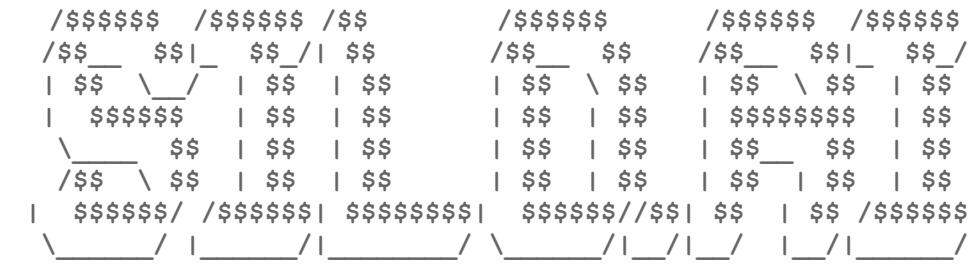
Real Time Bus Number Detection using Tensorflow, Keras and DarkNet .

Srikanth Ganta

This report is a brief write-up describing the workflow of the assignment and some results.

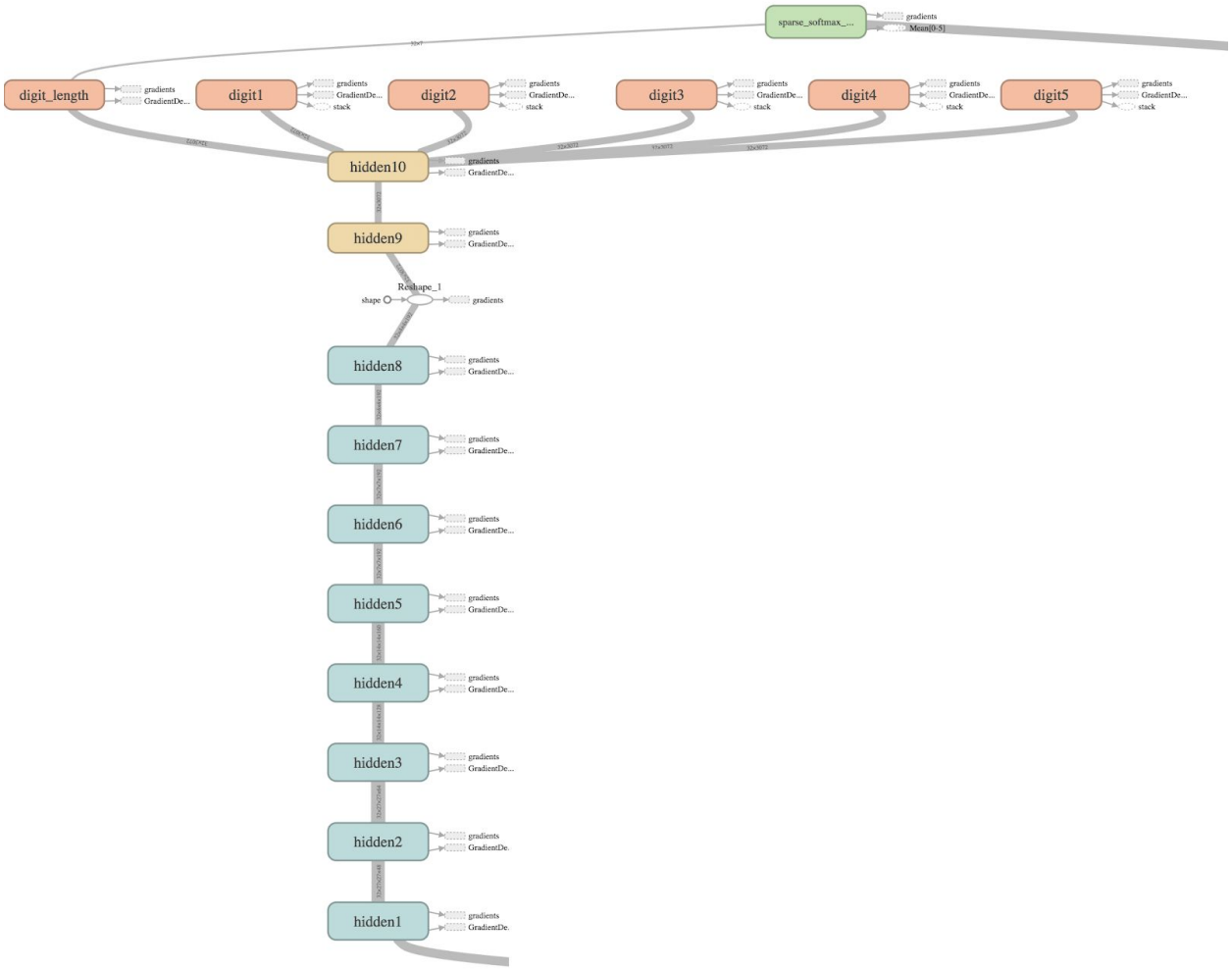
But firstly, I would like to thank Silo.AI for giving a fun-filled coding assignment. I was inspired and was determined to find a solution given my time and computing restrictions. I have learnt quite a few things along the way and I am proud of the outcome. Hope the results are satisfactory to you too. Looking forward to your feedback.



Task 1: Classifying house numbers on SVHN dataset.

For the first task i.e. classifying house numbers from the SVHN dataset. I referred to the paper '*Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks by Ian Goodfellow, et.al '* .

I referred to some existing implementations of this paper repo1 , repo2 , repo3 for the Multi-Digit classifier. The key idea here is to have multiple outputs where each output predicts a different digit of the image. The model architecture is as below.



Taken from here

Preprocessing includes extracting bounding boxes around images and converting to grayscale. Loss function used is `sparse_softmax_cross_entropy`. After about 30,000 iterations it achieved 94% accuracy on val and 90.4 % accuracy.

These results can be seen in ‘ Multi_digit_classifer.ipynb ‘.

The pre-processing of the data into h5 files can be found in ‘ Digit_Detection/TF_Implementation/SVHN-preprocess.ipynb ‘

Task 2: Detecting bus numbers in real-time in HD videos.

When I received the coding assignment, I initially thought that this was a rather trivial task. I thought that I could train an object detection model on the Street View House Numbers (SVHN) dataset and run it on the video for results, but I was proven wrong very soon.

At first, I trained a Mask-RCNN on the SVHN dataset to detect multiple digits and classify them, but due to the lack of good GPUs and it's architecture being too big I couldn't train it fast and long enough. I realised that even if I did train it for long, each frame would take at least one second to process one frame. This is too slow for real-time processing.

But before speed, I wanted to check its correctness. The Mask-RCNN trained on SVHN gave terrible results, so did YOLOv2. The dimensions of the video were just way too large for effective search. On the other hand, if the video was downscaled, the bus numbers would shrink too much and lose its features.

Hence, I decided to do the number detection in two steps:

1. First, a model detect the buses present in the image and extract them.
2. In these extracted images, another model finds the digits.

I tested YOLOv3, YOLOv2 and Mask-RCNN to detect buses and YOLOv2 performed the best.

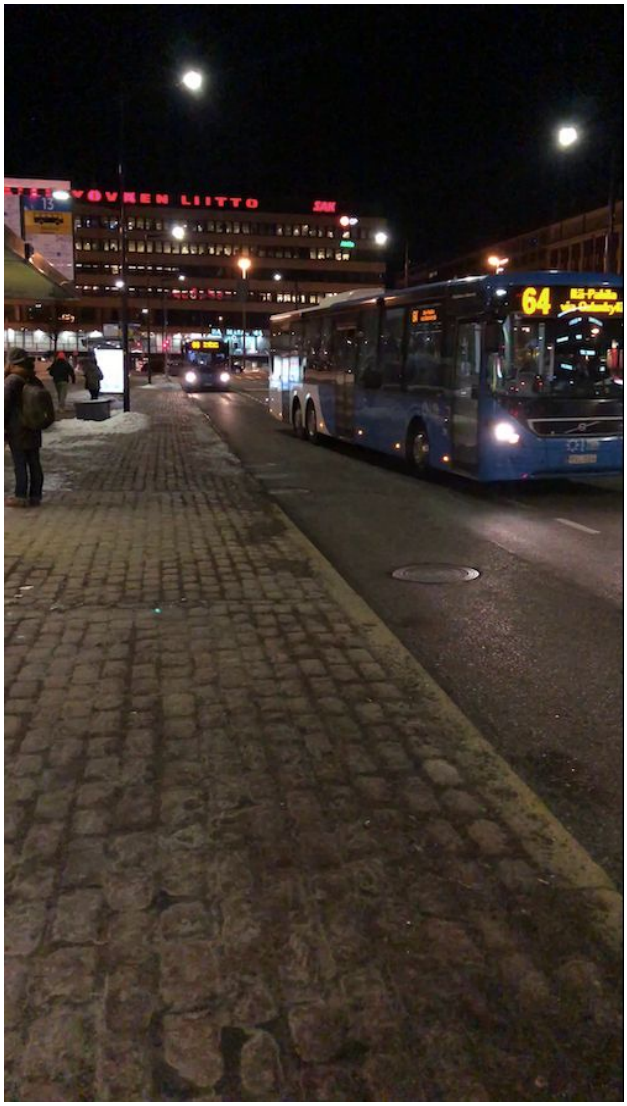
For digit detection I couldn't use the model from Task 1, as it was just a classifier and I needed bounding boxes around the digits. So, I started looking for implementations around this. I found an implementation with ResNet-50.

I integrated these two implementations to see the result. The results were still bad. The bus numbers were rarely getting detected. I think this is due to the fact that all the images in the SVHN dataset are highly secluded numbers with little to no background. All the SVHN images are heavily zoomed in with no context.

To overcome this problem, I decided to reduce the search space for the digits even more. Upon some searching , I came to the conclusion that most of the buses have the numbers in the top half.

So, that already cuts the search space in half. Furthermore, if I cut the extracted top half of the image into 4 smaller patches, I can reduce the background, which can help mimic the SVHN dataset.

Here's a visual demonstration



Original Image

Extracted bus image

Extracted top half image

Four patches of the top image.

This approach increased the detection immensely. However, the speed fell harshly. It would almost **8 seconds to process one frame**. Hence, there was a need to reduce the model sizes.

To improve the speed, I decided to use Tiny-Yolov2 pre-trained on MSCOCO dataset which would take around 0.04 seconds to detect buses. It was already a great increase in speed. But the bottle-neck was still the ResNet 50 digit detector. Each frame (4 patches) took 5 seconds to process.

I searched for a faster implementation, but to no luck. So, I decided to train it on my own. I used the Tensorflow Object Detection API for it. Upon several trails, I decided to use the SSD-MobileNetv1.

The training using the Tensorflow Object Detection API is done in four steps:

1. First turn the SVHN dataset (.mat file) into a viable CSV. (Pre-processing step)

   Digit_Detection/Training MobileNet/Pre-Process.ipynb

2. Convert those CSVs into a Tensorflow Records.

   Digit_Detection/Training MobileNet/Csv_to_Inference_Graph.ipynb

3. Use those records to train a pseudo-model, producing an inference graph.

   Digit_Detection/Training MobileNet/Csv_to_Inference_Graph.ipynb

4. Export an inference graph through the trained pseudo-model.

   Digit_Detection/Training MobileNet/Csv_to_Inference_Graph.ipynb

This inference graph can be decoded into a tensorflow graph when needed.

Upon changing from (YOLOv2 + ResNet50) to (YOLO-TINYv2 + MobileNetv1) there was a 50x speed increase. Now the final pipeline processes upto 8 fps on a lower end GPU with satisfactory results.