CS-6476 Computer Vision Final Project Report

0. Background

The final project topic selected is Classification and Detection with Convolutional Neural Networks. Based on my research, one of most popular modern approach for object detection is YOLO system which is both fast and accurate- with the idea that turn fully connected layer into convolutional layer,avoiding redundant computations for sliding windows, and turn a classification problem into a combination of classification problem- to predict whether there is a target in a specific window- and a regression problem- to predict the position of bounding boxes in the window that the object presented.

1. Pipeline in the digit-detection system

My general approach is to use variable-size sliding windows to detect digits. To be specific, my approach involves several steps. First, to achieve a robust system, preprocessing is introduced to handle scale invariance, lighting invariance and pose invariance. My preprocessing pipeline is to leverage image data generator to load training set and validation set in batches from local directory. the data flow from directory is normalized ranging from 0 to 1, and rotated within the range of 40 degrees and also zoomed with a factor of 0.2 before fed into the learning model. To do this,it is able to train the classifier to be robust to lighting invariance and pose invariance. This preprocessing method requires us to split our data set and place corresponding images with different class labels to separate directories. With the sliding window approach, most of the image patches are negative samples,meaning that there won't be any digits in it.To predict non-digit image patches, in our preprocessing pipeline, some non-digit image patches are cropped from five images which will be used for object detection in this project. As a result, our data generator will draw training samples from 11 directories containing 11 class labels- 0 to 9 and non-digit class. The next component of our pipeline is to input the training set and validation set to our deep learning models to train a classifier to predict among the 11 class labels. The performance analysis and implementation details of three learning models- VGG16, VGG16 with pretrained weights and a self-defined model will be given later in this report. Once the learning model is obtained, the next thing to do is using sliding window approach to detect digits in the image. Variable-size sliding windows are leverages to detect digits in the image. By using variable-size sliding windows, scale invariance and location invariance can be achieved in our detection system. Furthermore, various types of font and format are already included in our training and validation examples during the training process. As a result, font invariance can also be achieved in our system. With all these components included in the pipeline, there is one big drawback in the digit-detection system- too many false positives according to the experiments I have done. To handle high false positive rate, a collaborative prediction component is introduced to my system,specifically, two VGG16 models are trained- which leads to best performance among the three models and will be discussed later in the report- with different set of negative images, the reason is that intuitively with different samples to predict non-digit class, the two models will learn different features of the non-digit class, as a a result the probability that two models will predict the same digit in the same image patch will be small, thus the decision rule of the system is to declare a certain number presented in a image patch if and only if two models make the same prediction in this image patch or a image patch nearby- the two models is likely to come to the same prediction given two neighboring image patches depending on the step size of our sliding window which is eight in the default setting. By leveraging collaborative prediction using two models with different features learned from two set of non-digit traning samples, the false positive rate falls down significantly as will be illustrated later in the report.

2. Implementation and Analysis on learning models

As required, three learning models are trained- VGG16, VGG16 with pretrained weights and a self-defined model.

For VGG16, several modification is introduced- first, the original VGG16 requires 224-by-224 image size as input and the output is 1000 class labels. To suit our classification task, the input shape is modified to be 48-by-48 images for several reasons. First, our training examples consists of 32-by-32 images and 48-by-48 input shape is closer to the size of training examples. Second, with the input shape to be 48-by-48, the number of hyperparameters of our learning model decreases, however even with a smaller model, the predictive power of the modified VGG16 should be sufficient given our learning task and input shape. The advantage with a smaller model is that the time in both training and prediction will be shorter, thus the overall performance is improved. Another modification is to change the final layer to be a softmax layer with 11 nodes to predict the probabilities among the 11 class labels.

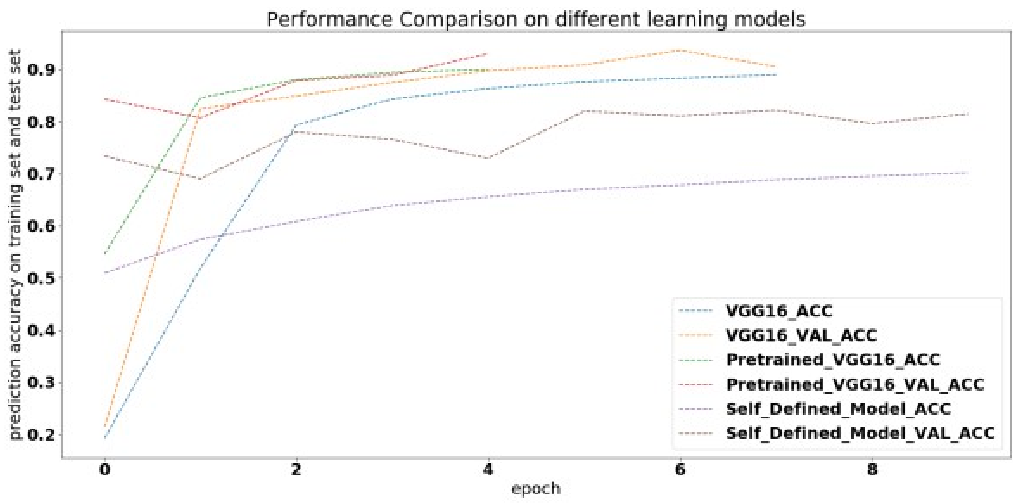
For VGG16 with pretrained weights, the same architecture is used with input shape 48-by-48. The weights comes from imageNet. The pretrained models can’t fully represent features to predict street house digits. To better suit for our specific task and leverage the features learned from imageNet dataset, the convolutional layers except for the last four layers are frozen while the last four convolutional layers and fully connected layers are retrained using SVHN dataset. By doing this, the low-level features are maintained, and the high-level features are tailored to better fit our goal- to predict the street house numbers.

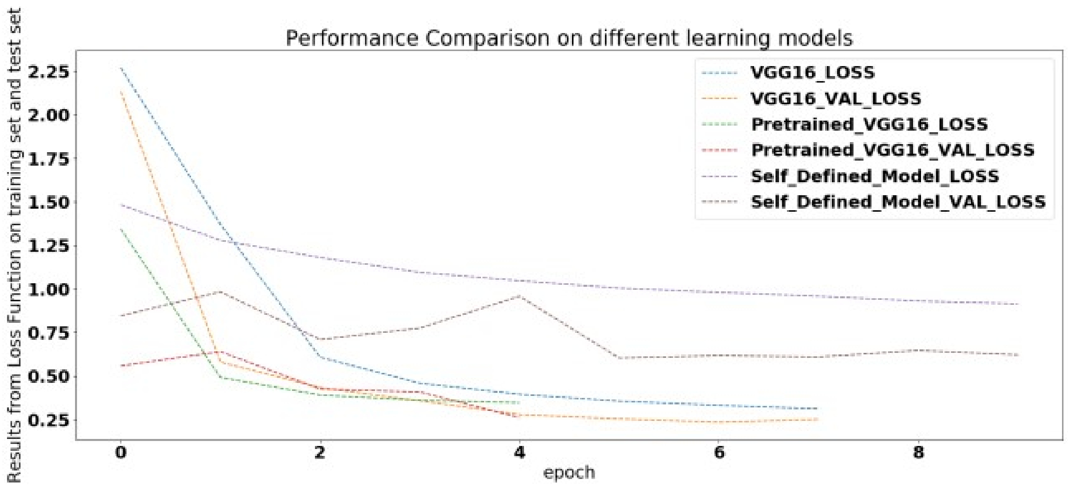
The final model that I use is a modified version of LeNet which is a famous deep learning model to predict digits. However, to predict both street view house number and non-digit image patch should be a harder problem. Hence, to enhancing the predictive power of LeNet, one more fully connected layer is added into the network. With more hyperparameters, the predictive power of this model is improved. The final output layer is a softmax layer with 11 nodes, similar to the previous VGG network.

Having implemented the three models,the training process is performed using the image data generator described in the previous section. To train the model, stochastic mini-batch gradient descent is leveraged with RSMProp optimizer. To train these models efficiently, various learning rates and batch sizes are tuned, and the variation in the loss is observed. For the learning rate quite small, the loss changes slowly, meaning that it might take too long for the model to converge, while it is too large, the loss oscilates- going up and down back and forth. By observing the learning rate, early stop can be performed to save training time and computation resource. I tuned the learning rate in the range of 1e-5 to 1e-2, finally 2e-4 is selected fro training VGG16 and Pretrained VGG16 with transfer learning, and 2e-3 is selected for training the modified version of LeNet. Various batch size is also tuned, according to the experiment, overly small batch size will slow down the learning as it is too noisy and doesn’t exploit the vectorization computation. With too large batch size, the training error increases. With several mini batch size tested-32,64,128,256- 128 is chosen in the training phase.

The prediction accuracy and loss of the three models are illustrated in the figures below.

As shown in the figures, VGG16 and VGG16 with retrained weights lead to the best validation accuracy- 90% and 88% respectively, and the modified version of LeNet merely results in around 81.4% validation accuracy. The reason the self-defined network is not as good as VGG16 is that the underlying predictive power is not sufficient for this classification task- the predictive power can be reflected by its number of hyperparameters.

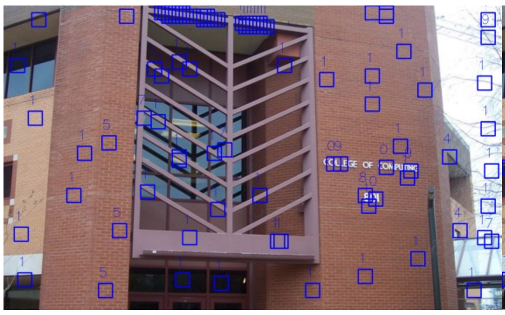
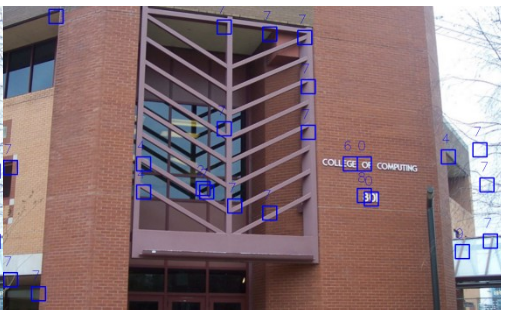


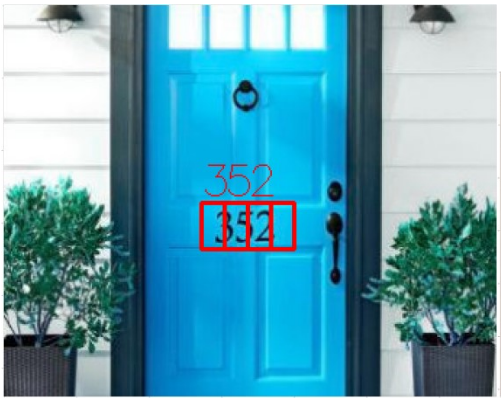


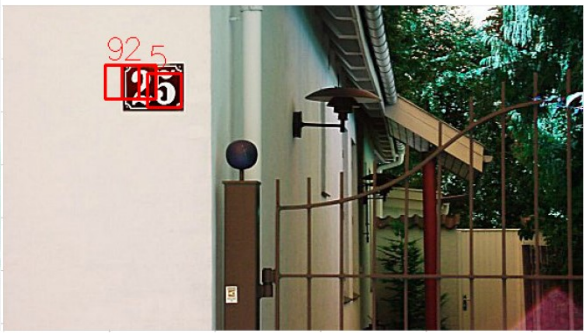
The classifier leveraged in our pipeline for object detection is VGG16 network which leads to best performance.

3. Final Result and Performance analysis

First, the figures resulted from the pipeline with single classifier is given by the two figure below. As shown, the results given by two different classifiers indicates that there are many false positives in each of the classifiers, however, there are rarely two false positives that lies in the same position with the same predicted label. In contrast, the true label is predicted correctly. Recall from the lecture on Hough Transform, this system also works like voting-the false positive is less likely to coincide, and the true positive will be voted collectively. The last five figure illustrates the results given by the final system with collaborative prediction.





As shown in the results, the system is able to achieve low false positive rate with high true positive rate. However one deficiency is that it often fails to predict ‘1’, possibly due to our selection of the non-digit training set that causes the classifier doesn’t able to recognize digit ‘1’ as some negative examples resembles it. Second, the system only detect 32-by-32, 64-by-64,128-by-128 and 256-by-256 square windows, hence, it doesn’t detect arbitrary size and shape digits. Third, by leveraging sliding window at different scales and two classifiers to perform collaborative prediction, the computations are quite intensive and ineffective. This approach is naive as compared with modern approach, for example YOLO system that can predict bounding box for arbitrary size and shape as well as parallel the computation of sliding window.
To improve this system, one can try to implement convolutional layer to replace fully connected layer to speed up computations for sliding windows. Another thing is that currently this system can only handle Gaussian noise by first leveraging Gaussian Filter to smooth the image before put into the classifier for prediction, however, it might change the statistical features of the image and won’t be a good idea to achieve noise invariance. One way to improve is to incorporate a denoising Autoencoder which is not implemented in this system due to limitation of time.

4. Reference

1) Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

2) LeCun et al., 1998, Gradient-Based Learning Applied to Document Recognition

3) Tutorial:Transfer Learning using pre-trained models in Keras from LearnOpenCV

4) Tutorial:Image Recognition using Convolutional Neural Network from LearnOpenCV