Winter 2021 CMPUT 466 Midterm Report

List of all group members

- Yuxi Chen
- Zijie Tan
- Lijiangnan Tian
- Ze Hui Peng

Introduction & Related Work

Introduction/Motivation

Nowadays, CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) has been a popular method to distinguish robots from human users, which has protected our internet security for a long time. From website login to transaction confirmation, CAPTCHA is always around us in our daily life. Typically, CAPTCHAs can be divided into two categories: image-based CAPTCHA, text-based CAPTCHA. In the past few years, CAPTCHA also had a significant change in its form, from the combination of numbers and letters to selecting the right images among several images. But as the CAPTCHA develops, its corresponding recognition technology is also improving. Under this circumstance, we want to focus on the text-based CAPTCHA and implement some recognition models, from traditional algorithms to convolutional neural network models. We will make a comparison and analyze the detailed reasons for the different results. Meanwhile, we also plan to explore the impact of the training data size on the models.

Related work

Traditionally, the text-based CAPTCHA problems can be addressed by the OCR (optical character recognition) technology. For image-text recognition, the traditional method is to first make an object detection for the image, locate the single character, and then divide the image into single characters to recognize. In 2008, Yanping Lv and his team used this method to recognize the Microsoft CAPTCHA and finally achieved 60% accuracy [1]. As machine learning and convolutional neural network fields continue to evolve, better models can be trained to reach higher accuracy. In 2016, Yan and his team used a convolutional neural network model to achieve about 90% accuracy on the Chinese CAPTCHA dataset [2]. And later, the addition of the RNN (recurrent neural network) and LSTM (Long Short-Term Memory) made the text-based CAPTCHA obtain high accuracy while significantly reducing the model size, which is a huge milestone. In 2017, Baoguang Shi and his team added the recurrent layers after the convolutional neural network and named it the Convolutional Recurrent Neural Network (CRNN). This model just has a size of 8.3M but still has about 97% accuracy on different datasets.[3]

Those researches aim to improve the quality and efficiency of the text recognition models, but they only perform simple analyses of different models and focus on their models. Thus, we want to compare these models to analyze their differences and explain why they make up for the deficiencies of these previous analyses.

Data

Our data is currently comprised of ten thousand CAPTCHA images of size 200 by 50 pixels, each of which has a darkened background and contains a 6-character-long string — a combination of warped digits and deformed letters with some squiggle curves drawn across and alongside. The datasets are either from Kaggle or generated from Google Kaptcha by one of our group members, Ethan Yuxi Chen. Every image file is designated in the format `{content}_{index}.jpg` where `{content}` serves as the corresponding label and contains the actual string in the image which our models need to recognize. For this CAPTCHA recognition problem, which is a supervised classification problem, the labels of our data are multiclass, there are a total of 36 classes, which contains 10 digits and 26 case-insensitive letters. The following figure contains one of the CAPTCHA images in the data.

A sample CAPTCHA image with content **882m62**

Figure 1: Sample CAPTCHA image data from [CAPTCHA-6-digits]

While considering the accuracy of our models, our models may not predict all of the six characters in an image correctly, i.e., the prediction result may be partially correct with some characters accurately predicted and others wrongly. However, we notice that in the real world CAPTCHA is an all-or-nothing situation, you do not pass unless you get all the characters correct. Therefore we use the rudimentary string equality instead of comparing the prediction result to the corresponding label character-wisely.

The interpretability of our models is not crucial since we do not need to know how our models come up with the predictions.

Analysis / Methodology

Most of the currently available text-based CAPTCHA recognition models can be classified into two categories: segmentation-based algorithms and segmentation-free algorithms, according to Thohbani et al.'s research [4]. We are interested in the differences between the effects of these models and algorithms, as well as the reasons behind these differences.

We plan to apply some most commonly-used methods of both categories and their combinations to train text-based CAPTCHA recognition models, compare their performances, and investigate the factors that may affect their performances.

The algorithms we are currently working on:

1. Segmentation-free algorithms

a) Convolutional neural network (CNN)

The first method we tried is the convolutional neural network. Before entering the neural network, the verification code is converted into a grayscale map and binarized. The structure of the convolutional neural network is depicted in Figure 3. The network starts with a Convolutional layer with 1 input channel, the batch normalization layer, the relu activation function, and 5×5 kernels. Followed by a 2×2 max-pooling layer. This model has 4 parts similar to this one, and they have 32, 48, 64, 64 neurons respectively. Then the data will be flattened and pass through two full connection layers. Finally, this model returns a sequence of length 216. This sequence will be divided into 6 equal segments, each segment has 36 values. The model can select the index of the maximum value in each segment of the sequence and generate the predicted CAPTCHA by finding the corresponding character in the data table with the selected index.

In our experiments, we use the same method to measure performance. And the details related to performance measurement are later in this section. For this model, we currently have 5 hyperparameters to tune, including learning rate, number of iterations, activation function, the number of the hidden layers, and the dropout function.

The learning rate controls the magnitude of the model updates. The number of iterations controls the progress of the model training. The activation function can give nonlinear functions to the model and improve the performance of the model. But different activation functions have different characteristics. The number of hidden layers affects network performance and the computational cost of training this network. The dropout function can prevent the model from overfitting.

But we need to pay attention that these hyperparameters are not the bigger the better. They usually have some problems when they are large. Thus, we need to find suitable values for them.

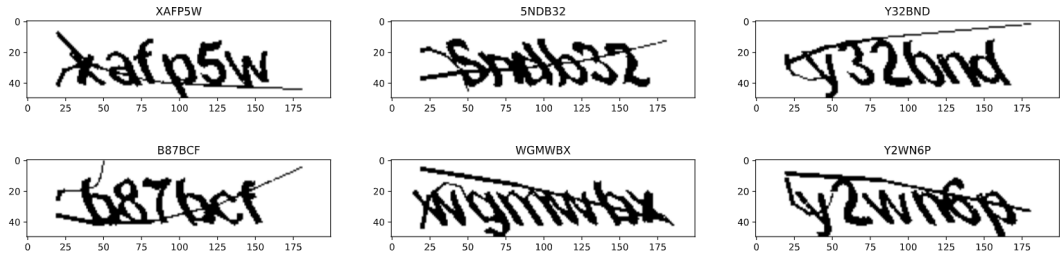

Figure 2: Sample CAPTCHA images after binarization

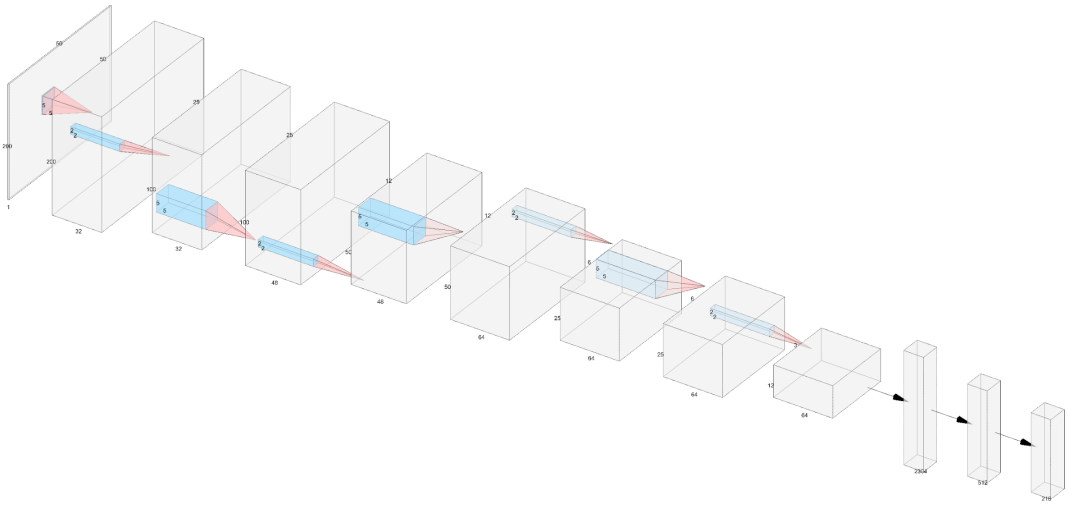

Figure 3: The Architecture of the Convolutional Neural Network

b) Recurrent neural network (RNN)

The second method that we are interested in is the recurrent neural network. Fig. 4 shows the structure of the most basic neural cell in the RNN. It uses the following equations to compute the output: $o_t = Wh_t$, $h_t = f(Ux_t + Vh_{t-1} + b_t)$, where f is the activation function, U, V, W are the 3 weight matrices, and b is the bias term. This model takes input sequentially and computes output recurrently, which provides it the potential to solve variable-length sequence prediction problems. In our case, we hope that this architecture will allow us to extend the model to process and recognize variable-length CAPTCHA.

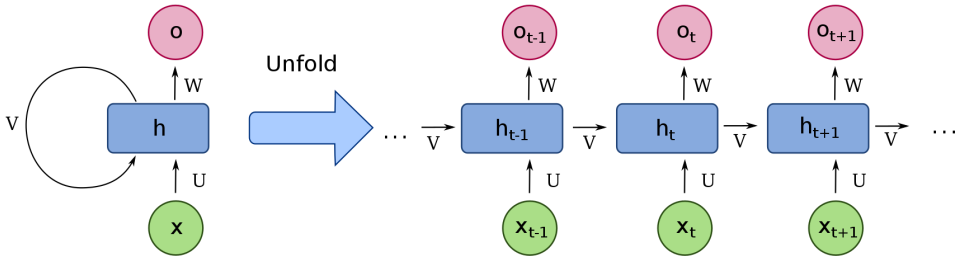

Figure 4: The structure of standard RNN cell and its data flow [6]

However, in practice, we found it quite difficult for the standard recurrent neural networks to efficiently complete this task. In our experiments, the naive RNN hardly converged, and its accuracy is extremely low (~0.02%). So we decided to turn to some other variants of the RNN network.

The text-based CAPTCHA recognition involves taking in an image, which is represented by tensor, as input and producing a string as output. So this task is a typical sequence-to-sequence task (seq2seq). Therefore, we decided to use the encoder-decoder RNN architecture, which has been proven an effective approach to such a seq2seq prediction problem [7].

The encoder-decoder RNN architecture consists of 2 RNN cores: one for encoding the input into a fixed-length vector, the other for mapping the vector representation back to a variable-length target sequence. We use `PyTorch` to implement an encoder-decoder RNN model that contains an input dense layer, an encoder layer, a decoder layer, and an output dense layer as one of our preliminary models. Fig. 5 and Fig. 6 demonstrate the architecture and the data flow.

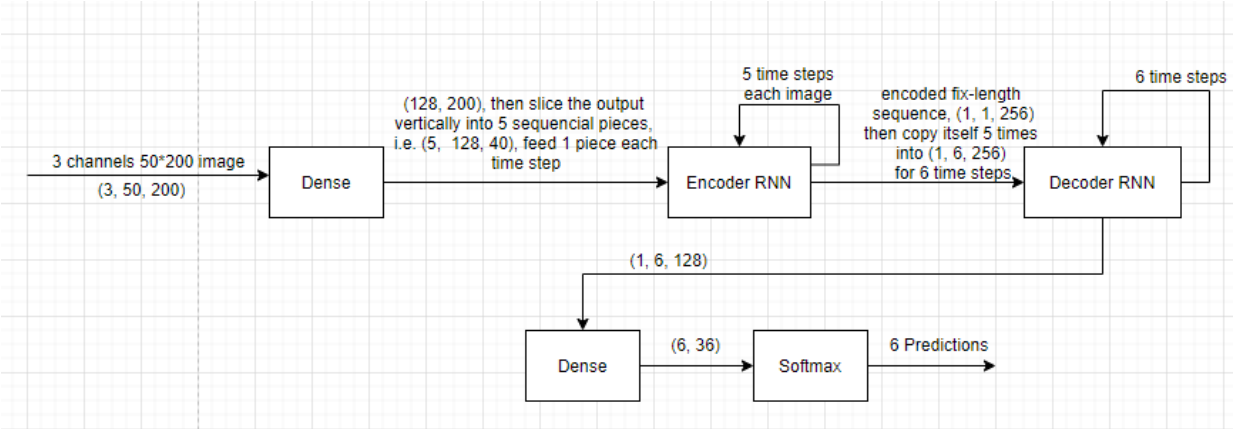

Figure 5: The architecture of our encoder-decoder RNN model

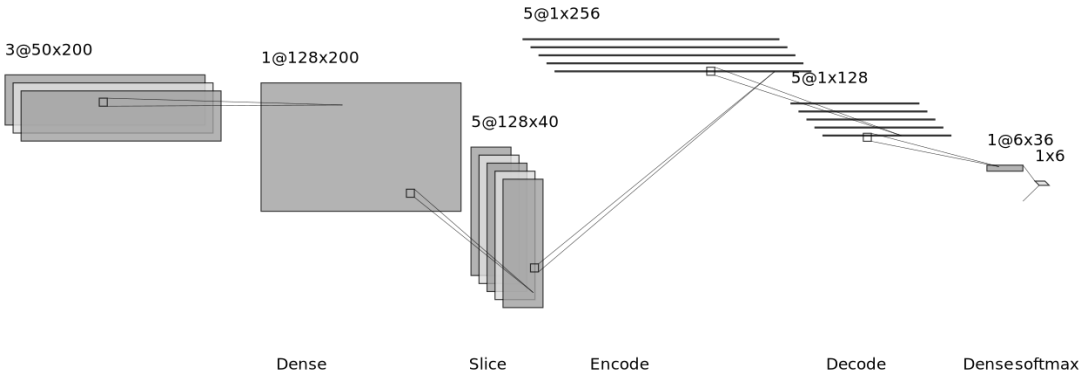

Figure 6: The data flow of our encoder-decoder RNN model, note that in the Slice step, we slice the image horizontally into 5 equal pieces so that they can be fed into the model as sequential inputs for 5 time-steps

2. Segmentation-based algorithms

The common step before applying the algorithms listed below is to do image segmentation to separate every digit or letter in the image. If you check the sample image listed in Figure 1, you will notice that the image has noise-lines which make it harder to identify the digits and letters. The first step we planned is to apply median filters to remove some of the line noises, after the line noises are removed, then we can apply character segmentation algorithms to separate each of the individual characters.

c) Support-vector machine (SVM)

The third method we explored was the support-vector machine model. As a segmentation-based algorithm, the first phase is to preprocess the image which helps to segment the CAPTCHA image into 6 smaller images each of which ideally contains a single alphanumeric character. However, segmentation is also the most difficult and most important step among all the steps in the preprocessing phase, and it significantly impacts the performance of the SVM in the sequential recognition step. Hence, the preprocessing procedure would be much more complicated than the one in a segmentation-free algorithm. The current preprocessing procedure consists of seven steps: erosion, grayscale, binarisation, morphological transformation, shear transformation, horizontal stretch, and lastly, segmentation. All those steps before segmentation aim to remove noises, take out the curvilinear patterns in the background, and straighten the characters and increase the spacing between adjacent characters so that it would be easier to separate them in the segmentation step. The current segmentation step is rudimentary as we just separated the preprocessed CAPTCHA image into even pieces, which works not well when there are characters in the picture

that are particularly wide such as `W` and `M`. A successful segmentation is illustrated in Figure 11. We plan to design a more efficient and effective segmentation method. The SVM model used is C-Support Vector Classification from `scikit-learn`. The SVM contains a hyperparameter C, called the regularization parameter, which helps us tune the importance of misclassified data points.

d) k-nearest neighbours algorithms (KNN)

The fourth method that we are trying is the k-nearest neighbours algorithm for classification. After the image has been segmented into individual letters, each letter in each image from the training dataset will be classified as one of the 36 classes (10 digits + 26 letters, case insensitive) that correspond to their true label. Then for each digit/letter in each image from the test dataset, we will calculate the euclidean distance with all the data from the training dataset. After all the distances are calculated, the data with the lowest K distances will be used for the classification, each digit/letter will be classified with the class that has the largest occurrence among the K nearest neighbors. Here K is a hyperparameter that represents the number of closest distance data, we will be trying out many different K values to check out which K value will yield the best performance.

Measuring the performance:

In general, this is a classification problem, and therefore it is natural to use classification accuracy/error to measure the performances of the models we train.

At this stage, we are still focusing on fixed-length 6-alphanumeric-characters CAPTCHA recognition. And another temporary assumption we made to simplify the preliminary models is that our CAPTCHAs contain only lowercase letters and numbers (so 36 possible characters in total).

We are using a database that contains 10,000 labeled CAPTCHA images at this point. For the consistency of model comparison, we decide to use the same data segmentation for all 4 models, that is, use 6000 images for training, 2000 images for validation, and the remaining 2000 images for testing.

Preliminary Results

1.CNN

Currently, the CNN model can achieve an accuracy of about 97% after 15 epochs. We think that in the CNN model, the convolution operation and batch normalization compress the information of the image, that is, extract the feature information. And in this process, the interference in the image will also be compressed. Moreover,

compared with characters, the percentage of interference in images is very small, and the influence of interference on features is greatly reduced in the process of multiple feature extraction. Finally, the neural network can recognize the features correctly.

Based on this conjecture, we can visualize the neural network, for example, by generating a heat map of the neural network to understand which parts of the image are at play. It is also possible to abandon the preprocessing of the image, i.e., abandon the grayscale processing and binarization of the image, and train it again to compare the performance.

```
Training
epoch: 1 loss: 0.1037985235 accuracy: 0.0000
epoch: 2 loss: 0.0726780668 accuracy: 0.4500
epoch: 3 loss: 0.0440325253 accuracy: 11.4000
epoch: 4 loss: 0.0162601117 accuracy: 75.7500
epoch: 5 loss: 0.0065423511 accuracy: 90.5000
epoch: 6 loss: 0.0036285631 accuracy: 94.1000
epoch: 7 loss: 0.0018757052 accuracy: 95.1500
epoch: 8 loss: 0.0012234495 accuracy: 96.4500
epoch: 9 loss: 0.0006631794 accuracy: 96.9000
epoch: 10 loss: 0.0004093000 accuracy: 96.8500
epoch: 11 loss: 0.0002872905 accuracy: 97.1000
epoch: 12 loss: 0.0002029521 accuracy: 97.2000
epoch: 13 loss: 0.0001710304 accuracy: 97.2500
epoch: 14 loss: 0.0001374489 accuracy: 97.2500
epoch: 15 loss: 0.0001029570 accuracy: 97.2500

Testing
Accuracy: 96.75
```

Figure 7: The training progress of the CNN model

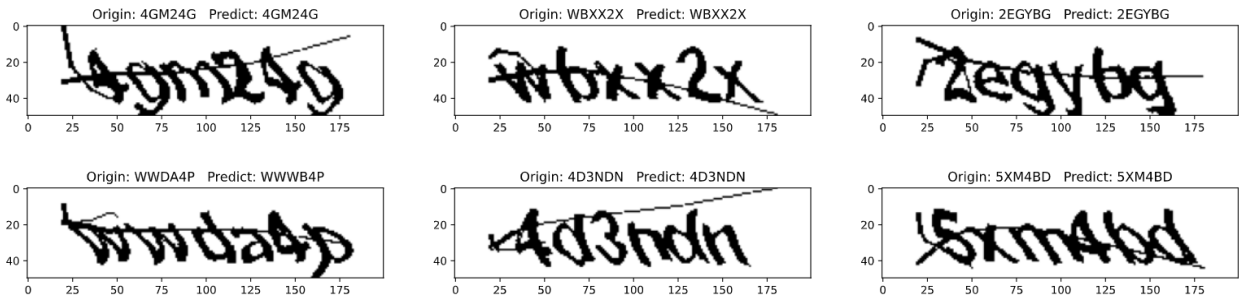

Figure 8: Examples of CNN model for prediction

2. RNN

Compared to the CNN model, the performance of our RNN model is still much lower after a relatively large amount of epochs, although we have applied many improvements to it, such as increasing the number of hidden layers, adding a dense layer, etc. It takes much more epochs to converge (nearly 60 epochs), and the final

accuracy of it is much lower (73.1% against 96.75%). One of the possible reasons for this is that, unlike the CNN model, when the vector representation of the images is fed into the model sequentially, the spatial properties of the images are not reserved in the computation.

Another hypothesis we made is that when the RNN model takes the previous output as one of the inputs at every time step, the past information becomes more heavily involved in predicting the next character of the CAPTCHA string. This may have negative effects on the prediction process since there is no strong connection between two consecutive characters in a CAPTCHA string. We still need some further experiments to prove or disprove this hypothesis. If the hypothesis is correct, adding a scalar to the previous output to lower its influence may improve our RNN model further.

There is one interesting property we discovered during the training process, that is if we use the character-wise accuracy instead of the CAPTCHA-wise accuracy, the accuracy reaches approximately 92%. In other words, although the model failed to predict most of the CAPTCHAs 100% correctly, it did succeed to correctly predict most of the characters and in most of the fail-case, it got only 1 character wrong. The reason behind this is still undercover.

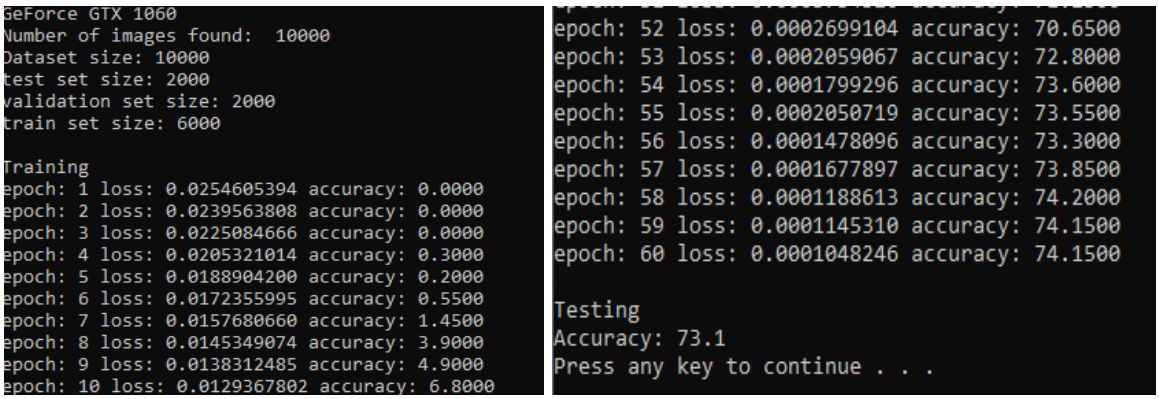

Figure 9: The result of the encoder-decoder RNN model experiments

3.SVM

Currently, our two segmentation-based algorithms lack an effective segmentation method, so we have not proceeded to study the models themselves. Nonetheless, the SVM model we are using can predict most characters in a CAPTCHA image (segmented by our rudimentary segmentation method) correctly most of the time.

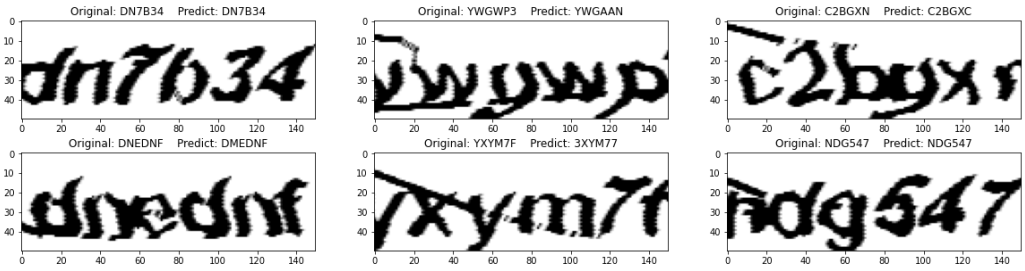

Figure 10: Samples of CAPTCHA with corresponding labels and predicted ones

The current rudimentary segmentation method works well when the alphanumeric characters in the CAPTCHA image have narrow and similar widths.

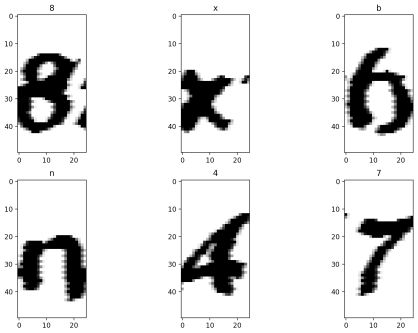

Figure 11: An example of segmentation of CAPTCHA **8xbn47**

4. KNN

Similar to SVM, we have not yet proceeded with the algorithm since we are currently trying to improve our image segmentation method to effectively remove the line noise and separating each of the letters for each CAPTCHA image.

Work Plan

Here is a screenshot of our work plan from Google Sheets:

Work Name	Estimated Hours	Expected Completion Date	Assignee	Estimated Progress	Complete	Additional Notes
Project Proposal	10	February 2, 2021	All	100%	☑	*The estimated hours is the sum of all sub-categories
↳ Introduction & Related Work Section	2.5	January 31, 2021	Yuxi Chen	100%	☑	
↳ Data Section	2.5	January 31, 2021	Zijie Tan	100%	☑	
↳ Analysis & Methodology Section	2.5	January 31, 2021	Lijiangnan Tian	100%	☑	
↳ Work Plan Section	2.5	January 31, 2021	Ze Hui Peng	100%	☑	
Create Github Repository	0.5	February 3, 2021	Ze Hui Peng	100%	☑	* Should disable user directly pushing to main/master * Every Pull Request(PR) should have at least one approval before merging
Assignment 1 Due	-	February 10, 2021	-		-	-
Exam 1	-	February 11, 2021	-		-	-
Cragarnize and/or Generate Training and Test Datasets	5	February 12, 2021	Yuxi Chen	100%	☑	* There are some datasets available on Kaggle * Yuxi also found out how to generate CAPTCHA images
Project Midterm Report	5	March 11, 2021	All	100%	☑	*The estimated hours is the sum of all sub-categories * see subsections below for more details
↳ Introduction & Related Work Section	0	March 10, 2021	-	100%	☑	* No changes required from project proposal
↳ Data Section	0.5	March 10, 2021	All	100%	☑	* Edit and clarify label characteristic based on TA feedback
↳ Analysis & Methodology Section	2	March 10, 2021	All	100%	☑	* rewrite this section based on the example structure given * and write more details regarding each algorithms we worked on
↳ Perimilary Results Section	2	March 10, 2021	All	100%	☑	* give some results and statistics based on what we have so far
↳ Work Plan Section	0.5	March 10, 2021	Ze Hui Peng	100%	☑	* Updating works that have completed thus far
Assignment 2 Due	-	March 11, 2021	-		-	-
Exam 2	-	March 16, 2021	-		-	-
Reading Assignment 2 Due	-	March 25, 2021	-		-	-
Implement&Train model using CNN	25	March 19, 2021	Yuxi Chen	60%	☐	
Implement&Train model using RNN	25	March 19, 2021	Zijie Tan	60%	☐	
Implementing Image Segmentation	10	March 19, 2021	Ze Hui Peng Lijiangnan Tian	50%	☐	* include removing image noise and character segmentation
Implement and train model using a combination of IS and KNN	15	March 26, 2021	Ze Hui Peng	50%	☐	
Implement and train model using a combination of IS and SVM	15	March 26, 2021	Lijiangnan Tian	50%	☐	
**Investigate into other possible algorithms	unknown	March 26, 2021	N/A as of now		☐	** If time permits
Test all available models	10	April 1, 2021	All		☐	* Report the running time and accurency for each model using the same set of test data * test out different hyperparameters
Project Demo Video	5	April 4, 2021	All		☐	
Assignment 3 Due	-	April 13, 2021	-		-	-
Project Report	15	April 14, 2021	All		☐	

Notes:
1. depending on the progress, the Work Plans are subject to change
2. any row highlighted in blue is not part of the project, but part of the course that might interrupt the work plan

Glossary:	
Covolutional Neural Network	CNN
Recurrent Neural Network	RNN
Image Segmentation	IS
K-Nearest Neighbour	KNN
Support-Vector Machine	SVM

Figure 12: Screenshot of our work plan timeline

If the image is too small or you are having trouble viewing the image, you can view our up-to-date version of our project work plan from our [Project Work Plan Google Sheet].

References

1. A low-cost attack on a Microsoft captcha. 2008. Accessed February 2, 2021. https://search-ebscohost-com.login.ezproxy.library.ualberta.ca/login.aspx?direct=true&db=edsoai&AN=edsoai.on1098300311&site=eds-live&scope=site

2. Lv Y, Cai F, Lin D, Cao D. Chinese character CAPTCHA recognition based on convolution neural network. 2016 IEEE Congress on Evolutionary Computation (CEC), Evolutionary Computation (CEC), 2016 IEEE Congress on. July 2016:4854-4859. doi:10.1109/CEC.2016.7744412

3. Shi B, Bai X, Yao C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39(11):2298-2304. doi:10.1109/TPAMI.2016.2646371

4. Thobhani A(1), Gao M(1), Hawbani A(2), Ali STM(3), Abdussalam A(4). CAPTCHA recognition using deep learning with attached binary images. Electronics (Switzerland). 9(9):1-19. doi:10.3390/electronics9091522

5. Character-Based Handwritten Text Transcription with Attention Networks. 2017. Accessed February 2, 2021. https://search-ebscohost-com.login.ezproxy.library.ualberta.ca/login.aspx?direct=true&db=edsoai&AN=edsoai.on1106281594&site=eds-live&scope=site

6. https://en.wikipedia.org/wiki/Recurrent_neural_network

7. Brownlee J., Encoder-Decoder Long Short-Term Memory Networks. https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/