Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

---

# CS7NS1- Scalable Computing

# Project 2

---

**Nachiketh Janapareddy, Student Id 23337215**

**Pradyumn Bhardwaj, Student Id 19321492**

October 27, 2023

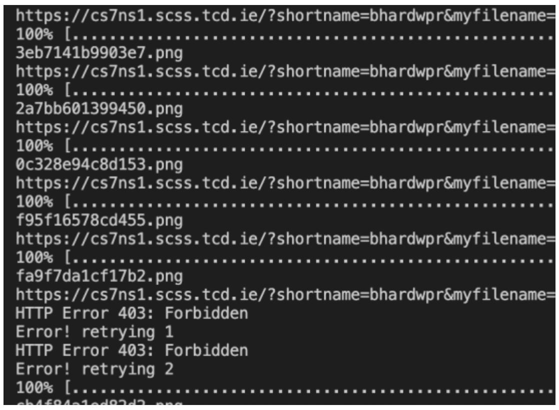
## 1. Abstract

This report presents a comprehensive examination of the strategies employed and the challenges encountered during the many stages of the project, including file retrieval, training set production, pre-processing, and the development of a new solution to enhance the Captcha decoding process. For this task, "Eamon" was the font that we were given.

## 2. Summary details of file retrieval

The main problem we encountered during the file retrieval was the traffic as it was taking a lot of time to download the images. We first downloaded the CSV file which hadthe individual image file names. Then for extraction of all the files, we created a Python script. We have used the 'wget' library to download files from the internet and the 'pandas' library to read the names from the CSV file. We added a retries and success variable to count the number of retries and successes. While downloading the files, if the download is successful, the success counter will turn true while if it is not the case, itwill keep trying to download the image again and again.

We also received an error message while trying to download the image i.e. Http Error 403: Forbidden. This error means that the server has understood the request. However, it still refuses to fulfil it because the browser does not have permission to access the requested resource. The most possible reason this error is coming in our case is because there might be security restrictions in place to restrict automated web crawling.
**We were able to retrieve files in 2 hours approximately.**



```
https://cs7ns1.scss.tcd.ie/?shortname=bhardwpr&myfilename=
100% [...............................................
3eb7141b9903e7.png
https://cs7ns1.scss.tcd.ie/?shortname=bhardwpr&myfilename=
100% [...............................................
2a7bb601399450.png
https://cs7ns1.scss.tcd.ie/?shortname=bhardwpr&myfilename=
100% [...............................................
0c328e94c8d153.png
https://cs7ns1.scss.tcd.ie/?shortname=bhardwpr&myfilename=
100% [...............................................
f95f16578cd455.png
https://cs7ns1.scss.tcd.ie/?shortname=bhardwpr&myfilename=
100% [...............................................
fa9f7da1cf17b2.png
https://cs7ns1.scss.tcd.ie/?shortname=bhardwpr&myfilename=
HTTP Error 403: Forbidden
Error! retrying 1
HTTP Error 403: Forbidden
Error! retrying 2
100% [...............................................
```

Fig.2 Error and Successful tries

## 3. Summary details of preparation/pre-processing

We applied a various image processing techniques on our different approaches. They are:

a) Normalization
Data normalization is a process that makes data more consistent by adjusting it sothat it fits within a specific range or scale. We used this in all our methods. This process scales the pixel values to be in the range [0, 1].

b) Denoising using median filter
Denoising is the process of removing unwanted or noisy elements from data to makeit cleaner and easier to work with. In the first step, the image is changed to make thetext white on a black background. Then, we reduce unwanted circle and line noise byeroding the image. Next, we revert the image to its original state and use filters to get rid of line noise. These filters help clean up pixel noise. We do another round of filtering to reduce circular noise. Then, we erode the image again to get it closer to its original form. More filtering is applied to remove any remaining "weak" noise, which could be lines or circles. We also use a method to find and remove circle noise.In the second step, we make the image look cleaner by dilating it. This helps bring back fine details and lines. We filter the image again to remove any extra noise. We erode the image once more and slightly dilate it to make it resemble the original image.
I have tried this on for the image segmentation method and added this preprocessing to padded method too. The accuracy dropped after preprocessing since it wasn’t consistent for all images and hence decided not to use it in padded method.

c) Erosion
Erosion is a process that makes objects in an image or data become smaller by removing data points from their edges. Finally, we do a small erosion to enhance the fine details of the image. The result is a cleaner image with most of the noise removed, including circles, lines, and other small disturbances. We can adjust the settings based on the specific image and noise we want to remove.
The accuracy dropped after preprocessing since it wasn’t consistent for all images and hence decided not to use it in padded method.

**To find decent image preprocessing it took 13 hours (mostly researching about techniques)**

## 4. Summary details on training and validation set creation

The generate.py file generates and saves a set of CAPTCHA images using the captcha library. It initializes the captcha generator using the font file and image dimensions. It then reads the symbols from the symbol.txt file. Unique captchas are made using a setnamed catch. For each captcha, random characters from the specified symbol set are chosen to create a random string. We ensured that the generated string was not a duplicate by checking it against the catch set.
The string is then padded with "@" characters to match the maximum captcha length.The CAPTCHA image is generated using the generate_image method. The generated image is converted to a NumPy array, and the CAPTCHA image is saved as a PNG file inthe output directory with a filename constructed from the CAPTCHA string.
Throughout this process, several difficulties emerged. Some symbols could not be used as the first character in file names due to constraints. One such symbol that was automatically replaced with "/" for Mac users was the colon (":") (Luckily, this symbol wasn't included in our symbol set, or else we would have hidden it in the created file itself). To fix this, we changed our training code to use ":" in place of "/".

**Training set details:** 100,000 Images, Width - 128, Height – 64 **[ takes 4 minutes to generate]**
**Validation set details:** 10,000 Images, Width - 128, Height – 64 **[ take 20 seconds to generate]**

The train.py file has the code to create a program for training our machine-learning model to recognize text-based CAPTCHAs. We implemented a 'create_captcha_model' tocreate a convolutional neural network model for recognizing captchas. Making the neural network, takes various parameters like length of CAPTCHAs, number of symbols used, input image dimensions, model depth and module size. The 'ImageSequence' classacts like a custom data generator which supplies both the training and validation data inbatches when the model training is happening. It loads captcha images from our directory, processes them and makes image-label pairs for training.
We then would convert the model (.h5 file) to a TensorFlow lite's model to run it in the Pi.

**We used a batch size of 25 and 10 epochs to train our model. To train a model once, it took us about four or five hours.**

## 5. Submitty solving/solution file summary

Our **initial** approach was to do preprocessing and image segmentation of captcha images. Preprocessing is the step where we prepare the image for further processing. Image segmentation focuses on splitting the captcha into individual letters. However, theproblem with this approach was that when the letters were touching, the system often interpreted them as a single letter instead of recognizing them as distinct individual letters.
*Must have spent one day trying to segment the letters correctly but the results weren't good.*

Our **second** approach was to create six models, each designed for a specific length of the captcha determined by another model. However, due to time constraints we were only able to train for five epochs per model and the accuracy of these models was significantly lower, we decided to abort this method.

*Must have spent one and half day training all the 7 different models but the results weren't good*.

The **final** approach which was successful was using a key to pad the captcha label to solve the variable length problem. In this a key, e.g.: @ is padded at the end of the label insuch a way that the label is always 6 characters long. But training mentions the key in the symbol set to train the model in such a way that the key is recognized as blanks in the captchas. After classifying through tflite in pi, we upload the CSV file in the submitty.

*This was the faster modeled trained, hence gave me time to optimise the parameters and improve the accuracy. In general, also this gave the highest accuracy without optimistation.*

## 6. RESULT

Using the padding methodology, after 6 epochs, our score was coming around 1750 and after optimizing and pushing the epochs to 10, our final score was 1820.
The classify.py file has the code to classify captcha images using our trained convolutional neural network model. After the model predicts the label, the label and the sorted files must be written to a CSV file. The formal of the is file name should be sorted, with no spaces between commas and a short name as the first row. After formatting, we must remove all the keys, i.e., @ from the label to bring it down to its actual length. The final CSV file which we get from this is later converted to CSV for final submitty submission.
**We didn't spend much time on classifying but took an hour to get the output format right.**

Pi is a 32-bit processor therefore 64-bit TensorFlow was not working. Hence, we used TensorFlow lite which converts the 64-bit TensorFlow model to a 32-bit version of TensorFlow. TFLite models are designed to be smaller and more memory-efficient thantheir TensorFlow counterparts. This is important for devices with limited computationalresources like the Raspberry Pi. Since we needed a huge amount of computation and parallel processed training, we used a faster system and the tensor flow lite model to classify in pi to generate a text file which was later converted to CSV.
**It took a lot of time to install packages into the Raspberry Pi and had to find alternatives to TensorFlow and OpenCV that are TFlite and Pillow.**

During this project, we found that it's crucial to know the nature of captchas, their variability, and potential challenges, such as noise and distortion. The quality and quantity of your training data greatly impact the model's performance. Training your model for the right number of epochs and batch size. It is important to optimize the model for inference speed and memory usage. TensorFlow Lite and hardware acceleration can be helpful here.

The capacity of a captcha detection system to scale up and manage a rising amount of captcha image processing while preserving performance and efficiency is referred to as scalability. Scalability is a critical factor for applications with different workloads since it guarantees that the captcha detection system will continue to function well and respondeven when there are more captchas to process.