Final Project : Classification and Detection with Convolutional Neural Networks

April 24, 2025

Introduction

The project aim is to detect house numbers in the SVHN dataset using a two step process.
First Step:
Detect and Bound Box the Digits: The aim is to get regions of interest from the image most likely to contain digits.
Second Step:
Use a Convolutional Neural Network based model to predict the digit in these regions of interest.
Dataset
The Street View House Numbers (SVHN) dataset is a real world dataset based on Google Street images.
Unlike MNIST, SVHN contains a lot of real world environmental noise which makes it far more challenging, requiring the above mentioned two step method to tackle.
The dataset is divided into three parts: train, test and extra.
Train contains around 33 thousand images and Test contains around 13 thousand images.
In the Train set, among 33 thousand images there are roughly 75 thousand pre labelled bounding boxes. The labelling is done for digits 0 to 9.
There also exist cropped sets at 32x32 size of street view digits. However, these sets were not used for the purposes of this project.

Step 1: Detection of Regions of Interest

There are a few major ways to detect Regions of Interest:

Maximally Stable Extremal Regions technique
Sliding window with image pyramids
DL based Semantic Segmentation eg. FCN
DL based Single Shot Systems eg. YOLO

For the purposes of this project we were asked to choose between MSER and Sliding Window techniques.
I chose to implement MSER because of following reasons:
Computational efficiency
Scale invariance
Resistance to perspective changes
Simpler parameter tuning
Maximally Stable Extremal Regions technique was introduced by Matas et at the 2002 British Machine Vision Conference[2].
MSER detects regions that are stable across varying thresholds of an ‘image intensity function’.
The paper itself uses simple grayscale intensity values of the image. However other researchers have expanded it to include color intensities as well, particularly with the Maximally Stable Colour Region techniques introduced by Forssen in 2007[3].
OpenCV provides an implementation of MSER that can only take in grayscale images.
I attempted to use color channel information in MSER by inputting each channel independently into MSER and then combining the boxes and further processing to collapse, merge and suppress the resulting array of boxes.

However, the results I obtained were not better than simply feeding a grayscale image to MSER. Probably a result of my poor bbox merging and NMS implementations.
I carefully tuned cv2.MSER's hyperparameters. The minimum area was set to 60 pixels and highest to 20000 pixels. max variation to 0.6 and Min diversity to 0.1. I found these to work the best.

1.1 Resolution and Size Handling
For this I created resolutionizer and deresolutionizer functions.
My resolutionizer function tries to convert any image into an image with maximum dimensions(height or width) of 500 pixels. This standardizes the image resolution being fed into the cv2.MSER() function. The aspect ratio is preserved.
After bounding boxes are returned from cv2.MSER(), they themselves go into a deresolutionizer which scales everything back to the original dimensions.
I do this because the CNNs are both trained on actual images and not on resolution normalized images.
1.2 Aspect Ratio Filter
I used an Aspect Ratio filter to exclude images with width:height ratios lower than 0.4 and higher than 1.1.
The aim here is to exclude and discard any boxes with aspect ratios that are unlikely to be digits.

1.3 Non-Maximal Suppression
Once I got the bounding boxes from the MSER, I used Non-Maximal Suppression to suppress and merge the remaining boxes.
Here we run into an interesting problem. Other detectors in the OpenCV library usually spit out a confidence score along with the bounding boxes. These scores allow us to direct NMS.
Since MSER does not produce confidence scores, we must implement our own.
For this I took the help of features identified in literature that are prognostic for digit ROI recognition.[4]
Two features that had been identified were aspect ratio and edge pixel density.
After an empirical survey of the training set, I realized that most digit boxes have an aspect ratio of 2:3. So, I penalized any whose aspect ratio strayed away from 2:3 and calculated a aspect ratio score.
Similarly I calculated the edge density after Canny edge detection and created an edge density score.
I combined both scores in a 1:1 ratio to create a surrogate MSER confidence score.
This score was next fed to the NMS algorithm.

1.4 Bounding Box Grouping
Even after NMS, we're still left with overlapping bounding boxes over the same digits. To combat this I used a grouping algorithm based on OpenCV's groupRectangles function.

Step 2: VGG based Digit Prediction

2.1 Non-digit sample creation
As we have talked about previously, the SVHN dataset does not include non-digit classes.
When we run our MSER pipeline, inevitably the majority of the ROIs will be of non-digit type. Therefore, it's important that our CNN training dataset contain enough non-digit data samples.
For this purpose, I run my MSER pipeline on the test images, extract ROIs, compare the ROIs with known digit bounding boxes.
If the ROIs do not match any known digit bounding boxes, then a new bounding box is created with the label 10.
In this manner I create 48 thousand new non-digit image-bounding box pairs for training.

2.2 VGG-16 Transfer Learning

To transfer learn and finetune VGG-16 on the SVHN dataset I followed simple process.
First, I retrieved vgg16 model from PyTorch model library.
I added a new classifier Linear layer with 11 output classes. This is for the 10 digits and the 11th non-digit class.I froze all the other layers.

2.2.1 Preprocessing and Augmentation

I transformed and normalized the images based on ImageNet mean and standard deviations.
This normalization helps with faster convergence speeds and adds stability to the transfer learning since VGG16 was itself trained on ImageNet.[8]

2.2.2 Loss Function

I used Cross Entropy Loss.
Cross Entropy Loss is known to work well with SoftMax and is ideal for classification problems. [5]
Also, our training set and the ROI we will get from MSER pipeline during inference are both highly biased. Most of the ROIs will be non-digit. Cross Entropy Loss is known to handle class imbalances better than other loss functions like Mean Squared Error[6]

2.2.3 Batch Size

Since I was training on a T4 GPU, I have batch size set to 128. This reliably saturate T4's 15 gb VRAM without any instability.

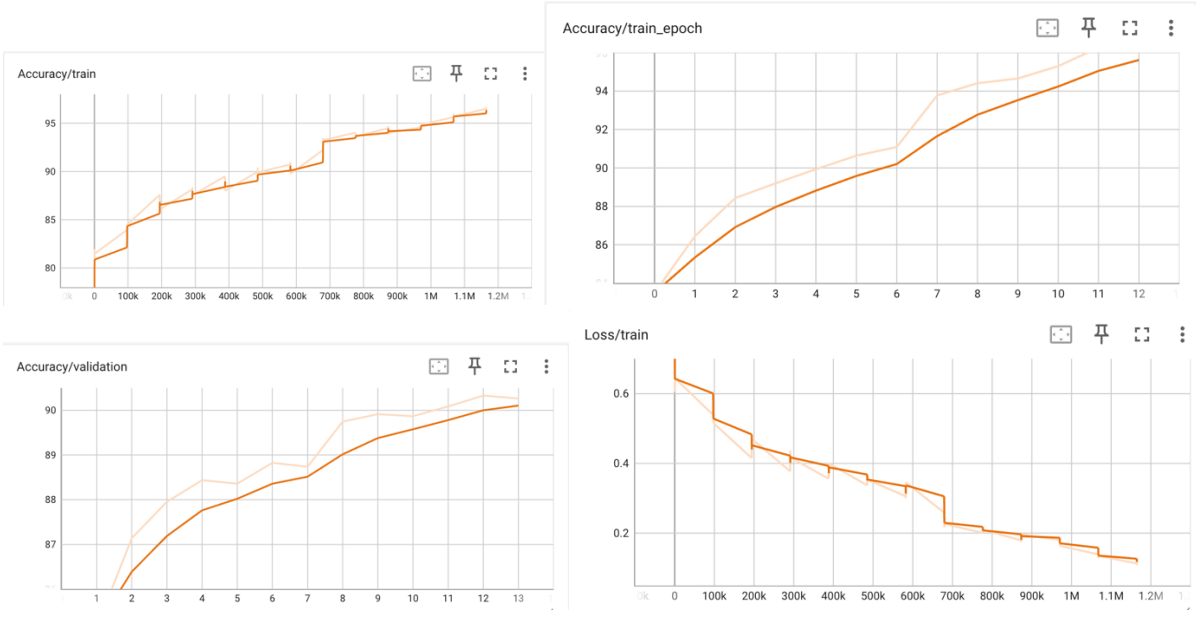
2.2.4 Early Stopping

To prevent overfitting and save computational resources, I implemented an Early Stopping technique which monitors average validation loss every epoch. If the average validation loss does not improve in 5 epochs, the training stops.[7]
My training stopped at epoch 14.

2.2.5 Optimizer

For optimization I used Adam from the torch package.
Adam automatically adjusts learning rate for all parameters unlike SGD.
Adam is able to use both momentum and adaptive learning rates together which makes the model converge much quicker.

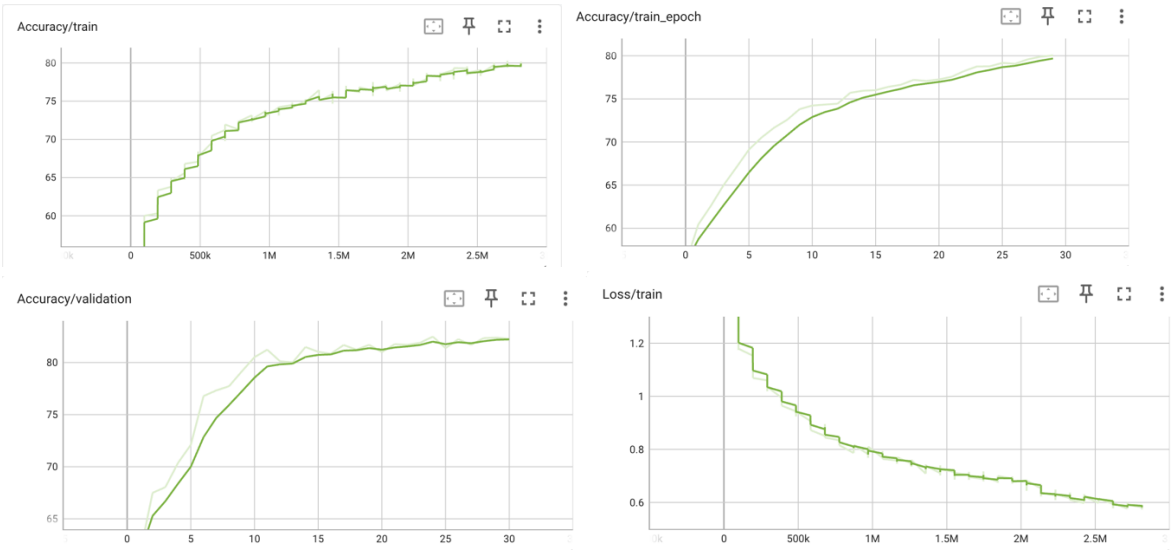
2.2.6 Training



The model achieved high accuracy, over 95% for both train and validation sets.
The model demonstrated consistent improvement in both train and validation which shows that there is effective and efficient learning.
There is very little gap between the performance on train and validation sets. This means there will be good generalization.
Overall, the performance of training was satisfactory.

Step 3: Custom CNN based Digit Prediction

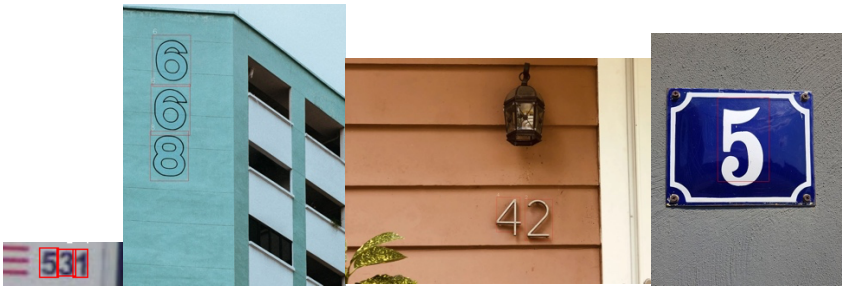
3.1 Custom CNN
The CNN architecture is an adaptation of the tutorial and consists of:
Three convolutional layers with increasing filter sizes (6, 16, and 32)
Batch normalization after each convolutional layer for training stability
MaxPooling layers for dimensionality reduction
Three fully connected layers with the final layer being of 11 output size (11 classes)
Dropout layers between fully connected layers to prevent overfitting
3.2 Carried Over Steps
Preprocessing and Augmentation, Loss Function, Batch Size, Early Stopping, Optimizer were kept same as VGG16 fine tuning.
3.3 Training



The model started with poor accuracy compared to VGG but ended up with around 80% accuracy by the end.
The model does not seem to have converged even after 30 epochs but given time and resource crunch, the best available model was taken.
Learning curves are healthy and show efficient learning with no overfitting.
The step like pattern even in the smoothed curves is probably due to using a learning rate scheduler which helps the model overcome local minimas.
Overall, the model code performed well and could have benefited from more epochs.
Step 4: Results
Please note that I also use a simple ensemble learning technique, using the confidence scores from both models to get the most correct label. This ensemble performs the best.

The successful results on the chosen 5 images are as follows:

4.1 Chosen Images



Actual House Number	VGG16 fine tune	Custom CNN	Ensemble of Both
531	531	531	531
668	668	665	668
42	42	42	42
5	32	5	5
3	3	3	3

4.2 Failures Images

4.3 Failure Interpretation

The model failed on the first 49 image because of unusual font which performs poorly when combined with perspective.The detector has also improperly segmented the digits (bounding boxes don't capture entire digits) which through the models off.
Models perform poorly with all backlight signs. This light creates shadows and reflections on the textured wall which confuses both the MSER and the CNNs.
Garden 401 sign: Even though I attempted suppressing green intensities in MSER pipeline, it did not help much. Both the pipeline and the model perform poorly when green noise is present in the image and when the digits are partially occluded. This is probably due to how they affect the edge density function in my NMS confidence score generator.

Conclusion

This project successfully implements a two-stage digit detection and recognition system using MSER for region proposals and custom CNN and a VGG fine tune for classification and an ensemble combining both CNNs.The VGG16-based transfer learning approach showed superior performance, but both models achieved high accuracy. The custom enhancements to the MSER algorithm significantly improved the detection stage.
The ensemble method performed the best and if we have low inference costs, we should always try to go for ensemble methods.

In future:

Do better green noise suppression
In the NMS confidence method, I would like to implement a continuity score, giving solid jointed areas higher score.
Attempt a single shot implementation like YOLO
Attempt attention mechanisms to improve localization in complex noisy backgrounds and obscured digits
References

[1]http://ufldl.stanford.edu/housenumbers/
[2]Matas, Jiri, et al. "Robust wide-baseline stereo from maximally stable extremal regions." Image and vision computing 22.10 (2004): 761-767.
[3]Forssén, Per-Erik. "Maximally stable colour regions for recognition and matching." 2007 IEEE Conference on computer vision and pattern recognition. IEEE, 2007.
[4] Misra, C., P. K. Swain, and J. K. Mantri. "Text extraction and recognition from image using neural network." *International Journal of Computer Applications* 40.2 (2012): 13-19.
[5] https://www.v7labs.com/blog/cross-entropy-loss-guide
[6] Aurelio, Yuri Sousa, et al. "Learning from imbalanced data sets with weighted cross-entropy function." *Neural processing letters* 50 (2019): 1937-1949.
[7]https://medium.com/biased-algorithms/a-practical-guide-to-implementing-early-stopping-in-pytorch-for-model-training-99a7cbd46e9d
[8]https://discuss.pytorch.org/t/discussion-why-normalise-according-to-imagenet-mean-and-std-dev-for-transfer-learning/115670