Name: Ahmad Khan

Multi-digit Classification using CNNs for Street House Numbers

Abstract

In this paper convolutional neural networks were utilized to detect street sign house numbers on images of street sign data taken from the SVHN dataset[1] to see how well existing CNN architectures performed when detecting multi-digit numbers in noisy real life images. Two different algorithms taken from existing research (the VGG algorithm and the CNN algorithm by Goodfellow et al) were trained on the data. In addition a custom model incorporating the key insights from the VGG algorithm was also trained and performance for all models was noted and evaluated. The best model (Goodfellow et al) was then utilized to detect and localize street numbers in a real video and its performance noted. Results showed that while the CNN architecture was mostly great at classifying and detecting images but it nonetheless couldn't beat the Goodfellow model in accuracy and struggled with localization in some extreme cases.

Related Work

Convolutional Neural Networks (CNN) first proposed by Yann LeCun (1998)[2] are deep learning networks which mimic how the human brain processes vision by learning image "features" using convolution operations based on nearby pixels in an image. The core framework of a CNN is a series of different convolution operation filters over nearby pixels. These stack of convolutional "filters" in turn can be arranged in individual layers so that multiple convolutional layers of convolutional filters can all stacked on top of each other with non-linear functions added to help the model learn highly complex and non linear image features relevant to the object being detected. In addition "pooling" operations to reduce dimensionality are used which summarize each convolutional layer, and extract the most important pieces of information, so that lower layers are able to detect more finer and complex features while the bigger top features detect bigger and broader features in an image leading to broader features detection.

The original CNNs (and recent ones like AlexNet) are good at detecting images because of these convolutional and pooling operations but they can be slow to train because of millions of parameters due to all the convolutional layers. In recent work modifications have been made which have made them even better at detecting images. VGG16 developed by the Visual Geometry Group at Oxford[3] improves CNNs by replacing large filters greater than size 3x3 kernels with only 3x3. The key insight being that deeper nets (with lots of layers) but smaller kernels (windows) for filters while convolving can improve the performance of CNNs because deeper nets allow the model to learn more complex and finer features but by reducing the convolution window and having more layers the receptive field still remains while still lowering the effective number of parameters to learn by a lot thereby reducing cost of training.

The above architectures however were built with the idea of focusing on classification exclusively. However in this paper we are concerned not just with classification but also the correct detection of digit sequence in addition to finding the individual digits. Goodfellow et all (2013)[4] proposed a CNN which not only detects these digits but also the correct sequence of digits in an image (123 vs 321 for example). The key insight from their research being that CNN architectures instead of having single softmax output layers to classify digits could instead have multiple output layers to predict the entire digit sequence with each output layer corresponding to predicting a digit number of the sequence (along with 1 output layer to predict length of sequence with digit length capped at 5 due to there being very few addresses with greater than 5 digits). So this model would have 5 outputs each of size N*11(for each digit value from 0-9 + no digit) and 1 output of size N*5 (for length) with the 5 output layers each corresponding to predictions for the 1st, 2nd, 3rd, 4th, and 5th digit of the sequence. Using this unified approach not only allows us to detect which numbers are in the sequence but also the correct ordering of the numbers in the sequence.

Initial Data Preprocessing

In this paper both the VGG algorithm and the Goodfellow architecture were tried on the SVHN dataset comprising 32K training images and 13K test images to measure performance on hidden data. Before training all the images had their digits cropped based on an approximate bounding box around the digits, standardized by the mean value and then were resized either to 224x224 (VGG) or 64x64 (Goodfellow). For the Goodfellow approach each image was further scaled up by 30% and then resized back to 54x54. In addition to help the model learn "no number" in an image a "negative training examples were created from each training image based on a location of the image that included no numbers.

Classification Model Architectures

Next, three variants of the VGG model were tried 1) exact same VGG16 architecture as in the VGG paper but with pre-trained weights from ImageNet data 2) exact same VGG16 architecture but using random weights 3) a modified VGG architecture that could use smaller sized digit data as opposed to the original 224x224. The original VGG16 architecture for Model 1) and 2) is as follows: 3 convolutional layers with 64 filters of kernel 3x3 and stride 1 followed by Max Pooling (2x2), then 2 convolutional layers with 128 filters of kernel 3x3 and stride 1 followed by Max Pooling (2x2), then 3 convolutional layers with 256 filters of kernel 3x3 and stride 1 followed by Max Pooling (2x2), then 3 convolutional layers with 512 filters of kernel 3x3 and stride 1 followed by Max Pooling (2x2), and another 3 convolutional layers with 512 filters with exactly the same parameters as the previous layer. Finally this layer is then flattened and connected to 3 consecutive Dense layers with 4096 neurons followed by softmax output activation for the 6 output layers described earlier. All intermediate convolutional layers have RELU activation.

For the third modified "custom VGG" model, the following changes were made. Because we are dealing with very small digit images usually less than 100 pixels in dimension it doesn't make a lot of sense to keep the 224x224 dimensions in the original VGG16. Second, very large images like 224x224 made training very difficult due to memory and disk space issues. The input therefore now is 64x64 images. Using the original VGG on 64x64 images however would cause the final layers to have tiny 2x2 image inputs (vs the 7x7 in the 224x224 VGG). To account for the reduced dimensionality of using 64x64 images such that the final dense layers are proportional to the original VGG layers in input dimensionality (and so that we don't end up with layer input sizes of 1x1 or 2x2 images) the final 3 convolutional layers with 512 filters of kernel 3x3 and Max Pooling 2x2 were removed so we now have 13 layers with final convolutional layer outputting 4x4 images). Second because the input into the dense layer is now 4x4 vs 7x7 only 4*512 = 2096 neurons for each dense layer were used vs 7*512=4096 neurons in the original VGG to account for the difference in dimensionality. Making these modifications therefore allow us to train on smaller 64x64 data more quickly while still retaining the general ideas behind the VGG16 paper.

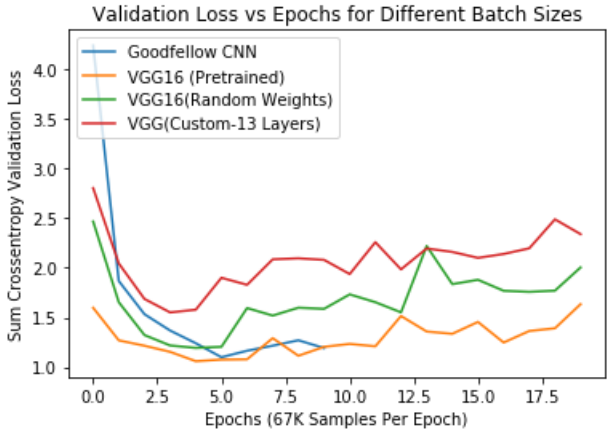
For the final model, the Goodfellow multi-digit CNN, the same architecture as described by Goodfellow was used. Input was 54x54 images followed by 8 convolutional layers of size 5x5 with the following number of filters [48,64,128,160,192,192,192,192] each with Max Pooling with alternating strides of 2 and 1 and Batch Normalization. These were followed by a Flattened Local layer and two Dense layers of 3072 neurons each. RELU activation was used at each layer except the outputs with 6 outputs layers as described earlier. The big difference here being that Dropout of 25% was added to the final two layers to improve generalization.

Model Performance Analysis

Results for each model can be shown below in terms of validation loss along with their accuracies on the test data. In this paper the sum of the categorical cross-entropies for 5 digits + length prediction was used as the loss function because we are doing classification and making categorical predictions for each digit class at each output layer unlike for example Mean Squared Error for a regression problem. Also

Accuracy was only the percentage of all images with the entire 5 digit + length sequence predicted correctly. Finally for learning optimization a variation of Stochastic Gradient Descent using Adam Optimization was used as this optimizer is easier to tune and has been shown to perform usually better.

We can see that the Goodfellow CNN architecture performs best with the lowest validation cross-entropy loss and highest test accuracy but the custom VGG (with 13 layers) performs worst (here each model was trained for 20 epochs and with the same parameters of batch size 32 and learning rate 1xE-4 with decay. 20 epochs was chosen as stopping epoch because after 7 or so epochs improvement was not observed on validation loss for any model and in fact validation loss started to worsen suggesting overfitting. Because the Goodfellow CNN and VGG Pretrained performed best, the other two VGG approaches were discarded. However VGG because of increased image size took much longer to train and second if thee size of the training set with increased with more data VGG16 would run out of memory. Because of this going forward improvements were made to help the Goodfellow based CNN model classify better only as the increased runtime/memory overhead from VGG made it more expensive and not worth it.



Model	Test Accuracy (All Digits)	Validation Loss (All Digits)
Goodfellow CNN	89%	1.21
VGG Pretrained	89%	1.23
VGG Random	82%	131
VGG Custom	72%	1.51

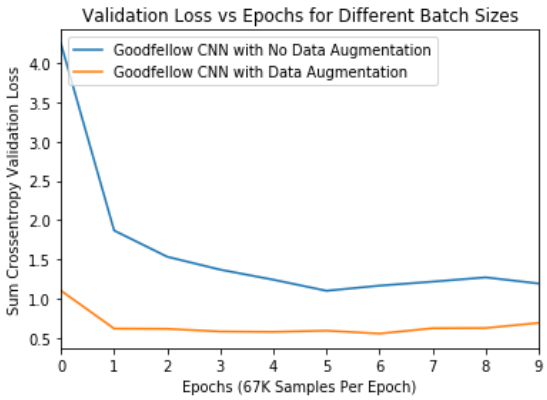
Improving Model Performance with additional Data Preprocessing

The above models so far were trained with little data pre-processing. Looking at individual misclassified images it was observed that the model was especially bad at classifying rotated images, and also images with extreme lighting or big variations in scale and noise. Hence, to make the classifier more robust to variations in scale, lighting, noise and rotation the following data augmentation was done:

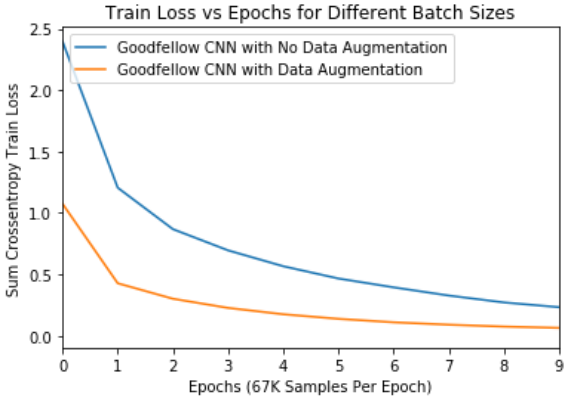
1) Adding 200K more training examples using the "extras" dataset. Having more examples to learn from would help the model learn more unique instances of images at different scales, rotations etc;
2) 5 Randomly rotated images between 8 and 35 degrees for each image were added to the data.
3) 5 Randomly shifted images between 0 and 10 pixels for each image were added to the data.
4) 200K more negative examples using the 200K extra dataset were generated.

Because of all the additional data augmentation the final size of the training set now came out to be 600K+ training images with the test set remaining unchanged. Below we can now see the performance of the CNN model both before and after additional data augmentation. We can see that the model performance has greatly improved as Validation loss is as low as 0.51 vs 1.1 before with no data augmentation. In addition, training loss is also lower for the data augmented model suggesting lower bias. We can also visualize bias and variance with both the training and test validation plotted for both models vs Epoch with the Augmented CNN model showing both curves shifted down with reduced loss. Finally out of sample accuracies on hidden test data are shown for both models and show that the augmented Goodfellow CNN model does better overall in classifying.

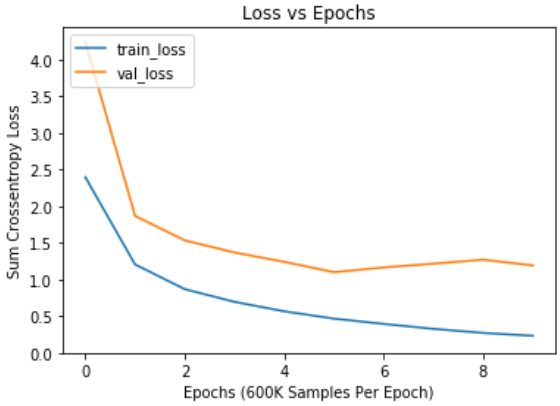
Goodfellow CNN Validation Loss (With and Without Augmentation)



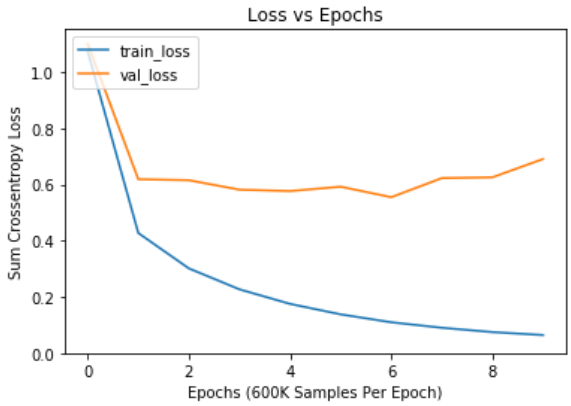
Goodfellow CNN Train Loss (With and Without Augmentation)



Goodfellow CNN Learning Curve (Without Augmentation)



Goodfellow CNN Learning Curve (With Augmentation)



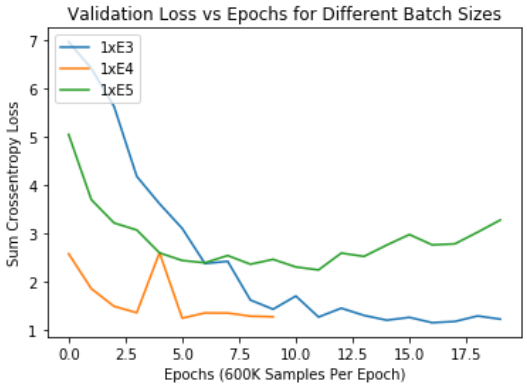
Finally, all digit test accuracy is 91.6% vs 89.1% before augmentation. However, this is still well below the state of the art results of 96% reported by Goodfellow possibly due to more robust parameter tuning.

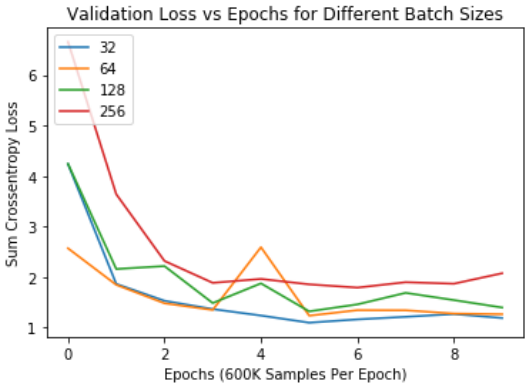
Model Performance on Test Data before and after Data Augmentation with rotated, shifted and extra data

Model	All Digit Validation Accuracy	Digit 1 Val Accuracy	Digit 2 Val Accuracy	Digit 3 Val Accuracy
Best CNN without Data Augmentation	89.1%	90.9%	90.1%	93.3%
Best CNN with Data Augmentation	**91.5%**	95.4%	95.0%	97.8%
Goodfellow Paper Accuracy (State of the Art)	96.0%	N/A	N/A	N/A

Model Parameter Tuning

The above models were trained with a Batch Size (the number of examples to feed for estimating the new gradient during gradient descent at each training step within an epoch) of 32 and Learning Rate (by how much the gradient should change after each update) of 1*E-4. These choices were discovered as the best values by grid searching over a few possible values for the Goodfellow CNN model. Shown below are the results on validation loss for different choices of Batch Size and Learning Rate. Small Batch sizes tend to reduce overfitting because by estimating the true gradient with a smaller sample we tend to be more noisy in our estimates. However this "noisineess" is what prevents the model from getting stuck in local optima (However too small a batch size can cause underfitting) and hence a small but not too small value of 32 was chosen as ideal for training. An intermediate learning rate of 1E-4 was used because as shown below too small learning rates can get stuck in local optima while too big can skip global minima entirely. Finally models were trained with and without dropout. Dropout reduces overfitting by only updating a few neurons at a time so that "bad weight updates' aren't applied to all the neurons allowing the model to learn from mistakes later on. Adding dropout increased model performance by 0.5-1%.





Localization Evaluation on Real Image and Video Data

Next, using the best CNN model based on Goodfellow's approach highlighted above, the model was used to predict the location of the digit sequence in real house images and videos. For localization a sliding window of size 40x40 with Stride 10 was used and each slice cropped from the real life image. The model was then asked to predict what or if a particular sequence was in the cropped slice. To infer the best sequence, for each cropped image the probabilities of the most likely digit number (with highest predicted model probability) at each sequence index was calculated up to its predicted length and then added to the sum of probabilities for all digits to determine likelihood. This is similar to the Goodfellow approach with the difference that probabilities are summeed and if a digit is predicted with length 0 or greater than 4 it gets likelihood = 0. To find the "best match" the cropped image location with the highest likelihood was selected.

There were however a few issues with this approach. First it wasn't scale invariant so that digits bigger than the window or too small would not be detected and even slight changes in rotation orientation and window location would cause issues. To make the localization more robust and invariant two approaches were tried and the sliding window run over the new set of smaller/bigger/rotated candidate images:

1) Create a Gaussian pyramid of the cropped image where each image in the pyramid is half the size of the previous one and then run sliding window over entire pyramid
2) Resizing each cropped image at many different scales like 10%, 20%, 30%, 50%, 70%, 150% (and also rotating at a few different rotation angles)

Between the two approaches the second approach resulted in more robust results but it was also more computationally expensive as much more candidate images were generated. However with this approach the localization algorithm was much more robust to scale, position, lighting and even slight rotation. Shown below are a few images generated by the classification CNN + sliding window/scaling localization which were correctly classified and localized. (Note: bounding box doesn't rotate intentionally)

Scale Invariance (Image Zoomed by > 100%)

Scale Invariance (No Zooming)

Rotation Invariance (Clockwise 20 degrees)

Rotation Invariance (Counter Clockwise 30 degrees)

Image Location Invariance: Change in Field of View to Left Side)

Lighting Change: Notice the Glare of Sunlight making 64 to the right blurry

However the model still struggled with extremely large rotations >= 35 degrees and sudden lighting changes/blurring/glass reflections with two misclassified images shown below

Extremely large Rotation (65 degrees) resulting in misclassification to digit 1277

Blurry Evening image + Glass reflection causing misclassification to 148

Second, as mentioned the localization was extremely slow as each image generated a few thousand cropped candidates to localized. To overcome this scale ranges were reduced knowing that digits are usually more wide than narrow (since we are looking at horizontal numbers) so that x values would be usually >= 50% and y_values could shrink by less than <= 50% to as much as 10% of the original. Second the window size and the stride were both increased to reduce the amount of cropped images.

Areas of Improvement

Despite this if given more time further changes could have been made to both the classification and localization algorithms to improve performance. For classification a static set of 600K images was created. It would have been better to use a random image generator which randomly generated rotated, shifted, scaled versions of training images in each batch vs using a static augmented dataset thus leading to more variation in training image leading and less overfitting. This would also have overcome memory issues with using VGG for augmented data. For localization, more cutting edge techniques like Faster RCNN[5] could have been utilized. Faster RCNN uses convolutional layers to identify region proposals most likely to contain the object and then uses pooling to refine and filter proposals. This way an entire exhaustive sliding window search can be avoided. Finally, image filtering techniques like MSER filtering could have been utilized to filter out non relevant image features prior to localization to increase speed.

References:

1) http://ufldl.stanford.edu/housenumbers
2) Gradient-based learning applied to document recognition: Y LeCun, L Bottou, Y Bengio, P Haffner - Proceedings of the IEEE, 1998 - dengfanxin.cn
3) Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015
4) Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay D. Shet: Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. ICLR 2014
5) Shaoqing Ren, Kaiming He, Ross B. Girshick, Jian Sun: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 39(6): 1137-1149 (2017)