Classification and Detection with Convolutional Neural Networks on Street View House Number Dataset

CS6476 Computer Vision, Spring 2018
Xiangnan He
xhe321@gatech.edu

Abstract

Convolutional neural networks were used to classify and detect the Street View House Numbers dataset for this assignment. A VGG-like convolutional neural network layers were implemented using Keras with Tensorflow as backend. SVHN photographs (not cropped) and a house number video were classified by using the trained model. Two classifier were used to firstly detect if any number in the photos, with train accuracy of 97% and test accuracy of 96%, and then classify the numbers from 0 to 9 with train accuracy is around 93% with test accuracy at 92%.

I. Introduction

Detecting and recognizing multi-digit numbers from photos has been studied broadly by different teams and scholars, due to the importance of modern map-making, handwritten character recognition, car number plate recognition, and etc. The application is vastly importance, for example, it is frustrating and time consuming to be led to the wrong location due to a mislabeling of the map. A classic example is Google's street view photographs. Due to the amount of images to recognize, an automatic transcribing algorithm with high performance is needed to pinpoint the location of houses, stores, buildings, and etc. Recognizing numbers from photographs is part of the optical character recognition (OCR) community. Although given the extensive work already done on OCR, arbitrary sequence of numbers from noisy street view with all kinds of lighting, angles, rotations, and occlusions, and etc., are still quite challenging.

CNN (Fukushima, 1980; LeCun et al., 1998) are neural networks with clusters of neurons having different parameters. There are multiple filtering layers with each layer applying an affine transformation to the vector input followed by an elementwise non-linearity. The CNN method uses discrete convolution rather than a fully general matrix multiplication, resulting in better computational efficiency and scalability. Image CNN normally use a pooling layer that summarizes the activation of adjacent filters with a single response. Pooling layers summarizes the response of groups of units with a function like maximum, mean, or L2 norm. The pooling layers improve the robustness of the networks to small translations of the input.

II. Related work

Google (Goodfellow et al., 2014) has studied this based on their DistBelief network (Dean et al., 2012). They applied deep ConvNets with 11 weight layers to the task of SVHN recognition, and showed that the increased depth led to better performance. The ConcNets achieved over 96% accuracy in recognizing complete street numbers, and achieved 98% on a per-digit recognition task, which represents the state-of-the-art performance at that time. A more challenging dataset gernerated from street view imagery containing several tens of millions of street number annotations and achieved over 90% accuracy. The team also used image thresholding based on a confidence value returned from the system to achieve specific accuracy level, which was called confidence thresholding.

There are also other types of deep CNN architectures such as AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2014), and Oxford VGG (Simonyan & Zisserman, 2014). The assignment asked for comparison with VGG 16 with and without pre-trained weights, but due to my limited time working on this, this part was not finished.

III. Implemented CNN

Our work in this study focuses on using building convolutional neural network (CNN) with Tensorflow and training the CNN with the publicly available Street View House Numbers (SVHN). Keras was used for implementation with Tensorflow as the backend module in the project.

Preprocessing was carried out for the dataset by converting the photos to gray images, and preparing the X_train, Y_train, X_test, and Y_test data. The labeling of the data was from online json file, which converted the .mat file available in the repository.

A simpler a VGG like deep neural network was implemented with 4 Conv2D layers and 2 fully connected layers. VGG is a CNN model proposed by (Simonyan et al. 2014). Instead of binary cross-entropy, we used categorical cross entropy as the loss function, because it is a multi-class classification problem with more than two exclusive targets. The algorithm implemented here has two classifiers designed to firstly detect if there is any house number in the photograph (two classes: 0, 1), and then classify which digit is in the photograph (10 classes: 0~9).

Adadelta is a novel per-dimension learning rate method for gradient descent and was used for optimization in this study. The method dynamically adapts over time using only first order information and has minimal computational overhead beyond stochastic gradient descent (SGD). More specifically, it is an algorithm for gradient-based optimization that adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters. Due to this property, it is well-suited for dealing with sparse data. In addition, it seeks to aggressive, monotonically decreasing learning rate. Normally at the beginning, the learning rate is high, so that it can converge faster, and then the learning rate needs to decrease, so that the algorithm doesn't take big steps when it is close to the optima. Larger learning rate at the beginning is also good to move out of local optima, like stimulated annealing, and it can reach the global optima with better chance.

The gradient descent batch size was set at 128. Batch size defines the number of samples that going to be propagated through the network. Thus, the batch size was chosen because it uses less memory compared with loading much more samples at a time, this is critical because we cannot fit all the data at once into the available memory of the computer/laptop. Also the ConvNets update its weights after each batch operation. We didn't choose too small a batch size either due to that smaller batch size normal render less accurate estimate of the gradient or more fluctuation in the gradient directions.

Kernel/filter size used in the CNN was (3, 3), and the pool size was (2, 2) with max pooling method. Dropout value is 0.5, which was used to reduce overfitting in CNN by preventing complex co-adaptations on training data.

A softmax function was used as an activation function after the fully connected layers. Softmax activation in the final layer, and rectified linear unit (ReLU) was used in between the convolution layers before pooling. ReLU is the most popular activation function for deep neural network. Comparing with sigmoid activation, is a generalization of the logistic function that squashes a K-

dimensional vector z of arbitrary real values to a K-dimensional vector of real values in the range (0,1) that add up to 1. See below is the equation for the softmax calculation:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$ For j = 1, …,K

Running the full classifier through a sliding window and Gaussian pyramids are very expensive, therefore, MSER (maximally stable extremal region) algorithm was implemented and utilized for propose segmentation region for training. MSERs are components that are maximally stable, i.e., have a local minimum of the rate of change. Another way to understand this is thresholding. All the pixels below a given threshold are “black” and all those above or equal are “white”. Given an input image, if we generate a sequence of thresholded result images where each image corresponds to an increasing threshold, we would observe first a white image, then black spots corresponding to local intensity minima will appear then grow larger. The black spots will eventually merge until whole image is black. The set of all connected components in the sequence is the set of all extremal regions. The concept of MSER is linked to the one of component tree of images.

The Non-Maximum Suppression algorithm was used for selecting the optimal window for the individual numbers. Cross-validation was also used for improving the generalization performance and reduce overfitting.

We studied and plotted 15 epochs due to the long computational time. One epoch is when the entire dataset is passed forward and backward though the neural network only once. Since one epoch is too big to feed to the computer at a time, we divided it into batches. Less epoch leads to underfitting whereas too much epochs leads to overfiting.

IV. Results and Discussion

The SVHN data photographs were taken from both train and test sets to demonstrate the results of the training.

Figure 1. Correctly classified photographs.

As shown above, the CNN was able to detect and classify numbers with different variations, such as lighting, shadow, rotation, noise, scales, fonts, and etc.

Figure 2. Incorrectly classified SVHN photographs.

As shown above, the trained model failed to classify some of the SVHN photographs. The Non-Max-Suppressor threshold of the model can be adjusted to further optimize this. Some other way to improve the results are discussed in the following sessions.

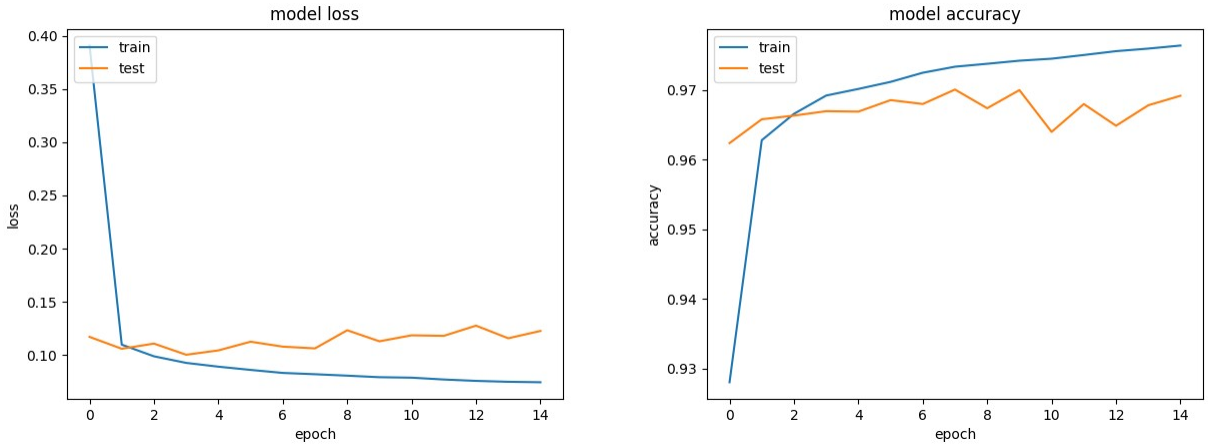

Figure 3. Detection (whether there is a number or not in the SVHN photograph) model loss and accuracy with growing epoch

Above are the model loss and model accuracy plot vs epoch. Total 15 epoch was collected. 80% data were used for training and 20% of the data were selected from the datasets for testing. As we can see, the train loss kept decreasing further away from 0.1 with more epochs, however, the test loss maintain higher than 0.1 and even shows a trend upward, which suggesting overfitting. On the right is the model accuracy, where the train accuracy keeps growing above 97%, however, the test accuracy remains between 96% and 97%, further epoch will not help due to overfitting.

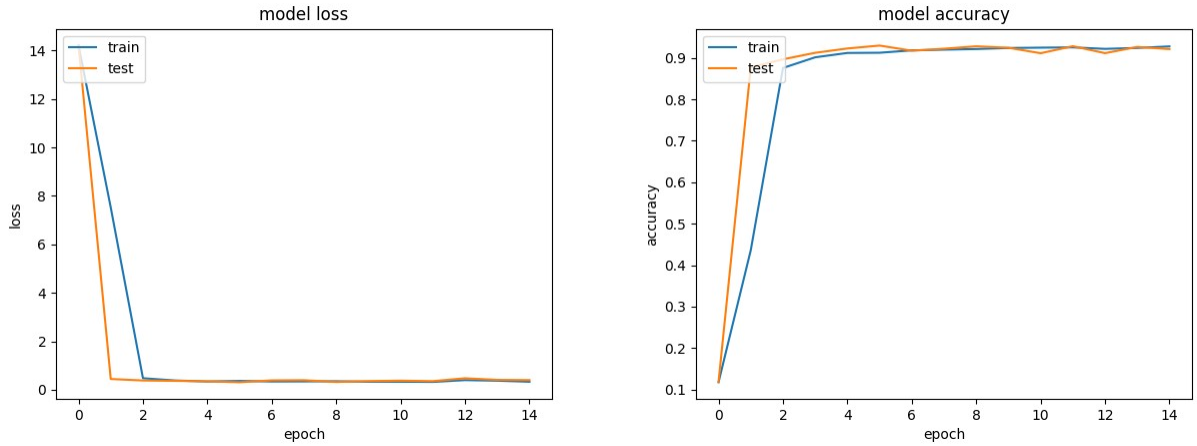

Figure 4. Classification (classify the numbers from 0 to 9 in the photos with numbers detected) model loss and accuracy

As shown above, loss and accuracy plots vs epoch for the classification model are shown. After three epochs (start counting from 0), the loss and accuracy start to converge to similar number and maintain the number even with more epochs. The train loss is slightly lower than test loss, which is expected, but the difference is very small, suggesting not so much overfitting. On the accuracy plot, test score is fluctuating up and down compared with train accuracy, and at 15 epoch, the test accuracy at 92% is slightly smaller than train accuracy at around 93%.

V. Video Output

The implemented ConvNets trained model was used for detection and classification of house number videos. The video taken with my cellphone comprises three videos of different building numbers. The first one has a building number of “2211” with “B” underneath it. The detection is good when we has the number nicely place without rotation, except that “B” was transcribed to “8” (limiting one sequence of number per image might help address this issue). Then, we can see when I rotate my cellphone, the “1”s changed to “7”, suggesting some augmentation with rotation might help reduce the mislabeling. The second one has a number of “2010” that is close to a “Researved Parking” sign, and the sign was misclassified to numbers. The third one has a number of “2090”, which is performing very well.

Opencv medianBlur function was used to smooth the images in the video to address noises in the images.

Figure 5. Screenshots from the video showing three different building numbers.

Link to my youtube demo video:
https://youtu.be/vZ0GCHLi9dg
Link to my youtube presentation video:

https://youtu.be/31zl20eFzLo

VI. Further Improvements

1) One special property of the SVHN is that they normally fall into certain length, N, which is mostly less than 5. Therefore, a improved model with an assumption that N <=5. This way, the systems will be able to identify when there are irrelevant features outside of the features, reducing false positives.

2) Another method for improvement is to use data augmentation to create more samples with different rotation, shift, zoom, shear, and etc. There is a function named “ImageDataGenerator” in “keras.preprocessing.image” that can be used for generate such augmentation images.

3) Deeper ConvNets can help improve the model performance, in this study, there is still room on the layers, since only 4 ConvNet layers and 2 fully connected layers were used. When the ConvNets grow deeper, some other methods

VII. Conclusions

VGG type of ConvNets were implemented to train the SVHN datasets for number detection and classification with train accuracy of 95% and test accuracy of 94%. Train and test accuracy, as well as train and test loss, were plotted along epoch number to analyze the performance of the CNN implemented. Photographs from the SVHN datasets were selected to look at the passing and failing numbers. The hypothesis and proposed improvements were also discussed to improve the model. A cellphone video of street view house numbers was labeled using the trained classifier to demonstrate the performance of the classifier.

VIII. References:

1) K Simonyan, A Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv preprint arXiv: 1409.1556
2) Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014b). Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint arXiv:1312.6082 v4.
3) Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large scale distributed deep networks. In NIPS’2012.
4) Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36, 193–202.
5) LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient based learning applied to document recognition. Proc. IEEE.
6) Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey. ImageNet classification with deep convolutional neural networks. In NIPS. 2012.
7) Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014.