Classification and Detection with Convolutional Neural Networks

Tianfang Xie, Georgia Institute of Technology

1 Introduction

Many traditional algorithms have been replaced by deep learning and machine learning methods. They play critical roles when dealing with target detection, image segmentation, image generation, and video understanding. Convolutional Neural Network (CNN) has been widely used for computer vision researches. Yann Lecun et al. [1] proposed a convolutional neural network algorithm based on gradient learning in 1989, and they successfully applied it to handwritten digit character recognition and achieved less than 1% errors.

Compared with other deep learning methods, the convolutional neural network has better potential in image or speech recognition, and this method needs to consider fewer parameters, which makes CNN become an attractive deep learning architecture [2]. The AlexNet [3] Convolutional Neural Network is the first convolutional neural network that has been widely used in computer vision, it is designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton, and it can use GPUs to greatly reduce the network training process time cost. The VGG [4] architecture has good generalization capabilities, and it has a pre-trained weight or model that can be used in various tasks. And the ResNet [5] architecture can further greatly reduce the cost of calculation and reduce the possibility of overfitting. The current development of convolutional neural networks is getting deeper and wider architectures, generating more branches, and using more effective methods.

2 Method of implementation

This work is about to use the convolutional neural network method to detect and recognize the digits in an image. The implementation method has three parts, first part is to use the OpenCV MSER algorithm [6] to get the return region of interest (ROI) in the image, the second part is to use a CNN method to decide whether these regions contain a digit or not, the third part is to use another CNN architecture to recognize the number of digits.

2.1 MSER algorithm and IoU guided non-maximum suppression

MSER has the adaptability of rotation, scale, and affine invariance, it uses a series of threshold parameters from range 0 to 255 to do the segmentation operation. As the threshold parameter keeps changing, a closed area will gradually appear. Finally, the area with the smallest change will be considered the most stable extreme area. This work used the OpenCV built-in MSER algorithm to do the ROI detection, the original building front door picture is shown on the top left side of Figure 1, the picture is from https://creativecaincabin.com/vinyl-house-numbers/, and the MSER detection results are shown on the top right side of Figure 1, each region is surrounded by a blue color bounding box.

After implementing the MSER algorithm, it is found that there are so many bounding boxes, to reduce the box numbers, two methods have been used. The first one is to compute the area occupancy rate [7] of each box and use a set of thresholds to delete the irrelevant boxes, the reduction results are shown in the bottom left side of Figure 1; the second method is to use the IoU (intersection over union) guided NMS method, this method can help to eliminate the suppression failure caused by the misleading classification confidences [8], it calculates the

overlap between two detection boxes and determines the boxes that need to be discarded. The final bounding boxes of ROI are shown on the bottom right side of Figure 1.

Fig 1. Original picture (top left), MSER detection ROI (top right), area occupancy rate reduction results (bottom left), and IoU guided NMS reduction results (bottom right)

2.2 A CNN architecture to classify the digit and non-digit ROI

In this part, a CNN architecture is used to detect whether the ROI contains a digit or not. The CNN architecture is similar to the LeNet introduced by LeCun et al [9], it has three sets of convolutional and pooling layers, followed by a flatten layer, then followed by two sets of fully connected layers and dropout layers, and finally a softmax classifier. The architecture is shown in Figure 2.

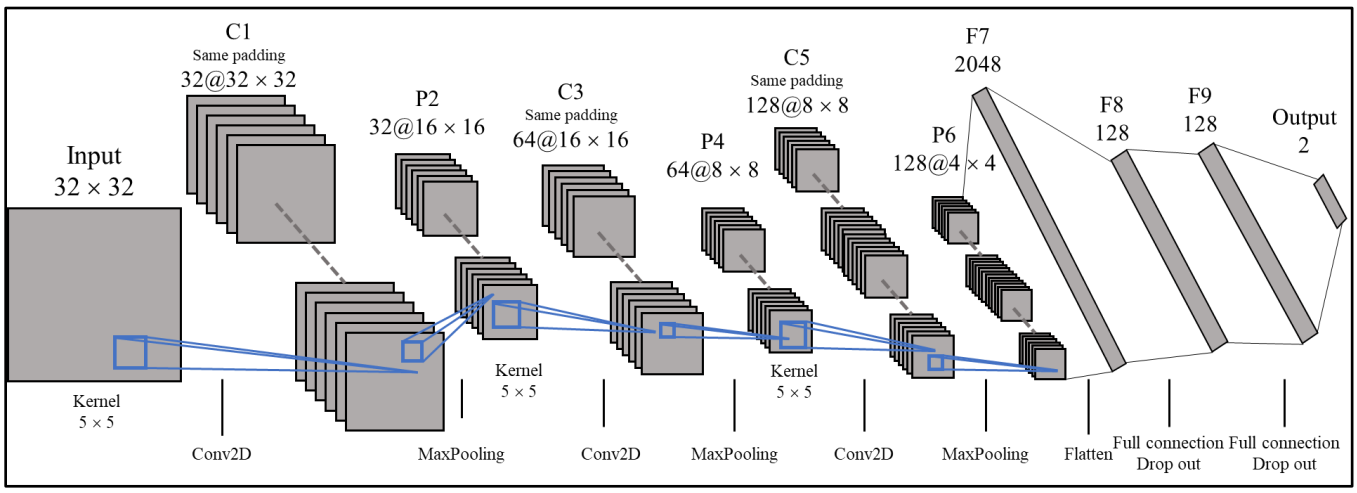

Fig 2. CNN architecture to classify digit and non-digit

The training dataset has two parts, first part is the cropped digits from the SVHN dataset [10], it contains 73,257 images which have been resized to a fixed resolution of 32-by-32 pixels; the second part is the self-generated non-digits dataset, 90 different street-view houses without front door numbers are cropped to generate 50,000 32-by-32 pixels non-digit images.

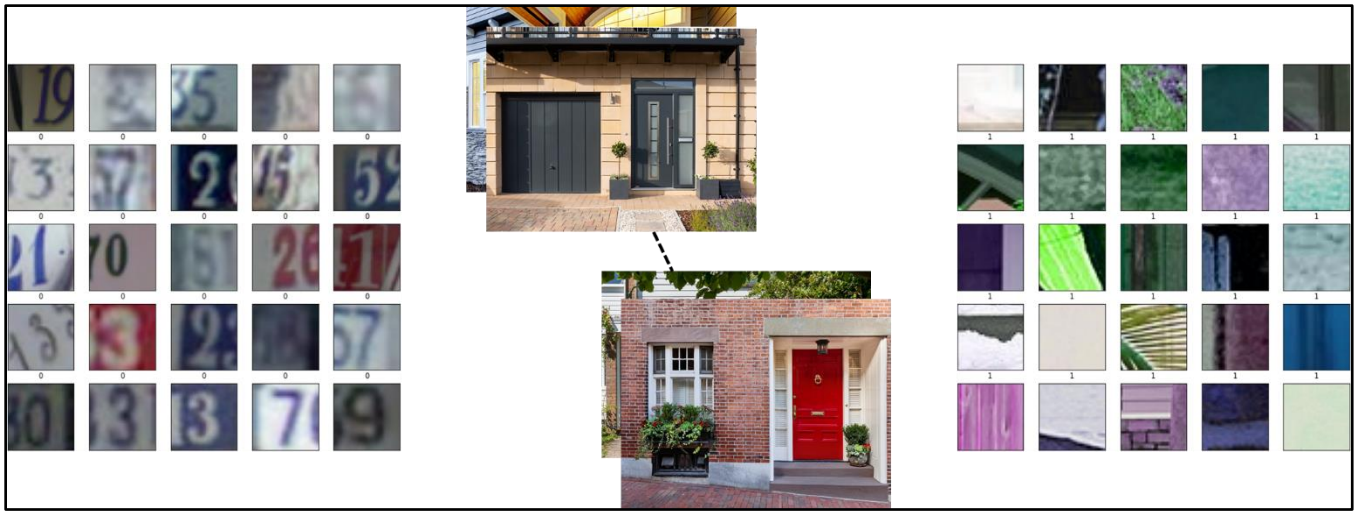

Fig 3. The digit dataset from SVHN (left side), the 90 input images to be used for the self-generated non-digit dataset (middle), the self-generated non-digit dataset (right side)

The hue values of images in these datasets are adjusted by a random factor, the final datasets are shuffled and shown in Figure 3, the digit images are labeled as "0", and the non-digit images are labeled as "1". These datasets are used to train the CNN architecture shown in Figure 2, and the trained CNN architecture is used to predict the ROI regions shown in Figure 1, the prediction results are shown in Figure 4, the CNN architecture can accurately identify whether the ROI contains digit or not.

Fig 4. The ROI regions bounding with blue boxes (left side), the CNN predict results of these regions (right side)

2.3 A CNN architecture to recognize the digit

In this part, another CNN architecture is used to recognize the digit from 0 to 9, it has a similar architecture as the CNN discussed in Part 2.2, it has three sets of convolutional and pooling layers, followed by a flatten layer, then followed by two sets of fully-connected layers and dropout layers, and finally a softmax classifier.

The training dataset is the cropped digits from SVHN dataset contains 73,257 images which have been resized to a fixed resolution of 32-by-32 pixels, the label of digit number zero is changed from "10" to "0" for convenience, the recognition results of digits in Figure 4 is shown in Figure 5.

Fig 5. CNN recognition results of digits (left side), final image output (right side)

3 Analysis of model performance and comparison of VGG

In this part, the choice of the loss function and the choice of batch size and learning rate will be discussed, and the evaluation of the model performance will be performed.

3.1 Choice of different parameters

For the choice of the loss function, the entropy is a measure of the uncertainty of random variables, and it is the expectation of the information generated by all possible events, the cross-entropy is used to describe the gap between the hypothetical distribution and the real distribution. The model gets better performance with a smaller value of the cross-entropy. The TensorFlow has serval loss functions, and the probabilistic loss functions are needed to be considered. The KLDivergence calculate the Kullback-Leibler divergence loss, the Poisson compute the Poisson loss, the BinaryCrossentropy is used when there are only two label classes, the CategoricalCrossentropy is used when there are two or more label classes and the labels are provided in an one_hot representation, the SparseCategoricalCrossentropy is used when there are two or more label classes and the labels are provided as integers. The loss function SparseCategoricalCrossentropy is used for this task, that is because the final results have 10 different classes, and the labels of the SVHN dataset are integers.

For the choice of batch size, the larger the batch size, the number of iterations required for convergence will decrease, but the calculation time of each iteration will also increase, and it will reduce the generalization ability and the performance of the model when larger than a critical value. On the other hand, batch size will also affect the optimal solution of the function, adjusting the batch size to a very small value can easily cause the network to not converge.

For the choice of the learning rate, the lower the learning rate, the slower the change speed of the loss function, and it is easy to cause overfitting. Although using a low learning rate can ensure

that any local minima value will not be missed, it also means that the converging time will become longer. In the meantime, if the learning rate is too high, gradient explosion will occur, and the model will become very difficult to get converge. Therefore, it is very important to choose an appropriate learning rate.

Overall, for this work, the loss function SparseCategoricalCrossentropy is used, the batch size is chosen as 32, and the learning rate is 5×10^{-4}, the plot of the historical training loss with a smooth decline shape confirms the validation of the choices of these parameters as shown in Figure 6.

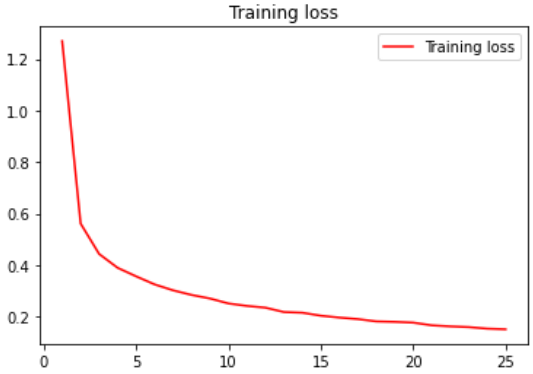

Fig 6. Training loss value of the self-generated CNN model

This task also used callbacks.ReduceLROnPlateau to optimize and decrease the value of learning rate during training, and used callbacks.EarlyStopping to monitor the value of loss and stop the training procedure when the change of the loss values becomes smaller than the threshold.

3.2 Comparison of pre-trained VGG16 model

A VGG16 model is implemented with pre-trained weights and compared with the previous self-generated CNN model to analyze the model performance. The pre-trained weights are chosen as ImageNet, which is a large visual database designed for use in visual object recognition software research with more than 14 million images [11]. All the layers of the VGG16 are set as un-trainable to prevent training over the pre-trained weights. A flatten layer, two sets of fully connected layers and dropout layers, and a softmax classifier are added to the pre-trained VGG16 model to get the results. The final VGG16 architecture is shown in Figure 7.

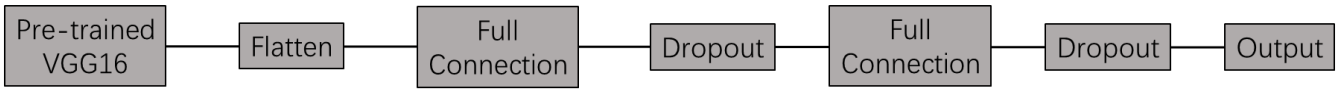

Fig 7. The comparison model used pre-trained VGG16

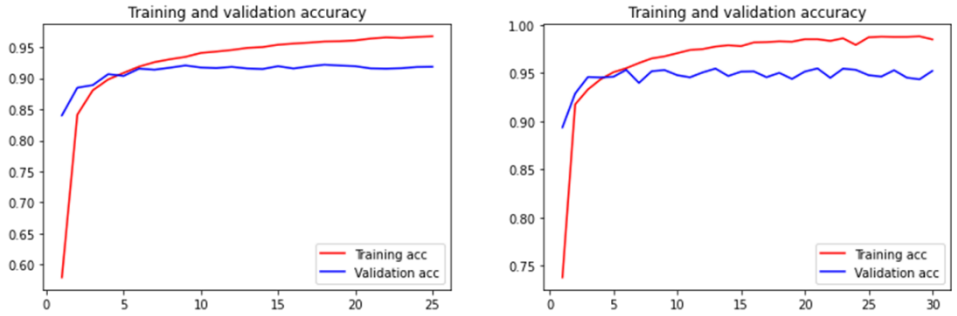

Fig 8. The accuracy of self-generated CNN (left side) and pre-trained VGG16 (right side)

The plots of training and validation accuracy and loss of the self-generated CNN and pre-trained VGG16 are shown in Figure 8. The self-generated CNN can get the training accuracy around 0.995 and the validation accuracy around 0.915, the pre-trained VGG16 can get the training accuracy around 0.991 and the validation accuracy around 0.951, both models have similar great accuracy when predicting the test data.

4 Final image classification results

The final results are shown in Figure 9, the detected digits are bounded with blue boxes and the red numbers are the predictions of classification. These plots prove the model have the ability for classification at different orientations (horizontal digitals in Subplot 1, and vertical digitals in Subplot 2), classification at different scales (small scales in Subplot 1 and 2, and large scales in Subplot 3, 4, and 5), classification with different lighting conditions (daylight conditions in the Subplot 1, 2, 3 and 5, and night light in Subplot 4 and 6), and the images have different digitals fonts and different digitals locations.

Subplot 5 shows the model can get a correct prediction of slanted numbers, and Subplot 6 is used to demonstrate that this model has some difficulties to predict the number "1", that is because some font types of the number "1" share the similar look as number "7" after been resized to 32-by-32 pixels. The sources of the original input images are cited in the README.md file.

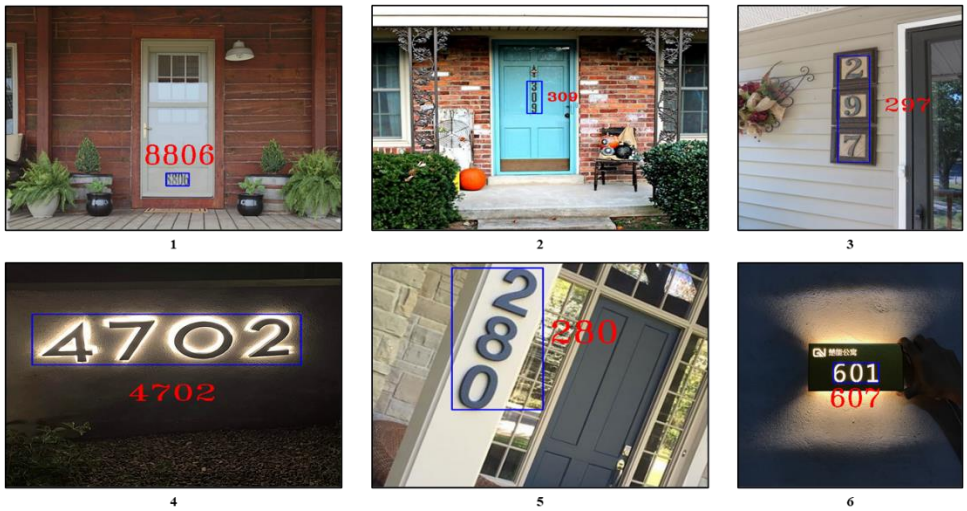

Fig 9. Final results

5 References

[1] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.Ciani, A., et al. "An experimental and numerical study of the structure and stability of laminar opposed-jet flows." Computers & fluids 39.1 (2010): 114-124.

[2] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] https://docs.opencv.org/4.1.0/d3/d28/classcv_1_1MSER.html

[7] https://nayan.co/blog/AI/Text-detection-in-number-plates/

[8] https://paperswithcode.com/method/iou-guided-nms

[9] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[10] 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.

[11] https://en.wikipedia.org/wiki/ImageNet