Classification and Detection with Convolutional Neural Networks

CS6476 Computer Vision Final Project (Fall 2018)
Chong Duan
cduan34@gatech.edu

1. Introduction

Number detection and recognition in an unconstrained natural image is a very challenging problem. Broadly speaking, recognizing digits/numbers in an image is a problem of interest in Optical Character Recognition (OCR). Recently, convolution neural networks (CNN) (1) have seen huge successes in number detection and recognition, especially due to the increased availability of computational resources, large size of training data (2,3), and the use of dropout training (4).

Among many successful applications of CNN in object detection (5, 6, 7), Goodfellow et al. (7) introduces an interesting method to solve multi-digit detection problem completely using CNN and output an ordered sequence of detected digits. In the present work, I built upon the method introduced in Goodfellow et al. (7), and extended it to directly detect number within a natural image (instead of images whose major content are numbers as shown in Goodfellow et al.).

2. Methods

2.1 Data

The Street View House Number (SVHN) dataset were used for training neural networks in this project. Both format 1 (i.e., the original, variable resolution, and color house number images) and format 2 (i.e., cropped single digits with 32x32x3 image shape) were used.

In addition, a custom video (~10 seconds) of an apartment address number (my apartment in Fenway, Boston) was taken as the final testing input.

2.2 Preprocessing

SVHN Fromat 2: *First,* the .mat format data from the website were read into the workspace. For the original labels, 10 represents digit 0. This was first changed to use label 0 represents digit 0. *Second,* an additional 13188 32x32 images were randomly cropped from street view images downloaded from http://crcv.ucf.edu/data/GMCP_Geolocalization/. These images were used as negative controls. In other words, those images have label 10, indicating there is no digit within the images. *Third,* a balanced validation data set with equal distribution for each digit was created. *Finally,* the mean was subtracted from each image (i.e., all processed image has zero mean).

SVHN Format 1: *First,* the numbers in the full image were cropped out using the digitStructure.mat, and then resized to 32x32. The rest steps for processing were similar to Format 2 as described above.

Custom Video: To incorporate noise, location, scale, lighting, and other invariances, the camera was moved and rotated and zoomed when taking the video. Further, after the video has been taken, 10 continuous frames were edited to incorporate random Gaussian noise, and 10 frames were edited to have a much lower lighting condition.

2.3 Classification

2.2.1 Models

A total of four convolutional neural networks were implemented in this project. These models are summarized in Table 1. Model 1 and 2 are implemented based on the VGG16 structure.

Table 1. Summary of Neural Network Models Used in This Project. FC represents fully connected layer (i.e., dense layer), Conv represents convlutional layer, and Pool represents max pooling layers.

Name	Pre-trained weights	Structure
Model 1: VGG16	No	Input – VGG16 – Flatten – FC1 – FC2 - Output
Model 2: VGG16	Yes	Input – VGG16 – Flatten – FC1 – FC2 - Output
Model 3: NN-single-digit	No	Input-Conv1-Pool1-Conv2-Pool2-Dropout-Flatten-FC1-FC2-Output
Model 4: NN-multi-digit	No	Input-Conv1-Conv2-Dropout1-Conv3-Conv4-Dropout2-Conv5-Conv6-Conv7-Flatten-FC1-Dropout3-FC2-ParallelPredLayer-Output

For Model 1&2, the VGG16 API provided in Keras (tensorlfow.keras) was directly used. The only modification is the final output layer. The original 1000 classes was changed to 11 classes, which accounts for digit from 0 to 9, with an additional class for no digit. The only difference between Model 1 and Model 2 is that Model 2 loads the "imagenet" weights, while Model 1 does not load any weights (i.e., train from scratch). "Imagenet" weights were re-used because the Imagenet data has been widely used to build various classification models, and it demonstrates a strong ability to generalize to image outside the imagenet dataset.

Model 3 is a custom built neural network for classifying single digit. It does the same work as Model 1 and 2 do, but with a much simpler and shallower structure. It's architecture is given in Table 1. Note that dropout layer was included to reduce possible overfitting.

Model 4 is another custom built neural network for multi-digit number classification. The idea was taken from Goodfellow et al. (7), and it was implemented with tensorflow directly. First of all, it is assumed that there's no more than 5 digits in the number. The output of the model is an array of digits with length = 5. If there is no digit at a specific location, label 10 is used. For example, [8,1,3,10,10] represents the number 813. In terms of the model structure, more convolutional layers and dropout layers were used comparing to Model 3. More importantly, a parallel prediction layer was included after the second fully connected layer. The pseudo code is given below:

$for\ i \in 1:5$:

$$\{ \quad logits_i = DenseLayer(FC2)$$

$$y_{pred,i} = softmax(logits_i)$$

$$loss_i = CrossEntropy(y_{pred,i}, y_i) \quad \}$$

$$Loss = \sum_{i=1}^{5} loss_i$$

2.2.2 Training

For training the classifiers for digit detection in this project, categorical crossentropy was used for loss:

$$Loss = -\sum_{c=0}^{10} \sum_{i=0}^{n} y_{i,c} \log(p_{i,c})$$

in which c indicates class, n is the total number of training samples, $y_{i,c}$ is the label of training sample i, and $p_{i,c}$ *is predicted probability for class c for sample i. Given this loss, a stochastic gradient descent optimizer was used:*

$$\theta_{t+1} = \theta_t - \epsilon(t) * \frac{1}{B} \sum_{b=0}^{B-1} \frac{\partial Loss}{\partial \theta}$$

in which B is the batch size, θ is model weight, $\epsilon(t)$ is learning rate, which decays as training process moving forward. For most of training in this project, a batch size of 128, and learning rate of 0.001 (with decaying rate = 1e-6 and momentum = 0.9). Batch size and learning rate were selected based on trade off between speed and memory size (O(B) computation and O(B) memory). It's worthwhile to note that the choice of learning rate and batch size are functions of how curved the loss function is as pointed out by Goodfellow et al. (ref)

In regards to stop training criteria, an early stopping callback was used via the Keras API. Specifically, the criterion was set that if the validation loss does not improve over 3 epochs, the training will be stopped. The code is: tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, verbose=0, mode='auto').

2.4 Overall Number Detection Pipeline

The overall pipeline is shown in Figure 1. Top-left panel represents the input image. A sliding window with size = 32x32 and stride = 8 is used to slide through a pyramid of the input image and identify regions with digits using Model 2 as described in the Methods section. Normally, a large number of windows will be detected and seletected. A thresholding-based method was used to remove windows with label 10 (i.e., no digit) or low probability (i.e., the max of the softmax output is smaller than 0.8). This results are shown in the top-right panel. After that, the detected region with digits was consolidated into a larger window (bottom-left panel). Finally, the detected region with digits was passed into the Model 4, and the number (i.e., 813) was detected and shown in the bottom-right panel.

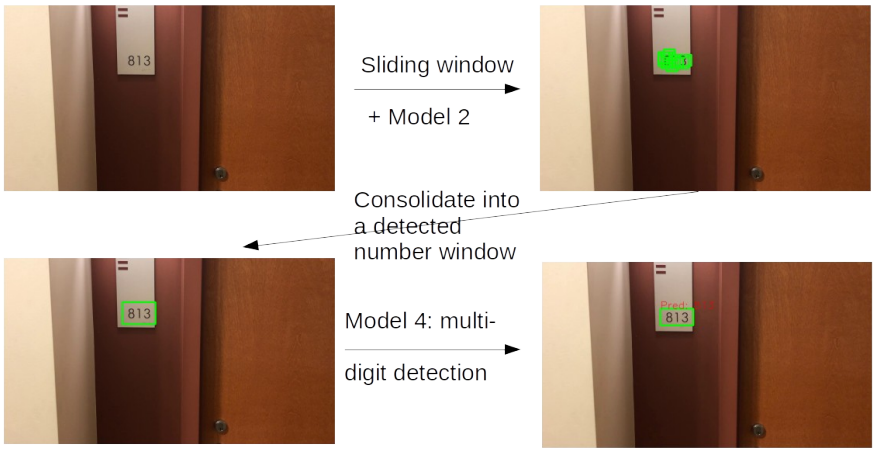

Figure 1. Flowchart for the overall pipeline

3. Results

3.1 Classification

3.1.1 Evaluating Performance

In this project, classification accuracy was used as a metric to evaluate the models. In order to evaluate the performance of the proposed models for classifying digits, training curves for both loss and accuracy were include in Figure 2. Note training curves for only the first model are shown here due to space limit. Training curves for Model 2 and 3 for single digit classification are very similar in shape. It's worthwhile to note that using pre-trained weights in Model 2 does make the model converges faster (i.e., with less epochs).

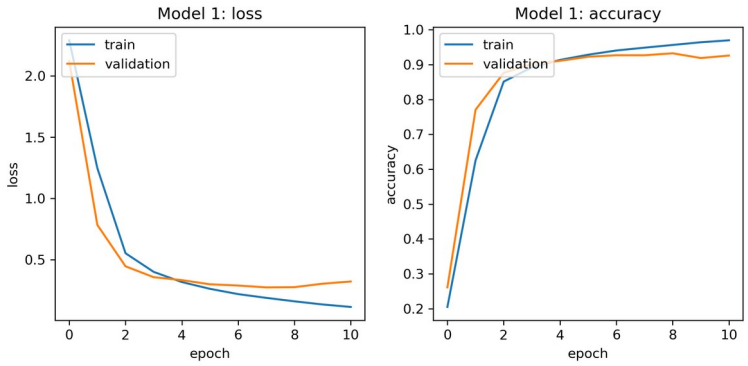

Figure 2. Training curves for loss (left) and accuracy (right) for Model 1. Model 2 and 3 are similar, but not shown here due to space limit.

The performances of the three classifiers as described in Methods were evaluated using the SVHN dataset (format 2, cropped single digits). The classification accuracy for each digit were given in Table 2.

Table 2. Classification Accuracy (%) in Testing Dataset for Each Digit (Note digit 10 indicates not a numer)

	0	1	2	3	4	5	6	7	8	9	10
Model 1	88.7	91.6	95.4	93.0	83.3	88.5	86.5	92.2	91.9	83.3	98.7
Model 2	95.8	96.6	95.4	92.2	95.8	91.0	93.8	96.3	93.2	92.2	97.4
Model 3	92.0	95.9	95.4	87.6	94.7	91.5	90.4	92.6	87.2	90.6	63.2

3.2 Video Results

As described in the Methods section, after training, the proposed network was employed to process a custom taken video containing address numbers. The final processed video can be found at https://youtu.be/PKWSysL56Ro or https://www.dropbox.com/s/it90v43a3jox1d3/video_out.avi?dl=0.

3.3 Image Classification Results

Five representative frames were selected from the processed video as described above. As shown in Figure 3, these frames demonstrates that the proposed network is able to detect and classify numbers correctly at different scales, orientations, lighting conditions, locations, and noise condition. In addition, some negative results were also shown in Figure 4. Those cased failed due to the false positive detection at the vertical lines along the door and wall (falsely detected as digit 1).

Figure 3. Results demonstrating the performance of the proposed network at different scales, locations, noises, lighting condition, and positions.

4. Discussion

In this project, a digit detection and recognition system was built using completely neural networks. This system takes an unconstrained natural image as input, and output an ordered list of the detected digits for a number.

Figure 4. Negative Results where the proposed system failed to correctly detect numbers.

Comparing to state-of-the-art work: When applied to the SVHN dataset with complete street number, this system achieves comparable testing accuracy (93.94% vs. 96%) comparing to the state-of-the-art method described in Goodfellow et al. (7).

Limits and future improvements: It's necessary to note that the system fails for some cases (Figure X). This is due to false positive at the segmentation/localization phase of the system. For example, the vertical lines along the door and wall were falsely detected as possible digit 1. More negative samples should be included in the training network in the future. Further, a robust suppression or clustering method could be included to exclude those false positive cases.

5. References

1. Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." *Competition and cooperation in neural nets*. Springer, Berlin, Heidelberg, 1982. 267-285.

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

3 http://crcv.ucf.edu/data/GMCP_Geolocalization/.

4. Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." *arXiv preprint arXiv:1207.0580* (2012).

5. Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014

6. Szegedy, Christian, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection." *Advances in neural information processing systems*. 2013.

7. Goodfellow, Ian J., et al. "Multi-digit number recognition from street view imagery using deep convolutional neural networks." *arXiv preprint arXiv:1312.6082* (2013).

8. Goodfellow, Ian, et al. *Deep learning*. Vol. 1. Cambridge: MIT press, 2016.

9. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).