Classification and Detection with Convolutional Neural Networks

Santhanu Venugopal Sunitha
Georgia Institute of Technology, Atlanta, Georgia, U.S.A

Abstract

Convolutional neural networks, though invented in the 1908s, gained massive popularity after AlexNet won the Imagenet challenge in 2012. They are widely adopted for object recognition and detection tasks in computer vision applications. Optical character recognition or OCR is one such problem addressed by CNNs in computer vision. This study proposes to use CNNs to detect sequence of numbers in an image by training the CNNs on the Street View House Number (SVHN) dataset[1].

The video presentation is available on Youtube here
The code is available on Github here

Introduction

In order to design the digit detection and recognition system, the famous SVHN dataset[1] link was utilized, which is a real-world dataset obtained from Google Street View images and used by many for developing digit recognition algorithms. This dataset was provided as part of the research study[1], which had some interesting ideas at that time. The paper describes how using unsupervised feature learning using auto encoders lead to vastly improved performance of identifying street numbers in real world images.

Approach

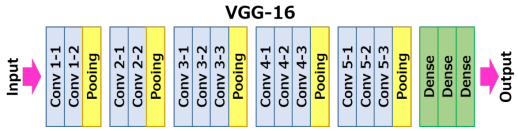
The digit sequence classification and detection was achieved by training two convolutional neural networks - a Detector and a Classifier.

The classifier was trained with the **Format 1** dataset which contains around 60,000 images was subjected to preprocessing. The dataset contains a test, train and extra folder and a corresponding .mat file which holds the image names, their label and bounding box coordinates. Using the bounding box coordinates, the digits were cropped out from the images and resized to 32 x 32 size and stored under 10 classess as train, test and validation set. After preprocessing, in order to avoid biasing the model, the number of training examples per class were balanced and the final training examples per class came around 43,000. Around 2000 samples per class were used for as a test set and 500 samples per class was used as a validation set.

For the detector, the **Format 2** dataset was used, which contains the cropped digits in 32 x 32 format of around 60,000 images, which were labeled as the 'Digit' class. A second 'Non Digit' class containing negative images (images without any digits) were built using Google Street View Dataset[2] link.The number of training examples per class were balanced here as well.

Model Variation

VGG16 is a convolutional neural network model proposed in the paper Very Deep Convolutional Networks for Large-Scale Image Recognition[5]. This model was trained with 224 x 224 image inputs and has small receptive fields for the 5 convolutional layers (3 x 3). The stride for the conv layers were fixed as 1 and each conv layer is followed by a max pooling layers with stride 2. The classifier section consist of 3 fully connected layers which finally outputs probabilities for the 1000 classes of the imagenet dataset. In this study, the VGG16 model was implemented using Keras which allow changing the input image size from 224 x 224 to 32 x 32.



(a) VGG16

Figure 1: VGG16 Architecture (Source link)

As part of designing the digit classfier CNN, the cropped out 32 x 32 images were fed to a VGG16 convolutional neural networt built using Keras with Tensorflow backend, with it's last FC layer replaced by a 11 node Dense layer for classifying the 11 classes folowed by a softmax layer. With the help of ImageDataGenerator class in Keras, Data augmentation techniques were applied to the traning images before feeding them to the ConvNet, which includes varying the brightness, zoom, shear etc.

The VGG16 was trained from scratch as well as using pretrained 'Imagenet' weights through transfer learning.

With regards to the detector CNN, the same pretrained VGG16 architecture was used with it's last FC layer replaced by a 2 node Dense layer for classifying 'Digit' and 'Not Digit'. The models were trained until it converged.

To build my own implementation of a CNN, I used three convolutional layers using 3x3 filters and a 2x2 max-pooling layer with a dense layer with a softmax function used at the end.

Training Variation

Loss function helps to evaluate our model on the given training data. It helps to meaure how good the model prediction is. With regards to training the networks, categorical cross entropy loss was used as the loss function, which is nothing but a softmax activation followed by cross-entropy loss and is mainly used in multiclass classification.

Stochastic Gradient Descent was chosen as the optimizer with a default learning rate of 0.01 and momentum of 0.9. Learning determines the rate at which the gradient is backpropagated for updating the weights. A higher learning rate may overshoot the local minima and hence miss convergence while a low learning rate will increase training time but makes the training more reliable.

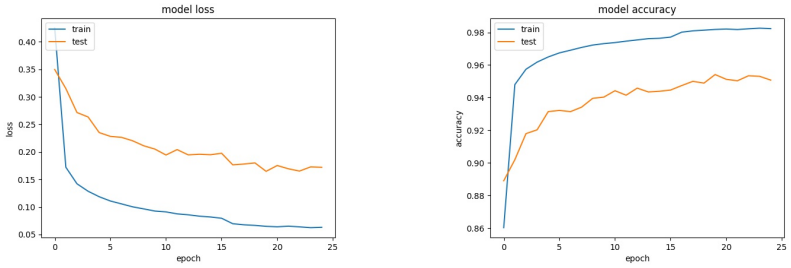
In order to reduce the learning rate when the learning stagnates, the ReduceLROnPlateau Callback function of Keras was utilized to reduce the learning rate by a factor of 0.2 when their is no change in validation accuracy over 5 epochs.

Batch size refers to the number of training samples used in one iteration for updating the model parameters. A batch size of 128 was fixed for the training. As the number of training examples were quite high, a lower batch size like 32 would slow down the model training time while a higher batch size like 512 would affect the accuracy of model due to less frequent model parameter update.

Keras has a built in callback functionality called Early Stopping, which was used to stop training the network once the validation accuracy did not improve for 5 consecutive epochs. This helps prevent the model from overfitting the data and also help reduce training time.

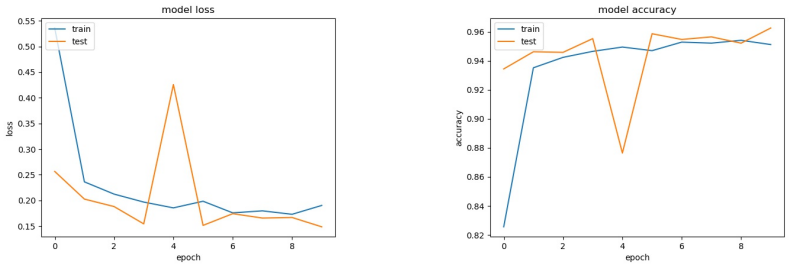
Evaluating Performance

Below plots depicts the learning curves for the VGG16 (pretrained and from scratch) and the Detector CNN.



(a) Loss Curve **(b)** Accuracy Curve

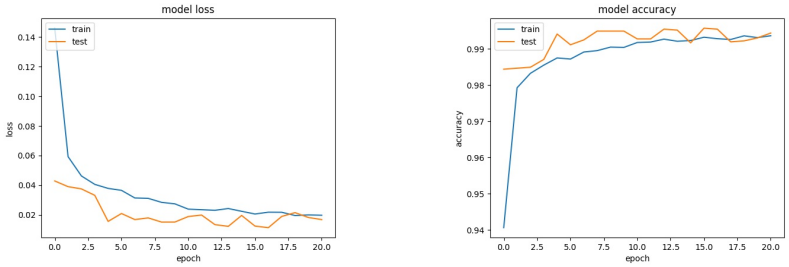
Figure 2: Learning Curves for VGG 16 trained with pretrained weights



(a) Loss Curve **(b)** Accuracy Curve

Figure 3: Learning Curves for VGG 16 trained from Scratch

As shown in the below table 1, VGG16 with pretrained weights achieved the best test set accuracy of 96%. This shows the effectiveness of pretrained models and transfer learning can be helpful



(a) Loss Curve **(b)** Accuracy Curve

Figure 4: Learning Curves for the Detector CNN

classfication tasks. Based on the learning curves, overfitting is not observed and all models stopped training as per the Early Stopping configuration in Keras. The training time for highest for VGG16 implemented from scratch, which was expected.

Model	**Train Accuracy %**	**Validation Accuracy %**	**Test Accuracy %**
VGG16 Pre-trained	98.87	95.58	96.45
VGG16 From Scratch	95.65	94.32	94.12
My CNN	90.14	90.02	85.87

Table 1: Model Performance

Detection Pipeline

The detection pipeline starts by resizing the input image to 200 X 200, followed by applying a gaussian blur with kernel size of 15 to smoothen the image. In an attempt to isolate the region containing the digits, the Maximally stable extremal regions (MSER) method was applied to the resultant image and the non textual regions were identified and eliminated from the subsequent processing steps. In order for our approach to be scale invariant, a image pyramid was constructed which downscaled the images upto 8 levels. As digits in input images can be located anywhere, a sliding window technique of 32 x 32 was implemented for scanning the segmented image. As sliding window approach is computationally slower, a stride of 15 was used to speed up the process.

Each 32 x 32 window was passed to the Detector CNN to classify whether the image contained digits or not. If it contained digits, the classified images were fed to the classifier CNN to obtain the digits.

Sliding window approach along with image pyramid can cause multiple detection of same digits. To circumvent this, a non-maximum suppression approach was implemented wherein overlapping bounding boxes of high probability for the same digit were consolidated and the ones with lower probability threshold was removed.

Results

Figure 5 shows the images which were correctly classifed by my approach. Figure 6 shows the images which were incorrectly classifed by my approach.

The approach was successful in classifying digit sequences in some images and is font, scale and orientation invariant. However, the approach was unsuccessful in classifying some digit sequences from low quality and noisy images. It also failed to classify some zoomed in images. Increasing the number of levels in image pyramid may help classify highly zoomed in images. Better image smoothening approach needs to be adopted to deal with noisy images.

(a) Digit - 32

(b) Digit - 1395

(c) Digit - 6950

(d) Digit - 87

(e) Digit - 52

(f) Digit - 670

Figure 5: Digit Detection Results - Correctly Classified

Conclusion and Future Work

There is a lot of scope for improvement in my approach with respect to false positive detections and being rotation variant. To handle false positives, the prediction threshold cut off was set very high which resulted in the elimination of detection of other true positives. This may be handled by creating a training set of loosely cropping the digits from the dataset , so as to include some part of background and add some padding around the outer boundary of the bounding box coordinates

(a) Actual - 926, Prediction - 02169

(b) Digit - 16, Prediction - 1607

(c) Actual - 926, Prediction - 02169

(d) Digit - 16, Prediction - 1607

Figure 6: Digit Detection Results - Misclassified

from the dataset, which can help the classifier to get better contextual information of the single digit.

The Detector CNN also detected a lot of false positives for digit localization.This could be improved by cropping out background negatives from the **Format 1** dataset instead of using Google Street View Dataset[2].

The sliding window approach in conjuction with image pyramid is slower and not so useful in terms of real time object recognition.State of the art techniques like YOLO[4] solves this problem by using spatial bounding box regression and removing lower threshold boxes based on confidence score.

Inspite of these drawbacks, my apporach was successful in classifying lot of digit sequences successfully and as part of future work, more research needs to be conducted to deal with noisy and zoomed in images.

References

1. 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.
2. Amir Roshan Zamir and Mubarak Shah, ”Image Geo-localization Based on Multiple Nearest Neighbor Feature Matching using Generalized Graphs”, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2014
3. Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2013). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.
4. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection.
5. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.