Street Number Classification and Detection

Junyuan Tan
jtan319@gatech.edu

Abstract

This report is about a street number classification and detection task using the famous Street View House Numbers (SVHN) dataset [1]. The dataset contains digits extracted from Google Street View for street numbers in front of real houses. A comparison is done between the VGG-16 model [2] with pre-trained weights and a simplified ResNet model [3].

Algorithm

Our detection algorithm works by first running MSER algorithm through the input images and pre-filtering the results, and then we produce image cutouts with the MSER bounding boxes and run those image segments through our CNN model trained using the below techniques for digit classification.

Transfer learning. The classification task is implemented using a technique called transfer learning. Machine learning relies heavily on the amount of data that the model trained on to improve accuracy. Our SVHN dataset has about 70,000 images, however, this is not enough as those images are extracted in a similar scene or settings—Google Street View. If the test image is not a number and it is extracted somewhere else, the model will no be able to recognize it. What transfer learning does is to use pre-trained models trained using thousands or millions of images from various sources. These images represent various classes of objects, and the resulting model can recognize more broadly in terms of object classes and less likely to make mistakes.

VGG-16 model [2]**.** We chose VGG-16 model with pre-trained weights as our basis for transfer learning. The model consists of 16 layers made of 3x3 convolution layers, max pooling layers and fully connected layers. We modified the last fully connected layer to output 11 classes instead of the default 1,000 classes. These 11 classes represent numbers from 0 to 10, and also a "not a number" class. The weights in this VGG-16 model are not modified except for the last layer which we modified. In that case, the weights are reset to random numbers between 0~0.01. We do this because this leverages the powerful information in the first few convolution layers trained using large amount of data while also fit the model to our problem to produce 11 classes. The convolution layers contain knowledge about information extracted from the image, and the fully connected layers are for decision making like "which class does the information in convolution layers represent".

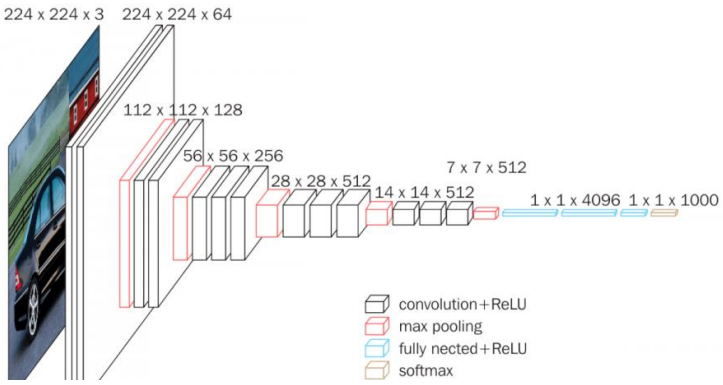

Figure 1: Layers in VGG-16 model

Filtering using Maximally Stable Extremal Region (MSER) [4]. The CNN model is used for classification of images only and we are missing parts about locating the digits in images so far. The MSER represents a region that has continuous transformation of image coordinates and monotonic transformation of image intensities [4]. This makes it an ideal algorithm for text detection in images since text usually have similar intensities around the text area in images. Once we apply MSER algorithm, we get preliminary locations of where the text could be in an image, then we supply those preliminary regions to the CNN classifier to detect the class of numbers they represent. Note that the digits in images may have different scale, larger or smaller, we need to perform hyper parameter tuning for MSER function. In this case, we scale the images input to MSER function to have a height of 250 pixels and set the MSER function parameter to detect areas of minimum size 100 and maximum size 2,000. There are some filtering we can also perform based on the domain knowledge of digits. For example, the bounding boxes of an actual digit is close to a square-like rectangle. If the MSER algorithm returns a bounding box that has very large height-width ratio (more than 2 times empirically), we can filter out that bounding box, too.

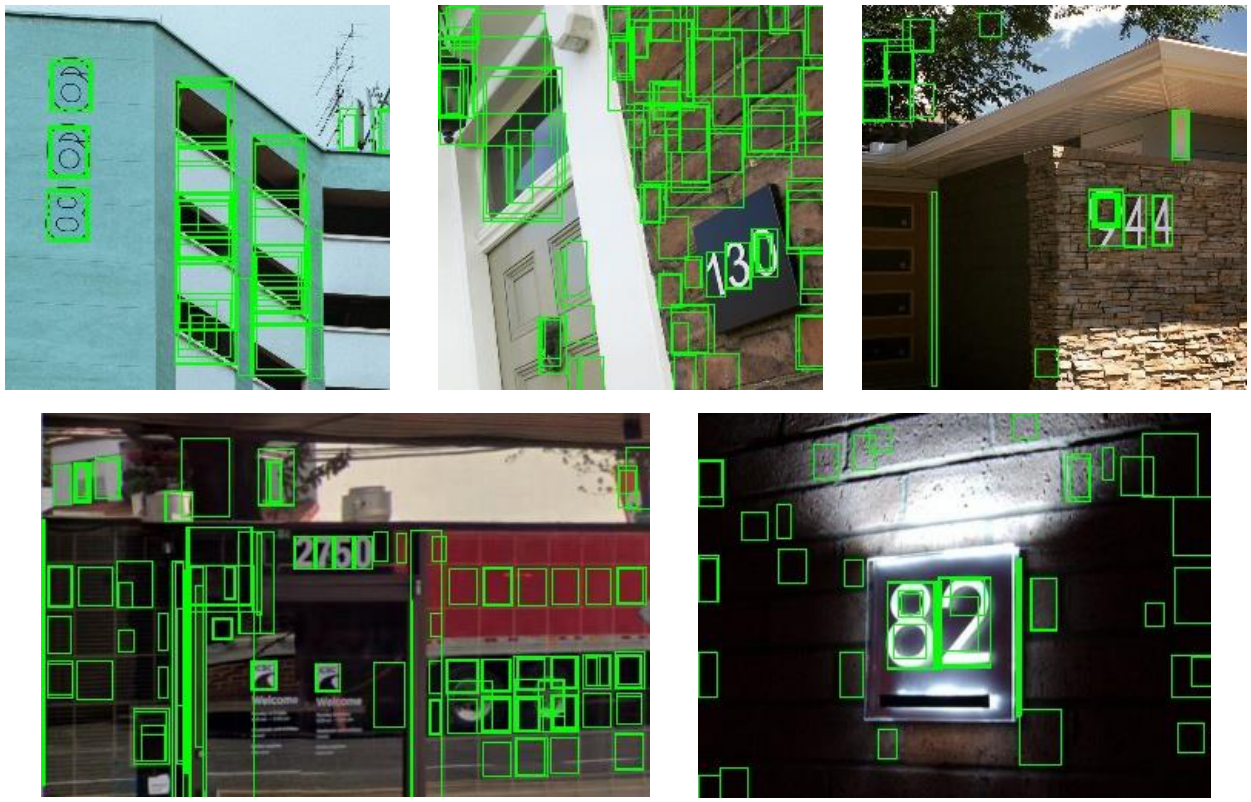

Figure 2: Results after the MSER algorithm without filtering.

Filter classification results with Non-Maxima Suppression (NMS). There are many cutout patches with labels resulting from passing through the MSER algorithm and the CNN classifier. We can reduce these information by filtering out cutout patches with label "not a number" and then run the NMS algorithm. The NMS algorithm aims to find a local maxima of classification scores in local regions. Local regions are considered "local" if the bounding boxes are close enough together defined by a parameter, or if one bounding box is completely inside another one. The NMS algorithm will only keep one cutout patch that has the best classification score and its classification label.

Filtering classification results with low scores. Based on experiment, we are filtering out results with a classification score lower than 5.0 empirically. This needs tuning or hyperparameter selection to get a better result.

Figure 3: Results after Non-Maxima Suppression and classification scores filtering.

Improving Robustness

Our classification model is equipped with the following steps to increase the robustness.

- Reuse image data from the SVHN dataset without bounding box cutouts. We remove the image intensities inside each bounding box within an image and create image data with cutouts of size 150×150 and stride 15 pixels. We use these data to construct a dataset of label "not a number" since they represent robust a "not a number" class that is close enough to the input images. We have tried using the CIFAR-100 dataset as an alternative, but it does not perform as good as our aforementioned approach. We believe the reason is that CIFAR-100 is trained on general objects like cats, dogs, cars, etc. which are available in our daily life. This dataset will not have a meaningful presence of image data around the street numbers.
- Randomized image transformations have been applied to input images. Our input transformation will change the image to grayscale with a probability of 0.1, apply perspective transformation to the image with a probability of 0.5, rotate the input image to between -20 to 20 degrees, set the brightness and contrast of the input image to between 0.8 to 1.2 times the original values, apply Gaussian noise of standard deviation of 0.2 and mean of 0 to the input image with a probability of 0.5. These approaches will help us detect images that are rotated, taken in different lighting conditions, and in different angles. Hence, this will increase the robustness of the classification model in terms of pose invariance, lighting invariance and noise invariance.

- Use Adam optimizer [5] for optimizing losses. Adam optimizer is widely used and became popular due to its efficiency and faster model convergence. It has an adaptive estimation approach about the model parameters which will adjust learning rates dynamically for them.
- Normalize input tensors. After applying randomized transformations described before, the image maybe outside the range of valid image intensities. We normalize the images using a mean and standard deviation value that is common from ImageNet. The mean is [0.485, 0.456, 0.406] while the standard deviation is [0.229, 0.224, 0.225] with respect to RGB channels.

Exploring an Alternative Model

For comparison, an alternative CNN model has also been explored. This alternative model is based on ResNet [3], a residual training network. ResNet aims to provide a mechanism so that training time can be reduced and at the same time, the model is converged faster using an identical version of input tensors added to the result after 2 convolutional layers. In the ResNet paper, the author calls this a basic residual building block. The ResNet model is scaled by adding a few of these basic residual building blocks chained together one after another. The most basic configuration mentioned in the original paper consists of 18 layers. However, in order to reduce complexity, and because we do not have a huge dataset to fully satisfy the network by training from scratch, we decided to have 2 basic residual blocks in our experiment.

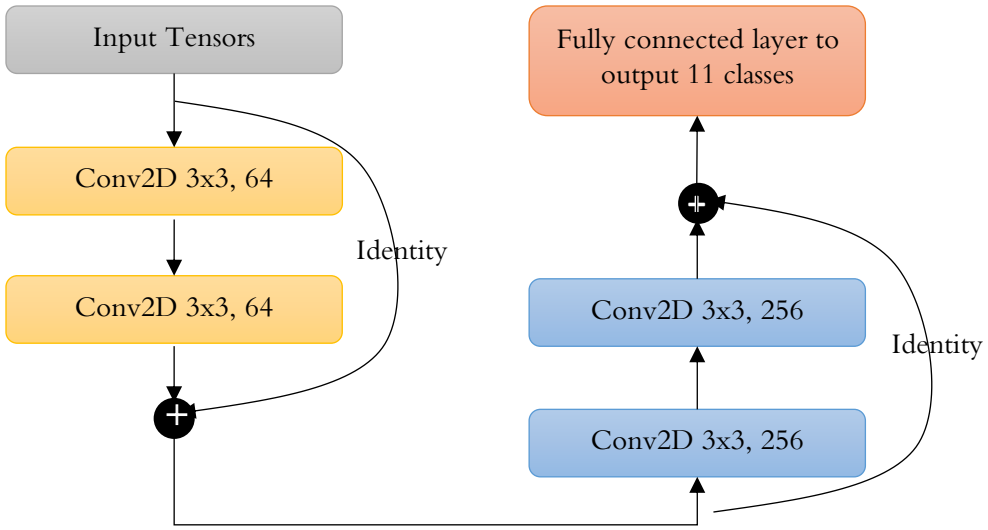

Figure 4: Layers of a simplified CNN model based on ResNet

From the experiment, there is no surprise that the pre-trained VGG-16 model performs better since it has more layers, more pre-trained weights from various images and categories. Our experimental model seems to stop improving after hitting a training accuracy of around 0.60. We believe this is because we did not use deep enough neural networks, and hence, the model is not capable of capturing that much information.

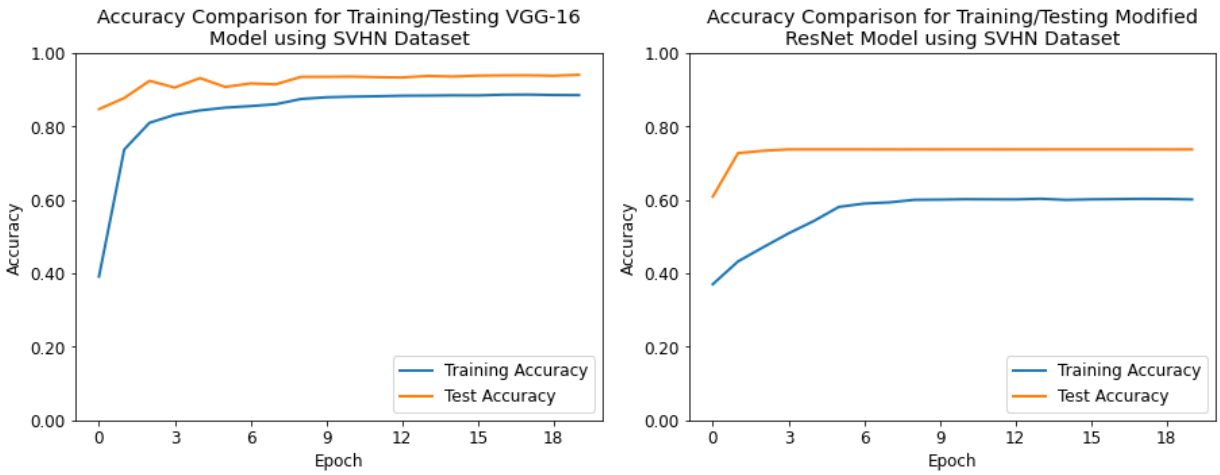

Figure 5: Comparison of training/testing accuracy for 2 different CNN models

References

[1] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011,* 2011.

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference for Learning Representations*, 2015.

[3] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385,* 2015.

[4] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing,* no. 22, pp. 761-767, 2004.

[5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference for Learning Representations*, San Diego, 2015.