Classification and Detection with Convolutional Neural Networks

Soham Ghormade
Georgia Institute of Technology
sghormade3@gatech.edu

This report summarizes the work done for the final project of the CS6476 Computer Vision class.The goal is to implement a Convolutional Neural Network based method that is capable of detecting and recognising any sequence of digits visible in a given image.

Existing Methods

Deep learning has been used for image classification in recent computer vision research.In particular AlexNet architecture[1] outperformed the state of the art ,in 2012, by using convolutional neural networks stacked on top of each other.The architecture also included dropouts[2] to reduce overfitting in fully connected layers.
VGGNet[3], published two years later, demonstrated that depth of the network is an important parameter for improving accuracy.

In context of multi-digit street view number recognition, Goodfollow et.al[5] proposed unifying the localization, segmentation and recognition steps.This was done by applying a softmax classifier for every digit.

In this project,VGG16 model has been used for single digit detection.The sections below describe in detail, the methods implemented for this project.

Preprocessing

The format 2 cropped images of size 32 x 32 from Street View House Numbers(SVHN) dataset were used for training,validation and testing.The training and validation set was split into 80/20 ratio.
The pre-processing step includes subtracting the mean[3] and also dividing by 255 for normalization.

Handling noise and lighting conditions

There were no negative images in the original SVHN dataset. In particular, all the images were positive images.Due to this imbalance in the dataset, the classifier had a lot of false positives.For example, non-digit areas were classified as digits.In order to

reduce the number of false positives, 30,000 negative images were added to the dataset.
The negative images were generated from image patches.The image patches were generated from images from format 1 after excluding the largest bounding box enclosing all the digits(Figure 1).

Figure 1 :The red bounding box encloses the digit(s).The blue boxes are the image patches.This figure is for illustration purposes only.

Location and scale invariance

The digits can be located anywhere within the image. The sliding window approach was applied.Specifically,a window of size 32 x 32 was moved over every image to generate a list of image patches.These image patches were fed to the prediction step.

The window size of 32 x 32 was picked that because that size matched the input image size of the SVHN cropped image dataset.

The digits in the images can be of varying scale.An image pyramid with non maximal suppression made the classifier robust to change in scale.For image pyramid, the input image was downsampled and blurred.The blur step was necessary to minimize image distortion.This operation continued until the window size became smaller than 32 x 32.

The sliding window with image pyramid outputted overlapping bounding boxes for digits.Non-max suppression was applied to select the bounding box which exceeded the overlap threshold.The default value of 0.5 for overlap threshold worked well and hence no further trial and error was necessary.

Performance improvement

The forward pass runtime is O(m*n) in terms of Big O time complexity for an image of size m x n.This is due to two “for” loops present.For performance improvement, strides of sizes 2,5 and 10 were experimented.Size 10 was useful for initial debugging.Size 5 was picked finally since it gave decent results while improving the runtime a bit.

Model variation

Own architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
conv2d_1 (Conv2D)	(None, 28, 28, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 4096)	9441280
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 11)	45067

Total params: 26,296,299
Trainable params: 26,296,299
Non-trainable params: 0

Figure 2: layers, output shape and number of parameters in custom architecture.

I started with VGG16 model with pretrained weights which gave good results in terms of accuracy(Details in Performance statistics analysis section).So the goal I set was to build an architecture having similar number of parameters(26 million) as VGG16(33 million).Initially,I tried out a model 3 convolutional layers with two pooling layers for downsampling.Consequently, 2 fully connected(FC) layers were added to increase the number of parameters to train.Finally, dropout layers, with default value of 0.5, were added to reduce the overfitting observed in FC layers.The final prediction layer was a softmax classifier for 11 classes.The classes 0 to 9 represented digits 0 to 9 ,while class 10 represented non-digits.

VGG16 with trained from scratch

This model used random weight initialization from the keras.vgg16 method.
Notably,the training accuracy for this model in the first epoch was lower than the training accuracy seen with vgg16 using pretrained weights.Consequently, batch normalization helped mitigate the reduction in train accuracy For this model as well a prediction layer with softmax classifier, with 11 classes, was added.

VGG16 with pretrained weights

This model used the pretrained weights for vgg16.In particular the weights from training on the ImageNet dataset were re-used for the multi-digit classification task.The pretrained weights were reused to apply transfer learning[4].

The convolutional layers of VGG16 were kept but the FC layers on top were intentionally removed.The convolutional layers are generic and hence reusable for different classification tasks such as digit classification.
However, the FC layers are specific to the classification task.Also, the 1000 class softmax classifier for ImageNet classification task is not relevant for digit detection task.Consequently, these layers were replaced with following layers as discussed below.

Two FC layers were added on the top of the VGG16 model.Sizes of 128 , 512, 1024 and 4096 were experimented.4096 was picked since increasing the number of parameters at the end of the network led to higher training and validation accuracy.Finally, a softmax classifier, with 11 classes, was added for predictions.

Loss function ,learning rate ,optimizer and batch size

The loss function outputs a value which indicates how well a classifier is doing on a particular classification task.A high value of loss indicates that the classifier is doing poorly whereas a low value indicates that the classifier is doing a good job.Multi-digit classification is a multi-class classification involving 11 classes.Consequently, the softmax classifier is used to output normalized probabilities for each class.Categorical cross entropy loss is the relevant loss function used for a softmax classifier.The labels were one-hot encoded to be compatible for use with categorical cross entropy loss.

A lower learning rate is usually recommended for transfer learning[4].This is because the pretrained weights already have been fine tuned.A learning rate of 0.1 and 0.001 both reduced the accuracy.This experiment confirmed the original assertion.So the default value of 0.01 as learning rate was used for all three models.

In terms of optimizers,both Stochastic Gradient Descent(SGD) and Adam were experimented with.SGD was chosen since the train and validation accuracy for it was higher than Adam for both the VGG16 models.For the custom architecture,however, SGD plateaued with a low validation accuracy of 38%.So Adam with learning rate of 0.001 was used since it worked better.

The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters.The batch size of 64 had less variation over 10 epochs compared to batch sizes of 32 and 128.However, size of 32 was picked

because it worked well enough to move to other parts of project.64 would have been the ideal choice.

Early Stopping

The training procedure was stopped with the validation accuracy stopped improving.In particular,the keras callback function for early stopping was used. The training stopped when the validation accuracy did not improve over 20 epochs.

Results

Demo video
:https://drive.google.com/open?id=1o6Q4m2yk5O-TKrGbyKNMl2fqSraGASeg

Image results

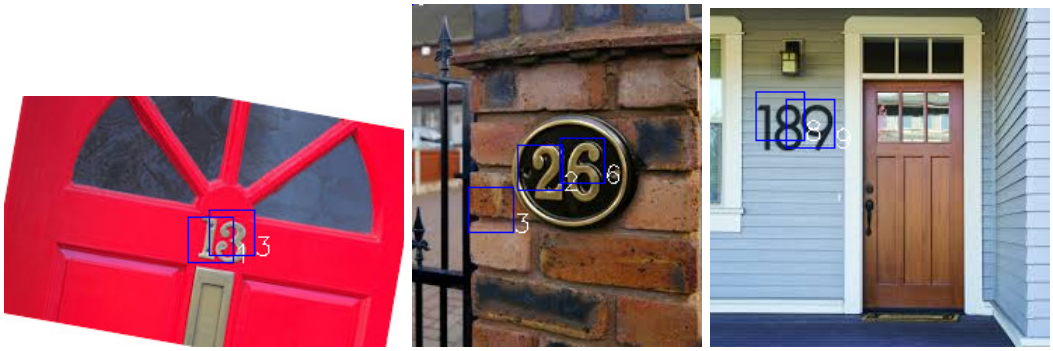

Figure 3 :Pose(left) ,font(middle) and scale invariance (right)

Figure 4 : lighting(left) and location invariance(middle) ,negative result
The classifier (incorrectly) detects 0 when the lower half of 8 is large(See negative result in Figure 4).It also detects 7 from an inverted 2.The reasoning for this behavior is because the classifier cannot discern the original digit compared to one with 180 degrees flipped orientation.It also cannot discern between a cropped digit and a similar uncropped digit.This limitation can potentially be overcome by training the classifier on labelled data for these cases.

Performance statistics analysis

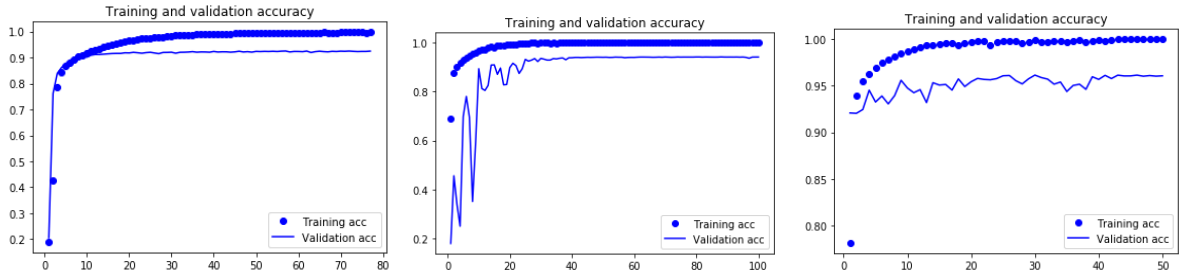

Figure 2 : Training and validation accuracy for model with own architecture, VGG16 model trained from scratch and VGG16 model with pretrained weights respectively

Sr No	Model Name	Accuracy(%)	Loss
1	Own architecture	91.54	0.71
2	VGG16 trained from scratch	93.91	0.47
3	**VGG16 with pretrained weights**	**95.84**	**0.32**

Table 1 : Test accuracy for the three models provided for comparison.

The VGG16 model was picked because it had the highest accuracy compared to the other two models and also the lowest loss.The accuracy is also close to the current state of the art[5] i.e 97.84 % for single digit detection.

Future work

The negative images added to dataset can be made diverse to include alphabets.This will make the classifier robust to non numeric inputs.

Presentation video
:https://drive.google.com/open?id=19tfkQmbPsQkMztQmvv0NcLGrAdrx3I_S

References

- [1] :A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012

- [2]:N. Srivastava, G. Hinton,A. Krizhevsky,I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.
- [3]:K. Simonyan and A. Zisserman.Very deep convolutional networks for large-scale image recognition. In ICLR, 2015
- [4]:cs231n :http://cs231n.github.io
- [5]:I. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In ICLR, 2014.