ldots, s_l) = \operatorname{argmax}_{L, S_1, \ldots, S_L} \log P(S \mid X).$$

This argmax can be computed in linear time. The argmax for each character can be computed independently. We then incrementally add up the log probabilities for each character. For each length l, the complete log probability is given by this running sum of character log probabilities, plus log P(l | x). The total runtime is thus O(N).

Another article that help us define the challenges during designing the model is Reading Digits in Natural Images with Unsupervised Feature Learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011. Thanks for the article, we found out that classify the digits in the real world environment is harder than on paper. Netzer points out that lightning, shadows, specularities, and occlusions, resolution, motion and focus blurs affect the classification model.

3. Data Sets

For this assignment, we are going to use **The Street View House Numbers (SVHN) Dataset.** SVHN is a dataset that is composed of real data from house numbers in Google Street View images. Each sample image is 32x32. The dataset contains large amount of data, splitting between training and testing by 73257 and 26032 respectively, and an additional of 531131 less difficult samples. It has total of 10 labels, each label per digit (0-9).



References: http://ufldl.stanford.edu/housenumbers/

4. Description of Technical Approach [at least 1 page]

The model is developed using a Convolutional Neural Network. This approach emphasizes the idea of breaking down the input of an image using many layers of the network. Each layer has its own functionality in order to output a better form of data so that the next layer in the system can process, analyze, and output efficiently. For example, a layer in the network can handle the task of transforming an input into a 3D volumetric data with some differentiable functions.

What gives the name of Deep Learning is the layering system, in which each layer is composed of many neuron, much like a human brain, and each neuron is randomized with its own parameters for weight and bias. A neuron is also tasked with a job of performing dot products on inputs and then it uses a nonlinear activation function to follow up the result of the calculation. This allows lots of neurons and large models while keeping fairly small amount of parameters, and values.

In this project, our model is developed with this many layers, and each of them is described below.

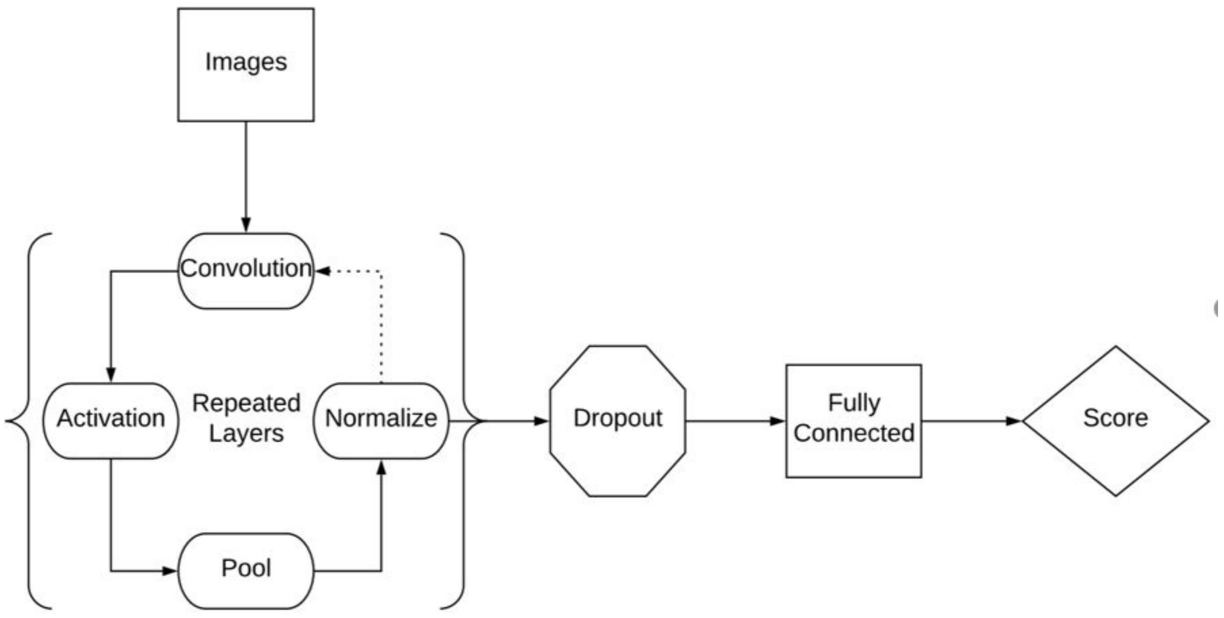


Figure. Convolution Neural Network Architecture.

Functionality of each layers:

- Convolution: 2D filtering image, perform dot products between filtered data and input. Filtered data then later used to create set of features at different layers of the data.

- Activation: any functions will be non-linearly approximate and transform into a range, -1 to 1 of 0 to 1, and tasked to detect nonlinear features in the data. This layer, in short, determines how the output should become activated.

- Pooling: is tasked to summarize and reduce the size of activations to limit the amount of parameters and computation in the network. This layer also control overfitting in setting the hyper parameters correctly.

- Dropout: this layer only works on training session to add protection against overfitting. It operates by keeping some neurons active. The amount of neurons being kept alive is randomized. The idea is to drop units from the network along with their connections randomly, preventing units from becoming familiarized too much (building a thinner training network).

- Fully-Connected: this layer is used to convert the output from previous layer into a vector, may have non-linear activation or a softmax activation to output probability of a class score.

- Output: This layer limits the size of input to the number of output classes. It usually has a softmax classifier to process when the output is one value per class.

4. Software [at least ½ a page]

- **Public-available resource:**
  Tensorflow, numpy, scipy, and ipython, python-tk, svhn (stanford)

- **Code will be written:**
  - Model.py: this script is the defined model of our Convolutional Network. It set up variables of weights and biases in each neuron.

  - Data_harvest.py: download all svhn data needed from the given url. This script takes in a path url such as `http://ufldl.stanford.edu/housenumbers/` to download the most updated dataset and then break them down into image arrays.

  - Single_digit_trainer.py: this script will train with training data that only contains single number. It uses tensorflow api to save and restore trainer model, and train that model with data produced from Data_harvest.py. It then produces a weight model of single_digit_model.

  - Multi_digit_trainer.py: Same as Single_digit_trainer. The only difference is that it will not be limited to just training on single digit data. It will also restore single_digit_trainer model and build up from there, then produces a weight model of multi_digit_model. Maximum number of digits is 5.

  - project.ipynb: This script will let user test out their accuracy of the model with real life images of their own choice. It requires an input of .png file as parameter.

5. Experiments and Evaluation

The metric that we used to measure the performance of our model was classification accuracy. To calculate classification accuracy, we divide total number of prediction by number of correct prediction. This measurement accounts for both true positives and true negatives. We regarded true positives and true negatives as equally weighted. When every element in the sequence is transcribed in correct order, we will obtain a correct prediction. For partial accuracy, we will give no evaluation because it is viewed as incorrect. For example, with digit sequence “2369” and prediction of “2366” is identified as false positive. We use formula below to calculate classification accuracy:

Accuracy = (Num of Correct Prediction / Num of All Prediction)*100

To set up our experiments:

1. Download and preprocess a range of images for training classifier model.
    a. To do this, we created a script called svhn_data.py to download and unpack images.
    b. Within svhn_data.py and digit_struct.py, we process our images into matrix vectors
2. Train classifier model for a single digit of well cropped 32px by 32px images
3. Save the weights of the convolution and pooling layer to reuse. The single digit classifier was trained using different hyper-parameters until the best one was found.
4. Reuse the convolution and pooling layer to train multi-digit sequence model.
5. Capture images found from around the house and neighborhood and feed into the multi-digit model for evaluation.
6. Develop and launch an android app to take pictures, crop, and feed into the multi-digit model for second evaluation.

Our early result was around 80% for single digit datasets and around 87% for multiple digit datasets. However, CNN provides us several parameters that can be adjusted to hopefully increase our model's performance. After cleaning up the architecture of our model and adjusting the model's hyper parameter several times, we able to achieve much better results with accuracy around 90% for both single digit and multiple digit. When refining our general architecture, we had some trials and errors to discover that number of convolution and pooling layers can be discovered. We also decided to add a step decay to reduce the learning rate at every epoch. Adjusting the hyper parameter is our next big step because it give the model insights and measurements on how much the current situation can affect the next step.

Hyper parameters:

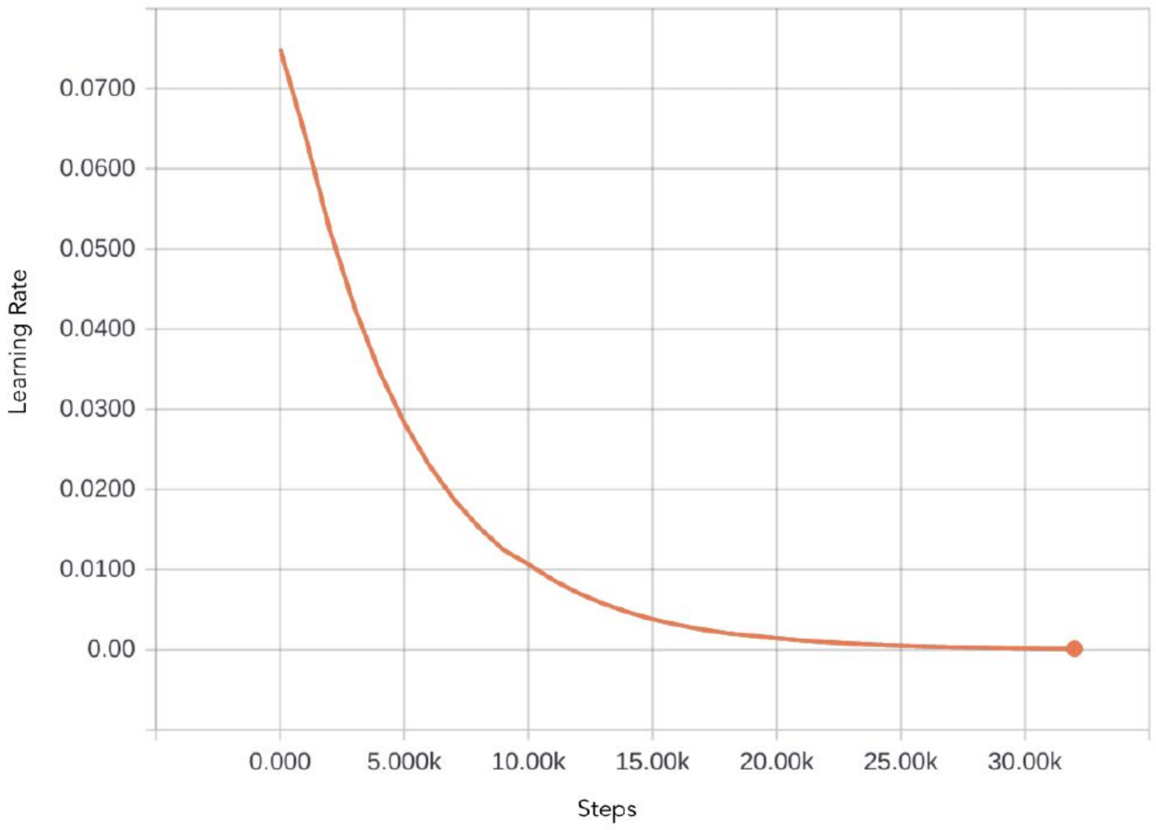
A. Neural Network Architecture
    a. Number of filters
    b. Shape of filter, depth of the output volume
    c. Types of pooling

B. Training Hyper-Parameters
    a. Number of Epochs - length of training
    b. Initial Learning Rate
    c. Decay Rate
    d. Dropout Rate

Measuring the performance of our model, we start out with 256 epochs or 70,000 steps. Later, we discover that with correct learning rate the model was converging closer to 128 epochs or 35,000 steps. The initial learning rate is set to 0.5 and staircased decay rate of 0.95. We found that this is the best rate after many trials and errors. Furthermore, we learned that If we set the value too high causes the classier to converge earlier but does not learn all of the intricate features. When we set this value too low, it also cause our model's the loss function to never converge. After the adjustment of the learning rate, I was able to increase the test accuracy about 2% to 4%.

A dropout layer with a 85% keep rate was removed from the multiple digit model because after some experiments, we found out that it affected our model performance and cost us 2% accuracy in testing. We think that it is because the dropout was reducing our training size while our dataset is large variance and overfitting did not even happen. In the end, the result that we get on single digit transcription with 128 epochs was 92% and on multiple digit with similar epochs was 91%.

Learning Rate



6. Discussion and Conclusion [at least ½ a page]

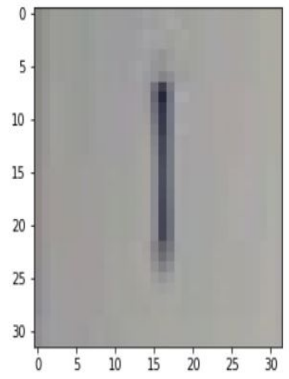
With the algorithms we worked with, we learned that we can generate a good results of decoding numbers from images taken in real world. However, there are some difficulties that we were not able to resolve. First of all, we could not recognize digits in images with 5 or more number. We understood that any images with 5 or more number will require more weight matrix. So when run on our android devices, it would require more memory that our devices do not have. Moreover, the classifier model is not flexible with unprocessed images. It requires specific dimension of 64 pixels by 64 pixels, and the center of sequence of digit must be in the middle of the image. These limitations are important when collecting our data because the classifier will not operate properly. If we were to be in charged of a research lab, we would focus on making the image recognition more flexible with accepting input images. In future direction, we would also want to create more memory in mobile devices to increase its capacity of storing and process more information. This would potentially make our digit recognition more powerful and useful.

Test Results

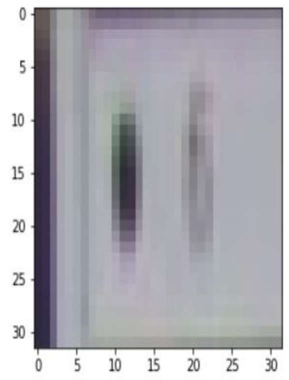
We test our model with test images and here are some of the results:
(Full result is in file **Executable.ipynb**)

For Single Digit Model:

```
==============================================================================================
==============================================================================================
==============================================================================================
==============================================================================================
Not using dropout
```



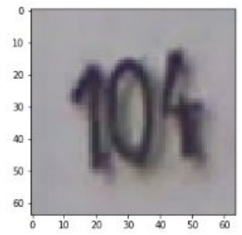
```
BEST PREDICTION IS: 1
==============================================================================================
==============================================================================================
==============================================================================================
==============================================================================================
Not using dropout
```



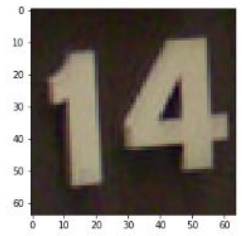
```
BEST PREDICTION IS: 1
```

For Multi Digit Model:

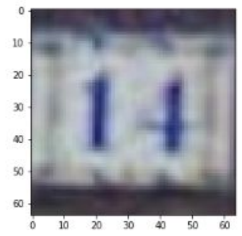
```
==================================================================================================
==================================================================================================
-------------------Attemped resize img------------------------------
```



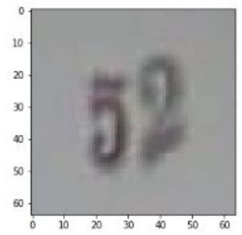
```
BEST PREDICTION IS: 104
==================================================================================================
==================================================================================================
==================================================================================================
==================================================================================================
-------------------Attemped resize img------------------------------
```



```
BEST PREDICTION IS: 14
==================================================================================================
==================================================================================================
==================================================================================================
==================================================================================================
-------------------Attemped resize img------------------------------
```



```
BEST PREDICTION IS: 14
==================================================================================================
==================================================================================================
==================================================================================================
==================================================================================================
-------------------Attemped resize img------------------------------
```



```
BEST PREDICTION IS: 31
```

Word cited

Goodfellow Ian J, Bulatov Yaroslav, Ibarz Julian, Arnold Sacha, Shet Vinay. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks Google Inc., Mountain View, CA.

Netzer Yuval, Wang Tao, Coates Adam, Bissacco Alessandro, Bo Wu, Ng Andrew Y.. Reading Digits in Natural Images with Unsupervised Feature Learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011. (PDF)