Group 06-No.1: STAT 430 Progress Report

Shubo Sun
UIUC Statistics Department
Netid: shubos2

Yizhen Jia
UIUC Economics Department
Netid: yizhenj2

Vaughn Ellison Hilpp
UIUC Statistics Department
Netid: vhilpp2

***Abstract*—We investagted the problem of text-based captchas recognition, a popular research direction to create AI to attack current captcha systems or to build adversarial networks as a repairment to protect webpages. In this paper, we propose our CRNN network for text-based captchas by training on $100,000$ generated captchas with a prediction accuracy of 87.5% on the testing data. Note that this report is an old version only includes some loss graphs and sample output. To see a detailed description of the network, please go to "Captcha_CRNN.ipynb"**

I. Introduction

The rise of the internet in the late 1990's and early 2000's radically changed the way business was conducted and how humans interacted writ large. Just as in person business has forever needed measures of security, the digital marketplaces of the internet are no exception. One of the first such online security systems was CAPTCHA, an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart". CAPTCHA requires internet users to disseminate information such as text or images in a way to differentiate humans and computers. The first forms of CAPTCHA were distorted alphanumeric characters that the user needed to correctly identify and type out. Later implementations included image identification.

Hackers and fraudsters do not rest and are constantly inventing clever machine learning ways to bypass online security such as CAPTCHA. If academic researchers can come up with a way to beat the current iteration of CAPTCHA, it signals a time for change in the type of test being used. This is the reason alphanumeric tests were replaced with image tests as they are more challenging to solve. Our motivation for solving CAPTCHA is to further identify the shortfalls of the current CAPTCHA implementations in order for developers to patch the metaphorical holes in the system to stop bot attacks and fraud.

The input to our network will be a five character, alphanumeric image with typical CAPTCHA distortions such as rotated characters, noise (fuzziness) and limited to no whitespace in between characters. The output will be the predicted 5 characters.

II. Related Works

Existing approaches for text captcha recognition primarily focus on CNN and RNN models. Nouri and Rezaei [1] performed a Deep-CAPTCHA solver consisting of three Convolutional Maxpool layers, followed by a flattern layer and novel parallel Softmax layers. [2]–[4] relied on DCNN-based networks, and Chen et al. [5] introduced a Selective Learning approach by improving the recognition accuracy of confusing classes in the confusion-class DCNN. Shi et al. [6] proposed an RCNN network containing Convolutional layers to extract feature sequences from the input images, predicted a label distribution for each feature sequence based on Deep Bidirectional LSTM network, and translated the predictions into label sequences with a transciption layer.

In addition to text-based captchas solving, image-based captchas solving has become one of the most popular fields these years, see [7]–[9] for CNN based methods to crack CAPTCHA images. Another interesting field would be adversarial CAPTCHA generation algorithms, see Ye et al. [10] for a proposed GAN network to target a wide range of features of text CAPTCHAs and protect text CAPTCHAS by building a base solver with training on very few real CAPTCHAs and refining the base solver using transfer learning.

In this report, we created a CRNN network followed by ideas in [6] and [11] to correctly identify the characters in text CAPTCHA images. We generated $100,000$ image CAPTCHAs using [16], and designed the model based on codes from [11]–[16]. Different from these GitHub or Kaggle pages, where [11], [13] used a less dense CRNN network (less than 6 layers) with CTC loss, [15] fitted a denser CRNN model but is still different from ours in multiple ways (e.g. smaller hidden layer size, only one LSTM layer, only works for a small amount of data, SGD optimizer, etc.), [12], [14], [16] used pure CNN based models with categorical cross-entropy loss, our CRNN model [17] is the densest one (16 Convolutional layers, 2 GRU layers and 2 linear layers) with ADAM optimizer and CTC loss. Note that the idea of our CRNN model and the optimization design follow the illustration in [11]. Our model could work for more than $100,000$ images while maintaining the prediction accuracy. We will discuss the model and the optimization in detail for the following sections.

III. The Data

The data we are working with are alphanumeric CAPTCHA images. These images were generated by implementing Python's "captcha" library, specifically using the function "ImageCaptcha" which generated CAPTCHA style images

based on input characters. The resolution of the images is 368 x 158 pixels.

Each image has a width of 50 and a height of 135 and is composed of three layers (RGB). Data preprocessing included batch normalization and the first iteration of our model used a simple 80-20 train-test split.

A. Special Steps

Special steps: Our loss function will be Connectionist Temporal Classification or CTC loss. This lets us compare a sequence or inputs to a sequence of outputs rather than one input to one output at a time. Using this loss function, there is a classification being made at each step in the input sequence, one step being a column of pixels within the CAPTCHA image. The classification at each step can be one of any of the 55 alphanumeric characters plus the addition of "-" to signal the classifier was unable to make a prediction. The addition of this dash is the special step.

B. Data examples

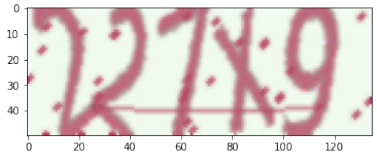

Fig. 1. Five character CAPTCHA IMAGE (227X9)

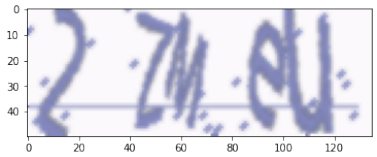

Fig. 2. Five character CAPTCHA IMAGE (2ZMeu)

IV. Model Architecture

A. Overview of hyper parameters

Our model has 16 Convolutional layers, each followed by a batchnorm layer. Next we have fully connected, with two GRU layers to follow that. Finally, there is one last fully connected layer that is fed to our CTC loss function. After each convolutional and batchnorm layer we have a ReLU activation function in place. Our inaugural model ran over 20 epochs, on batch sizes of 16. We had a learning rate of 0.001 with weight decay of 0.001 as well. For convolutional layers, a 3x3 kernel matrix on all but two layers. One convolution layer downsamples the data by using a 1x1 kernel with a stride of two. The final convolutional layer uses a 3x6 kernel matrix. Padding is set to one for all convolutional layers. The batchnorm layers all used a momentum value of 0.1. Note that some convolutional layers in our model come from a pretrained Resnet18 architecture.

B. How It Works

Our network combines the architectures of Convolutional Networks (CNN) and Recurrent Neural Networks (RNN). Our network consists of 16 convolutional layers followed by a linear layer to transition to the RNN portion of the network by implementing two Gated Recurrent Unit (GRU) layers. The convolutional layers extract a feature sequence from the input image by applying a kernel matrix over the image. Recurrent layers predict a label distribution for each frame by a fully connected linear layer, which translates the per-frame predictions into the final label sequence. This follows the idea of Shi et al. [6].

C. Loss Function

Our loss function for this model is Connectionist Temporal Classification or CTC loss. For this loss function, there is a classification being made at each step in the input sequence, one step being a column of pixels within the CAPTCHA image. The classification at each step can be one of any of the 55 alphanumeric characters plus the addition of "-" to signal the classifier was unable to make a prediction. We choose this loss function because it lets us compare a sequence of inputs to a sequence of outputs rather than aligning each character. The ability to predict a blank space in the input is also particularly useful for CAPTCHA images, which often have irregular formatting.

V. Preliminary Results

Fig. 3. shows a summary table of inputs, predictions and correctly formatted predictions of CAPTCHAs from testing data. Fig. 4. contains loss graphs over epochs and all images, and Fig. 5. presents our training and testing accuracy.

	true	prediction	correct_pred
0	BPXuN	BPX-uN	BPXuN
1	edRQ7	edR-Q7	edRQ7
2	tjztM	t-jZtM	tjZtM
3	bFFyc	bFFy-C	bFyC
4	GugBE	Gug-8E	Gug8E
...	...	...	...
995	vwfjQ	vwffjQ	vwfjQ
996	kRaEJ	kRB-EJ	kRBEJ
997	ZJhcu	zJh-Cu	zJhCu
998	XzHPn	xz-HPn	xzHPn
999	GBc5r	GBcc5r	GBc5r

Fig. 3. Table of Inputs, Predictions and correctly formatted predictions

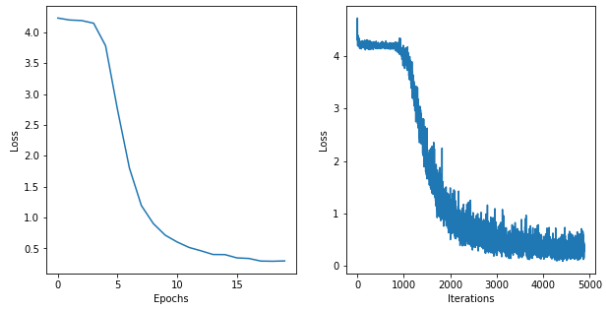

Fig. 4. Loss Graph over Epochs and Iterations

```
Train accuracy 0.7085
Test accuracy 0.42200000000000004
```

Fig. 5. Initial Train and Test Accuracy

VI. Project Progress

The chart Fig.6. is our projects Gantt chart. The link here takes you to our Colab notebook with our model.

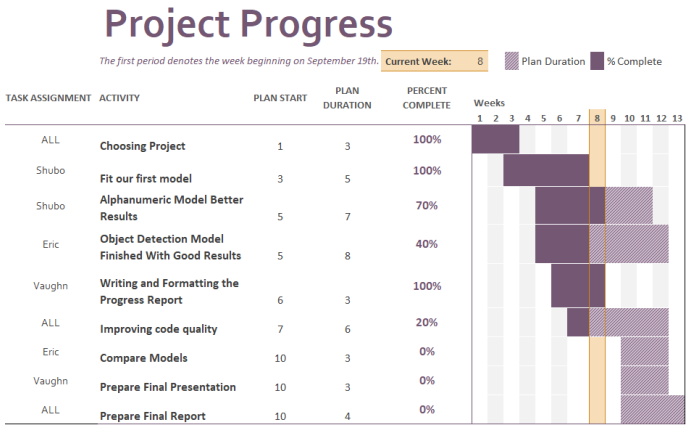

Fig. 6. Gantt Chart

A. What's To Come

The "arms race" between website holders and robots never ends. Beyond the simple alphanumeric CAPTCHAs mentioned above, reCAPTCHA v2, developed by Google, also plays an important role in website security. The final goal of our project is to adopt the object technique to solve the image puzzles in the reCAPTCHA v2. To have smooth project progress, we decide to first implement the original YOLO v1 model on the alphanumeric CAPTCHAs to test the basic framework and algorithm (Joseph, et al.). By training on the 8 classes one-digit number-only CAPTCHAs (generated and labeled by ourselves), YOLO v1 shows a pretty good result (Fig.7).

However, the alphanumeric CAPTCHAs may not need such a complex model, while image puzzles (i.e., reCAPTCHAs) may need a more complex model to obtain better performance. In the future, we will make some significant changes and split our model to adapt to different CAPTCAHs. For the first part, we are going to make predictions on the 5-digit alphanumeric CAPTCHAs. For the second part, we will use the pre-labeled images for specific classes in the reCAPTCHAs v2 from Open Images Dataset as training data to solve the reCAPTCHAs.

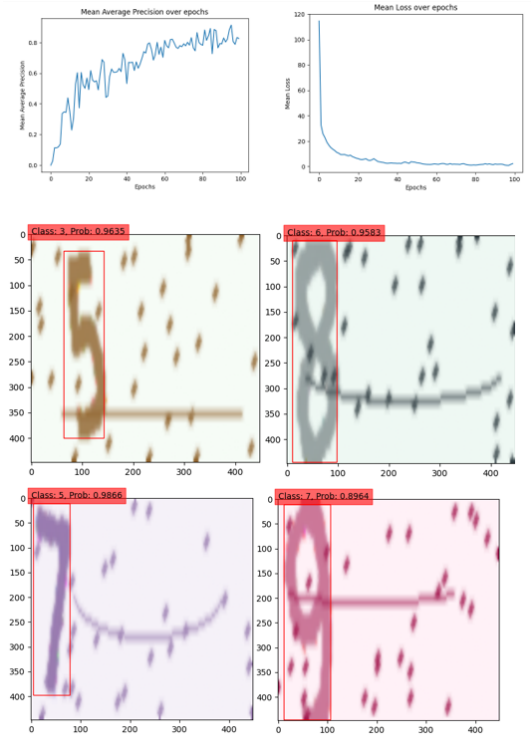

Fig. 7. Initial Train and Test Accuracy

References

[1] Z. Noury and M. Rezaei, "Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment," arXiv:2006.08296, June 2020.

[2] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow CAPTCHAs," in Proc. ACM Conf. Comput. Commun. Secur., 2013, pp. 1075–1086.

[3] E. Bursztein, M. Martin, and J. Mitchell, "Text-based CAPTCHA Strengths and Weaknesses," in Proc. ACM Conf. Comput. Commun. Secur., 2011, pp. 125–138.

[4] K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (HIPs)," in Proc. Adv. Neural Inf. Process. Syst., 2004, pp. 265–272.

[5] J. Chen, X. Luo, Y. Liu, J. Wang and Y. Ma, "Selective Learning Confusion Class for Text-Based CAPTCHA Recognition," in IEEE Access, vol. 7, pp. 22246-22259, 2019, doi: 10.1109/ACCESS.2019.2899044.

[6] B. Shi, X. Bai and C. Yao, "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition," arXiv:1507.05717.

[7] Karthik, CHBL-P., and Rajendran Adria Recasens. "Breaking microsofts CAPTCHA." Technical report (2015).

[8] Kopp, Martin, Matej Nikl, and Martin Holena. "Breaking captchas with convolutional neural networks." ITAT 2017 Proceedings (2017): 93-99.

[9] Zhao, Nathan, Yi Liu, and Yijun Jiang."CAPTCHA Breaking with Deep Learning." (2017)

[10] Ye et al., "Using Generative Adversarial Networks to Break and Protect Text Captcha", ACM Transactions on Privacy and Security, Volume 23, Issue 2, May 2020, Article No.: 7, pp 1–29, https://doi.org/10.1145/3378446.
[11] https://actamachina.com/notebooks/2019/03/28/captcha.html.
[12] https://github.com/shishishu/pytorch-captcha-recognition.
[13] https://github.com/abhishekkrthakur/captcha-recognition-pytorch.
[14] https://www.kaggle.com/shawon10/captcha-recognition.
[15] https://github.com/GokulKarthik/Deep-Learning-Projects.pytorch/blob/master/5-Captcha-Text-Recognition-With-CRNN.ipynb.
[16] https://github.com/DrMahdiRezaei/Deep-CAPTCHA/blob/master.
[17] https://github.com/shubosun0113/STAT-430. This is the GitHub repository to save the code. The data is too large (729 MB) to upload.

Lastly Participation:

Vaughn Hilpp: 31%, writing the progress report and searching for papers

Shubo Sun: 35%, fitting the CRNN model

Yizhen (Eric) Jia: 34%, fitting the Yolo v1 model