Final Project: Classification and Detection with Convolutional Neural Networks

Hyesung Ji (hji61@gatech.edu)

***Abstract*—In modern society, detecting and classifying digits from a given image is a widely used and crucial technique. This technique enables people to do their work automatically such as finding out an address or identifying a number. Normally, SVHN (Street View House Number) dataset is widely used for the detection and classification of digits. In this project, simplified and pretrained VG models and a custom designed CNN model are implemented and compared their performances in terms of loss function, classification accuracy, and runtime. Also, MSER and NMS algorithms are applied to detect digits from a given image. Finally, the detection and classification techniques are combined and implemented with some arbitrary images which are randomly downloaded from Google. Some good cases and bad cases will be shown and discussed the pros and cons of the suggested approach in this project.**

I. INTRODUCTION

THIS project aims to detect and classify digits from a given image. To detect digits MSER (Maximally Stable Extremal Regions) and NMS algorithm were utilized. Section II describes the detection method in detail, including how the parameters for the MSER algorithm were determined. To classify the detected digits, CNNs (Convolutional Neural Networks) are well-known and effective algorithms. Among them, VGG-16 algorithm is particularly popular due to its robustness and high performance. However, this model requires significant computational resources. On the other hand, a simpler model can also be used to classify digits from an image. Even if such a simplified algorithm may be less accurate than the original VGG-16, it can greatly reduce computational cost. Therefore, in this project, a simplified CNN algorithm and a simplified VGG-16 architecture are introduced and explained extensively in section III. To compare the performance between a simplified VGG16, pre-trained VGG-16, and a simplified CNN model, these accuracies and runtime are presented in section IV. In addition, real-world digit images from Google are used to test the detection and classification pipeline, and the results are also discussed in section IV.

II. DETECTION METHODOLOGY

To classify digits from an image, it is necessary to detect the digit. Without the detection of digits, classification cannot be implemented. Considering the digits in an image, the digits should be noticeable easily. So, digits have different colors compared to the surrounding background. It means that digits usually have boundaries and consist of the same color. So, it can be detected by MSER algorithm. In case of significantly smaller image, image pyramid can magnify the image and digits can be detected. Also, if there are a lot of detections and overlaps in the same digit, eliminating the duplicates would be necessary. So, the NMS algorithm is used in this project.

A. MSER and image pyramid

In this project, the MSER and image pyramid are combined. The image pyramid is applied to the scales 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, and 2.0, respectively. The process is the following. An image is loaded by applying image pyramid with a certain scale and convert it into gray scale. After that, a 3 × 3 Gaussian filter is applied to the gray-scale image. The MSER algorithm finds interesting regions. At this point, if the region is too small or the height-to-width ratio is abnormal, those regions are eliminated. Also, contrast can be an important factor in digit detection. Normally, digits in an image have distinct intensity differences compared to their surrounding background. So, computing the standard deviation of a local image patch can show an effective approximation of local contrast. Finally, the location and size of the region is restored by its scale and stored in the box-list.

B. NMS

Applying MSER to several sizes of image results in a lot of regions. This can detect an interesting region effectively, but a lot of overlaps will be created. Eliminating the duplicates and overlaps is also a necessary process in detection algorithm. This computes IoU (Intersection over Union) and if the IoU is larger than a certain threshold, the box is eliminated. IoU is computed

as the ratio of the intersection area to the union area. In this project, the threshold is 10%, so if a box (region) is overlapped with the previous box more than 10%, the box is eliminated.

III. Classification Methodology

In this section, a simplified VGG-16, a simplified pre-trained VGG-16, and a custom-designed CNN were employed. To implement the complex VGG-16 architecture on a local machine, a simplified version with only three blocks was used. Each block includes convolutional layers, ReLU activation, and a max-pooling layer. The simplified pre-trained VGG-16 also includes three such blocks and uses pre-trained weights from ImageNet. Also, a non-digit class is included to detect and classify digits more effectively. The original dataset consists of 10 digits (i.e., classes 0 through 9), some background images and their bound boxes are additionally included in the dataset. Therefore, these models have 11 classes to classify 10 digits classes and non-digit class. The details of each architecture are provided below.

A. A simplified VGG-16

VGG-16 algorithm is very efficient and well-known neural network architecture. It consists of five convolutional blocks and three fully connected layers. Each block has multiple convolutional layers with 3 × 3 kernel and the number of convolutional filters increased gradually like 64, 128, 256, and 512 filters. Also, at the end of each convolutional layer, RELU function is used and at the end of each block max-pooling is used. The reason for using 3 × 3 kernel is to represent more complex features of a given image by using more non-linear layer and to reduce the number of parameters [1]. The diagram is the following [1].

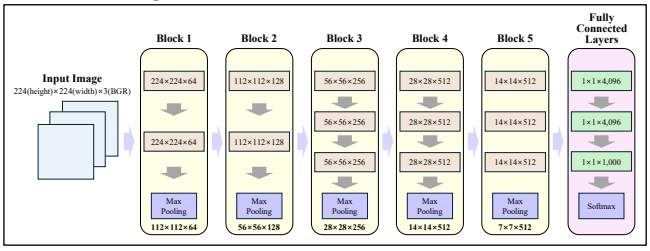

Fig. 1. VGG-16 Model

In this project, only three blocks were used. First, due to limited memory and computing power, the input size could not be as large as the suggested size (224 × 224). On a local machine, the optimal input size is usually 32 × 32. Second, if all blocks are used on a local machine, the final output of the last block becomes 1 × 1 × 512, which may lose important spatial information from the original image. Therefore, considering the hardware limitations and need to preserve the spatial information, three blocks were used.

B. A simplified pre-trained VGG-16

Due to the computational cost of VGG-16 architecture, the torchvision library in Pytorch provides pre-trained weights trained on the ImageNet 1K dataset. As stated above, the dimension of input images is 32 × 32. So, three blocks are used in this project and the architecture follows the same shape as simplified VGG-16 model described in subsection A. The convolutional layers are kept frozen, and only the fully connected layers are updated during training.

C. A custom designed CNN

As stated in introduction and subsection A, computational cost is one of the most important factors. The purpose of this project is to classify digits from an input image. So, if a smaller convolutional layer and light architecture model is able to classify digits effectively, there is no reason to use a heavy architecture model to classify digits. Currently, the architecture of VGG-16 is too heavy to run it on a local machine. So, considering the dimensions of the input image, a significantly light architecture is designed. The diagram is below.

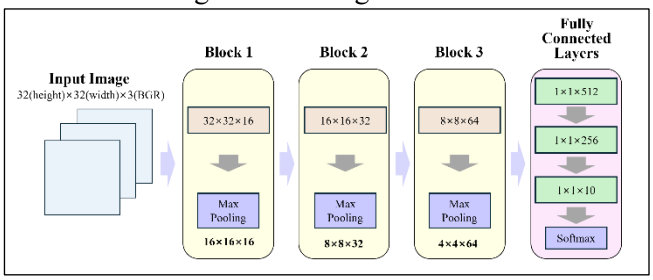

Fig. 2. Custom designed CNN

Each block consists of a convolutional layer followed by a max-pooling layer, which reduces the spatial dimensions by half. This model has fully connected layers and the numbers of neurons are 512, 256, and 11 respectively.

Also, the distribution of digits is not even. In the training dataset, 0' digits have a high frequency of occurrence. In contrast, 8' and '9' digits appear relatively infrequently. So, this indicates that the dataset has a class imbalance. The distribution is the following.

Digits	0	1	2	3	4	5	6	7	8	9
Occurrence	4,948	13,861	10,585	8,497	7,458	6,882	5,727	5,595	5,045	4,659

Table 1. The number of each class is in training set.

To overcome the class imbalance in table 1, "focal loss" technique is introduced in this project. Focal-loss technique is to focus on misclassified examples by reducing the loss value for well-classified examples [2]. Normally, the cross-entropy function is the below.

$CE(p_i) = -\log(p_t)$, p_t is a predictive probability for answer class.

In the focal-loss technique, the factor gamma is introduced to reduce well-classified examples, and it makes the model concentrate more on the infrequent samples rather than the frequent samples [2].

$$L_{focal} = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

In this formula, α_t is a weighting factor for class t and γ is a focusing parameter.

D. Train process

Considering machine-learning or deep-learning research, the training process may lead to overfitting. Therefore, preventing overfitting is a critical aspect of model development. In this project, the original training dataset is split into a training set and a validation set, and the validation set is 20% of the total data. The model's performance is evaluated using this validation set. When a model overfits, it typically shows decreasing training loss while validation loss increases [3]. Due to the capability of the local machine, the number of training epochs is limited to 20. To mitigate overfitting, early stopping is introduced based on the validation loss [4]. Specifically, if the validation loss does not improve for three consecutive epochs, the training process is automatically terminated. Since validation loss is a more significant indicator of model generalization performance than training loss, it is used as the primary criterion for early stopping [5].

IV. Experimental Results

In this section, two experiments results are shown in this section. The first experiment is the classification experiment. As discussed in the above section, three models are implemented and evaluated. All models are trained using the Adam optimizer with learning rates of 0.001, 0.0005, and 0.00001, respectively. Each experiment is run for 20 epochs, considering the computational limitations of the local machine. The second experiment is detection. The detection algorithm is applied to downloaded images from Google. Finally, two experiments are combined, and five images are shown in case of good examples, also the other five images are shown as a bad example.

A. Results of classification experiments

1) A simplified VGG-16

As described in section III, a simplified VGG-16 with three blocks is implemented on a local machine. In this case, cross-entropy is used to evaluate the model's training performance. According to the learning curves, this model fails to train properly when a learning rate is greater than 0.0001, as the loss does not decrease as shown in Fig 3 and 4. So, the model with a learning rate of 0.0001 has the best training results among the cases.

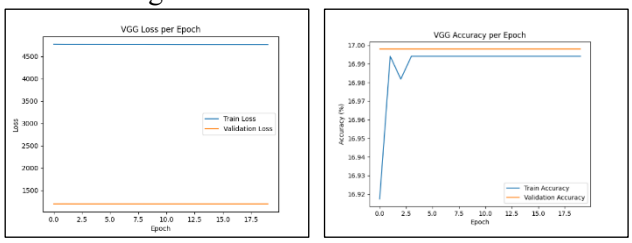

Fig. 3. Learning and accuracy curve at learning rate=0.001

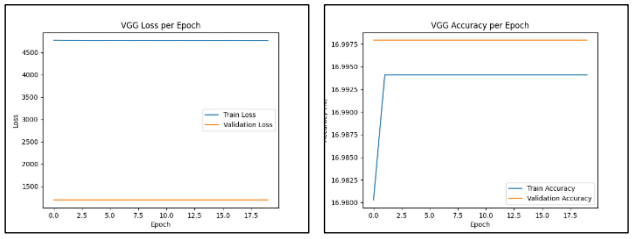

Fig. 4. Learning and accuracy curve at learning rate=0.0005

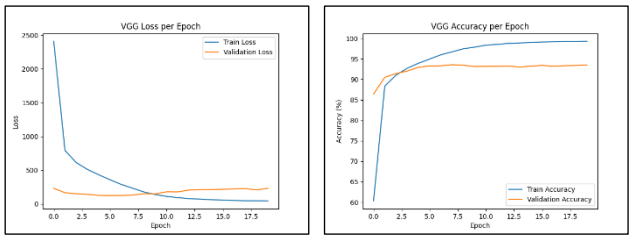

Fig. 5. Learning and accuracy curve at learning rate=0.0001

2) The pretrained VGG-16

This model has the same process as the simplified VGG-16 model. In this case, when a learning rate is greater than 0.0001, the model trains well, but it does not converge properly. Similarly, the model has the best performance with a learning rate of 0.0001.

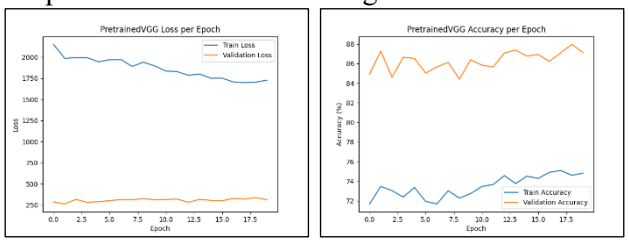

Fig. 6. Learning and accuracy curve at learning rate = 0.001

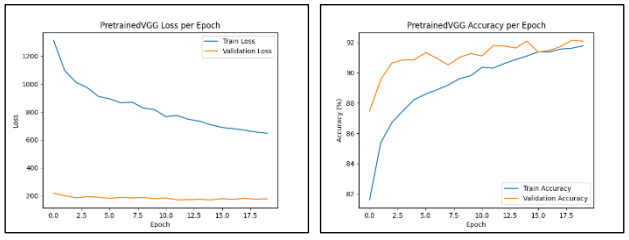

Fig. 7. Learning and accuracy curve at learning rate=0.0005

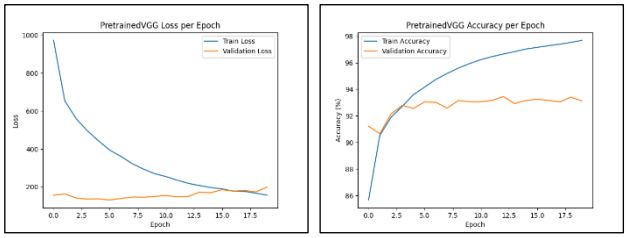

Fig. 8. Learning and accuracy curve at learning rate=0.0001

3) A custom designed CNN

As described in section III, this model uses 'focal-loss' is used to evaluate the model's training performance. In this case, when a learning rate is 0.001 and 0.0005, the model trains successfully, but when a learning rate is 0.0001, the underfitting is seen in Fig 11. In this model, the best performance is shown at learning rate 0.0005.

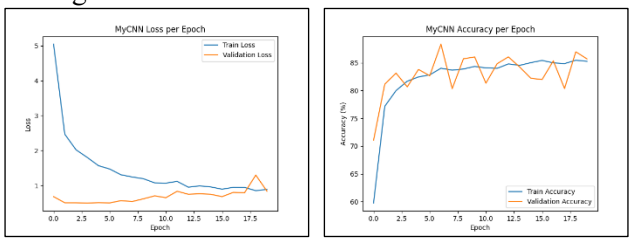

Fig. 9. Learning and accuracy curve at learning rate=0.001

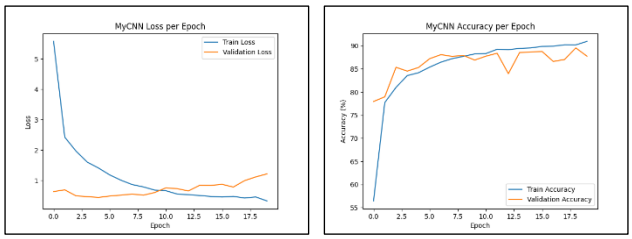

Fig. 10. Learning and accuracy curve at learning rate=0.0005

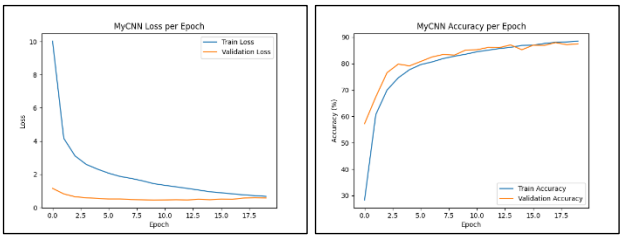

Fig. 11. Learning and accuracy curve at learning rate=0.0001

4) Overall performance

In terms of loss, accuracy, and runtime, this performance of each model is described below. As described in section III, all models have the identical early stopping condition and same number of epochs. The performances are measured from training, validation, and test dataset of SVHN.

Model	Best accuracy			Best loss			Runtime
	Train	Val	Test	Train	Val	Test	
Simplified VGG-16	99.28 %	93.49 %	92.79 %	229.49	232.11	229.49	15,888 sec
Pretrained VGG-16	97.69 %	93.12 %	92.95 %	156.47	198.75	239.52	10.539 sec
Custom CNN	88.36 %	88.26 %	81.66 %	0.506	1.018	0.89	657 sec

Table 2. Accuracy, loss, and runtime of each model

As shown in table 2, even if the custom CNN has the lowest accuracy, it has the best loss value. This means that the custom CNN model trains properly and it has an opportunity to be improved. Also, it can significantly reduce runtime compared to VGG-16 models. So, if a faster model is necessary, the custom CNN model could be an alternative option instead of VGG-16 models.

B. Results of detection and classification experiments

As discussed in section I, detection and classification should be implemented simultaneously to recognize digits from a given image. Five images representing successful cases and three images representing failure cases are selected and shown below. Among the successful cases, the detection algorithm detects not only digits but also detects unnecessary parts of an image. In these situations, the classification algorithm filters out the unnecessary parts by applying a threshold criterion. If the maximum classification probability of cropped image is greater than 50%, the cropped image classifies as a digit. Otherwise, the cropped image classifies as a non-digit and is removed from the cropped image set. Successful cases are shown on Fig. 12 to 16, and failure cases are shown on Fig. 17 to 19. The custom combined model is faster model, but it is not very accurate, so it sometimes results in failures and misclassifications.

1) Successful cases

Fig 12. Number: 7419 / Detected and Classified: 7419

Fig 13. Number: 302 / Detected and Classified: 302

Fig 14. Number: 395 / Detected and Classified: 395

Fig 15. Number: 56 / Detected and Classified: 56

Fig 16. Number: 5750 / Detected and Classified: 5750

2) Failure cases

Fig 17. Detection failure

Fig 18. Misclassification

Fig 19. Misclassification and detection failure

V. Discussion

The images used in good cases are very clear and have high contrast. To be specific, all digits have different colors compared to the surrounding background. In terms of detection, if the digit is too small or blurred, the detection possibility might be lower. Also, the intensity difference between the digit and the background is an important factor. In terms of classification, some cropped regions containing only background were incorrectly classified as a digit. Considering the bad cases, a dark object and white background might result in misclassification.

Another reason for the bad cases is a dataset issue. The model was trained on SVHN, and the properties of tested images are significantly different from SVHN images in terms of resolution, dimensions, and contrast. If some similar images have been included in training sets, the results might be accurate. The most fundamental reason is the architecture of the detection algorithm and custom CNN. Even if the detection algorithm used MSER, image pyramid, and NMS, the selection of parameters is not analytic. The parameters such as delta, min area, and max area in MSER were heuristically tuned based on visual inspection of the images without quantitative validation or gird search. Also, image pyramid and NMS are implemented based on visual heuristics. If the detection algorithm is more intricate, the detection rate could significantly improve. Also, the custom CNN algorithm uses just three convolutional layers and three fully connected layers. Additionally, the number of parameters in MyCNN is just 153,803. On the other hand, the number of parameters in the original VGG-16 is approximately 138 million. Therefore, MyCNN is 897 times lighter than the original VGG-16. This means that the model is significantly smaller, and its simplicity may contribute to misclassification, particularly in challenging or visually ambiguous cases.

VI. Conclusion and Future work

The purpose of this project is to build a custom dectection and classification algorithm from scratch. Even if there are some cutting-edge algorithms to detect and classify digits simultaneously, it can be a very valuable experience for me to study the characteristics of images and architecture of CNN models. To detect digits more effectively, a CNN model dedicated to detection can be a solution. In the future, I would like to have an opportunity to research and tune recent techniques such as YOLO, RetinaNet, or DETR algorithm.

References

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

[2] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” arXiv preprint arXiv:1708.02002, Aug. 2017.

[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[4] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” arXiv preprint arXiv:1611.03530, Nov. 2016.

[5] T. Ishida, I. Yamane, T. Sakai, G. Niu, and M. Sugiyama, “Do we need zero training loss after achieving zero training error?” arXiv preprint arXiv:2002.08709, Feb. 2020.