**Main Project CS7NS1 – Scalable Computing**

**Team – Xizhen Luo -25332297 , Hirthick Raj – 25343356 Strand - MSc Data Science**

## **1. File Retrieval**

We used VPN and proxy to connect to our PIs. Datasets are only permitted to download on RaspberryPI, so we firstly accessed the target website on the pi with following command:

**wget -O cs7ns1.scss.tcd.ie/?shortname= (luox1 or dheenadh)**

Of course, here the value of "shortname" depends on specific person. For Xizhen Luo, it is "luox1"; for Hirthick Raj Deheenadhayalan Kalpana, it is "deheenadh". Then a file contains a list of 4000 filenames was downloaded to our PI. To download all the png files listed in the file, we wrote a bash or python script to automatically do this.

**mkdir -p data  
base\_url="https://cs7ns1.scss.tcd.ie/"  
shortname=("luox1" or “dheenadh”)  
  
while IFS= read -r filename; do  
 trimmed\_filename=$(echo -n "$filename" | tr -d '  
')  
 full\_url="${base\_url}?shortname=${shortname}\ &myfilename=${trimmed\_filename}"  
 wget -O "data/$trimmed\_filename" "$full\_url"  
done < luox1-challenge-filenames**

## **2. Captcha Generation**

Before getting started, there are three challenges that we need to overcome. One, there are two different and strange fonts in both our datasets, and we must figure out what the two fonts are; Two, we must find out what characters are used in the dataset, but it takes many efforts because both fonts are so artistic that we can hardly recognize characters by eye. Three, we noticed that the length of each captcha varies, but our generate.py can only generate captchas with fixed length.

With class-wide cooperation working on the symbols and fonts, it is concluded that the symbol set includes **"0123456789adeghjknoswxBCFMPQRTUVYZ#{}[]%+-\|"**. We also confirmed that the two fonts used in our dataset are RingOfKerry and TheJjester.

To solve the variable length problem, we adopted a "padding strategy", i.e. to generate a string whose length is between 1 and the maximum (we concluded that the maximum is 6) and padding the rest blank with a placeholder symbol "?" that never appears in dataset. So, the blank will be recognized as "?" during training.

**python generate.py --width 192 --height 96 --length 6 --symbols symbols.txt --count 100000 --output-dir training\_data  
python generate.py --width 192 --height 96 --length 6 --symbols symbols.txt --count 20000 --output-dir validation\_data**

## **3. Model Training**

The architecture of used RaspberryPI is armv7I, so TensorFlow cannot be installed. Alternatively, we installed TensorFlow-Lite 'tflite'. Because TensorFlow Lite does not support full functions such as training, we had to carry out the training on another platform. Considering the computation resource, the training was done on Google Colab, which provides GPU to accelerate the speed.

**python train.py --width 192 --height 96 --length 6 --symbols symbols.txt --batch-size 64 --epochs 50 --output-model test --train-dataset training\_data --validate-dataset validation\_data**

## **4. Captcha Classification**

We gained the trained model test.h5 and test.json, but they cannot be directly used on Pi due to TensorFlow-Lite. So, we converted the h5 format to tflite format before we executed following command:

**python classify.py --model-name test --captcha-dir data/ --output stuff.txt --symbols symbols.txt**

## **5. Conclusion**

We generated captcha images whose length are random and variable with two fonts, and trained a model on Google Colab using GPU. Then we classified our dataset on RaspberryPI. According to the judgment of submitty, our model gained score 1781/4000, 1624/4000. However, we also noticed that our model has some flaws like character misplacement.

We analysed these limits and give following reasons:

1. The symbol set is still imperfect. Although we identified as many symbols as we could, mistakes and omissions are unavoidable.
2. There are two distinct fonts in our dataset and we merged the two fonts into a single training, which may have confused the model.