Development of a Model for Reading CAPTCHA Characters

Team : ToNight
DL project of DAY

1 Team details

Jeongwoo Kang	whitdisa03@gmail.com	Project Management
Yunseok Jeon	seok220hun284@yonsei.ac.kr	Project Design & Report Writing
Junseob Ko	jcgo0845@yonsei.ac.kr	Implementation & Model Tuning

2 Project Statement

The project aims to develop an advanced model capable of recognizing and interpreting characters in CAPTCHA input. By overcoming the challenge of distinguishing between distorted or obscured characters, which serve as an automated entry prevention mechanism on websites, this model seeks to push the boundaries of current AI capabilities in pattern recognition.

3 Motivation for solving this problem

As artificial intelligence progresses, its ability to interpret complex visual data has seen significant advancements. This project is motivated by the potential to leverage these AI advancements for automating the solving of CAPTCHA mechanisms, traditionally designed to deter automated access. Solving CAPTCHAs automatically could significantly benefit various fields, including accessibility for users with disabilities, testing of automated services, and improving AI's understanding of distorted or obscured texts and images.

4 Specific Objectives

4.1 Task Description

Input : Distorted character images
Output : Original characters

4.2 Dataset

- MNIST database(dataset for single-digit numbers)
- EMNIST database(dataset for single characters and single-digit numbers)
- Easy CAPTCHA datasets for alphabet sequences
- CAPTCHA datasets in three levels

4.3 Approach

- Curriculum Learning 1 : Train the model using individual characters to enhance its understanding of characters. After training the model with individual characters, further train it to enhance its ability to

recognize words. To recognize distorted words, additional training with relevant datasets is necessary.

- Curriculum Learning 2 : Collecting diverse CAPTCHA image datasets through Kaggle. CAPTCHA images are classified into three main stages based on the level of recognition difficulty. The first stage focuses on relatively simple alphabet recognition, while the second stage transitions to word formation recognition. In the final third stage, challenges arise with highly complex CAPTCHA images due to significant variations in word arrangement, background noise, and overlapping characters. Utilizing CAPTCHA datasets from each stage, models are trained incrementally, allowing them to progressively develop the ability to recognize more complex patterns and variations.

5 Results expected

Upon successful completion of the project, we anticipate the following outcomes:

- Development of an artificial intelligence model capable of accurately recognizing characters in CAPTCHA inputs.
- Enhanced efficiency in automated CAPTCHA solving, potentially reducing human intervention in accessing websites with CAPTCHA mechanisms.

6 Implementation

6.1 Data Processing

Before we start, we made three subsets of our datasets for 3 purposes:

- training dataset to train our model,
- validation and test dataset to evaluate our train model.

6.2 Parameter Training

We will use the following hyperparameters:

- Cross Entropy as our loss function, since it works well for multi-class classification,
- Adam as our optimizer.

In our training loop we will call images and their labels from the directory for training data at each batch, calculate loss, update our hyperparameters by using back propagation, and repeat this process for the number of epochs.

Our hyperparameters of our trained model will be saved in our weights directory. These hyperparameters will be reused in our evaluation model.

We will call images and their labels from the directory for validation data, and the model will try to predict what the CAPTCHA image's label is. Another function will be used to compared the predicted and actual label, calculate the accuracy of the model, and return it.

By switching the batch size, learning rate, and number of epochs we will find the hyperparameter values that has the highest concordance rate.

7 Results

7.1 ResNet-18 Based Model

We started with a model based on ResNet-18, which is long used and has a low computational cost. We took 6 different tests, each test had made differences in the number of epochs, learning rate, and batch size. The following table shows the accuracy of each test.

	Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
A	150	0.001	16	850	0.7850
B	90	0.001	16	850	0.7009
C	90	0.001	8	850	0.7850
D	90	0.001	16	850	0.8505
E	90	0.003	16	850	0.5421
F	90	0.0005	16	850	0.7850

Above the others, test D, which increased the batch size to 32, shows the highest Accuracy.

Figure 1: **Visualized result of test D**

We did not expect the visualized data to expand from the original image as shown in figure 1, which may reduce the accuracy. Therefore we added padding so that the size of our data could be fixed. The following shows the accuracy after padding is implemented, maintaining the hyperparameters used in test D.

Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
90	0.001	32	850	0.8785

7.2 ViT Based Model

ViT is excellent at learning the global features of the given images, which may make it a better model for interpreting CAPTCHA images. We attempted to learn and interpret CAPTCHA images using a ViT based model.

Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
90	0.001	16	850	0.0000

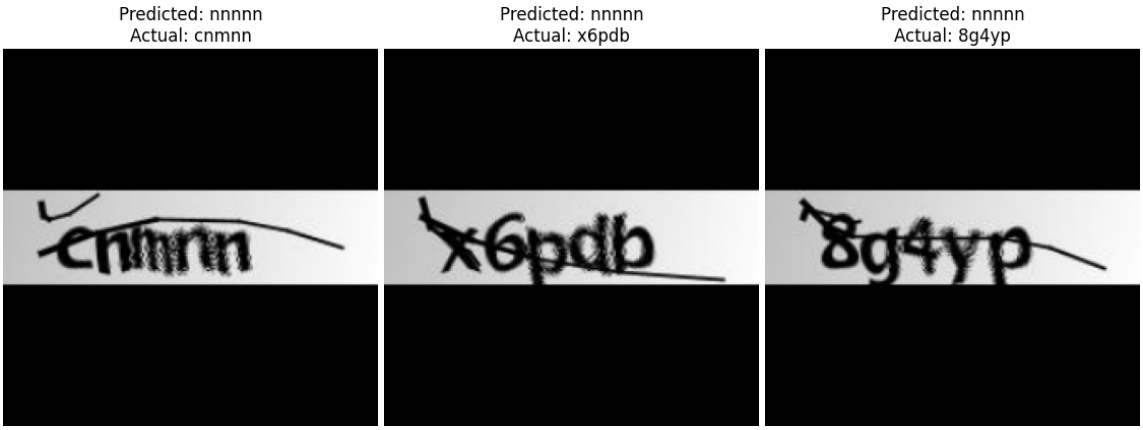

Figure 2: **ViT based model**

However, this model failed in interpreting the test data. Since ViT requires a lot of data, our data size might have been smaller for ViT. Therefore we decided to keep going on using our ResNet-18 based model, and give differences from a different perspective.

7.3 Data Augmentation

Data augmentation is another method to increase the accuracy of the model by artificially giving diversity to the original dataset and learn a diversity of situations. By implementing modification functions we can give random modifications to the original image files in angle, color, filtering, and distortion.

Applied Modifications	Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
Rotation	90	0.001	32	850	0.7290

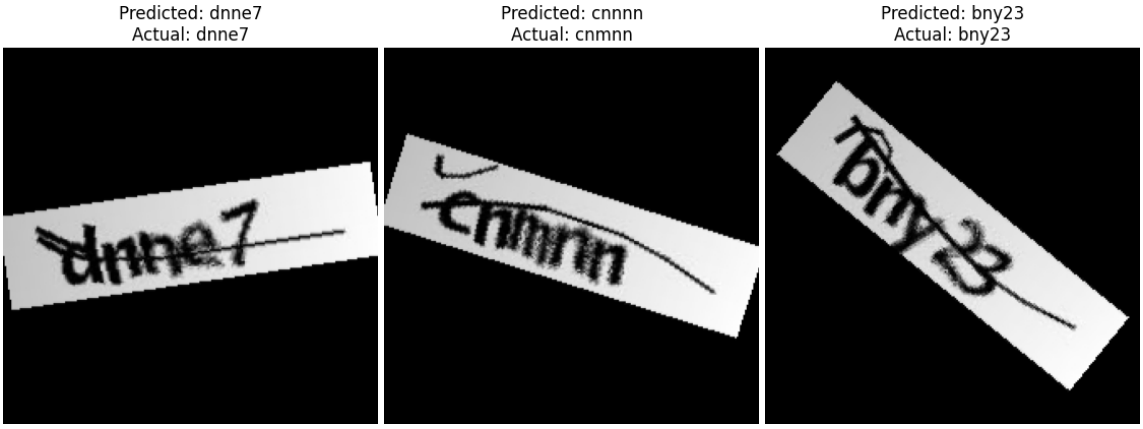

Figure 3: **Augmentation of datasets by applying rotation to the original images at random**

Applied Modifications	Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
Color Change	90	0.001	32	850	0.8972

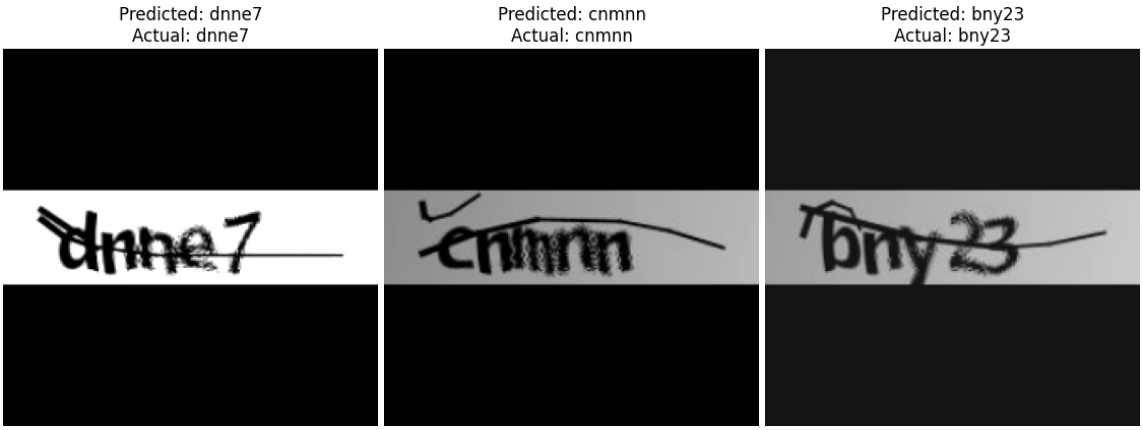

Figure 4: **Augmentation of datasets by applying change of color to the original images at random**

Applied Modifications	Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
Box Filtering	90	0.001	32	850	0.8131

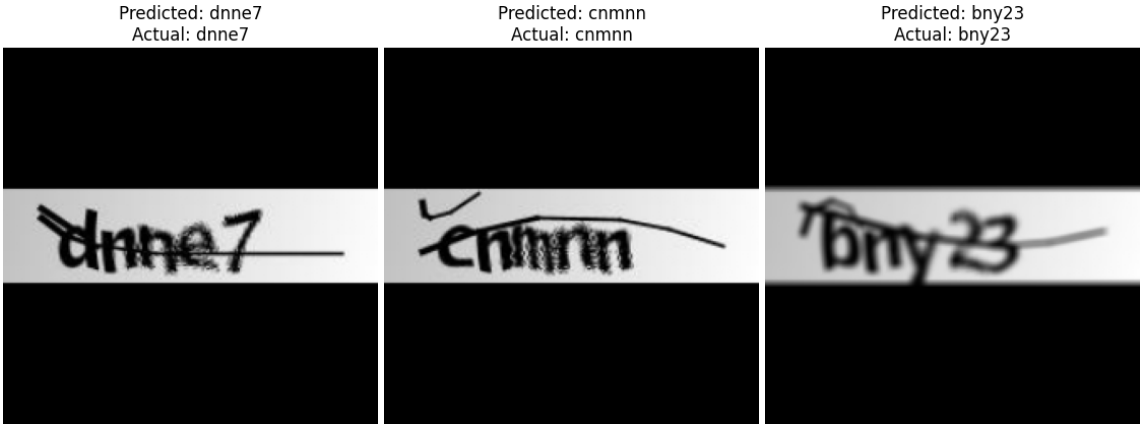

Figure 5: **Augmentation of datasets by applying box filtering to the original images at random**

Applied Modifications	Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
Perspective Distortion	90	0.001	32	850	0.6636

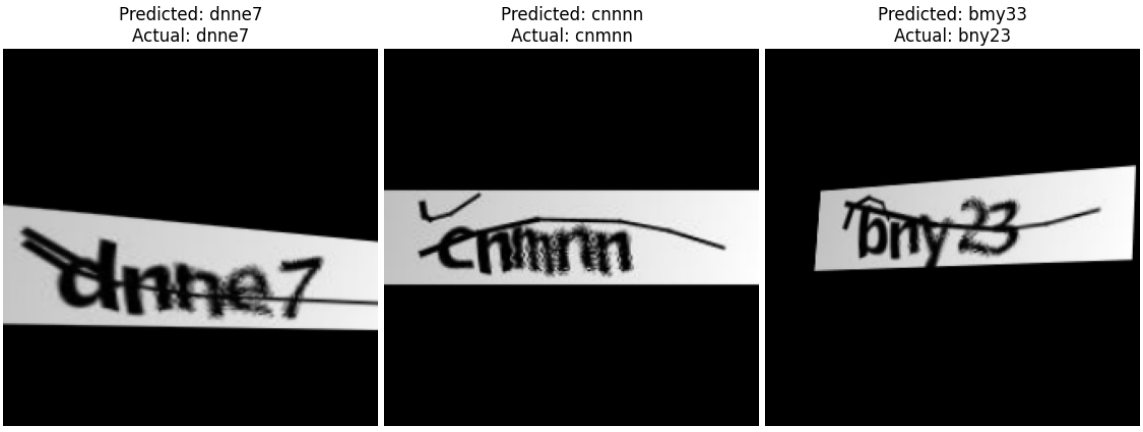

Figure 6: **Augmentation of datasets by applying perspective distortion to the original images at random**

Applied Modifications	Epochs	Learning Rate	Batch Size	Dataset Size	Accuracy
All mentioned above	90	0.001	32	850	0.5234

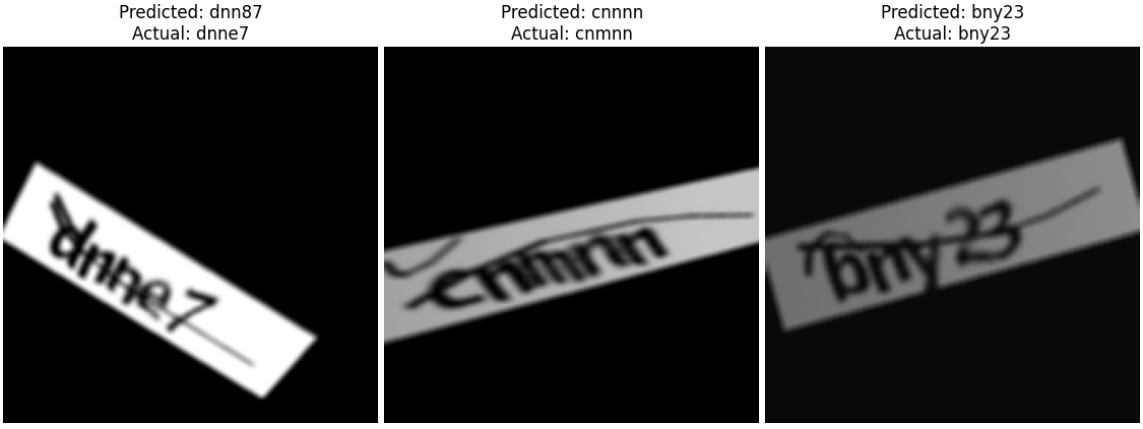

Figure 7: **Augmentation of datasets by applying all of the variations to the original images at random**

8 Discussion

Upon completing our project on recognizing CAPTCHA images, we have identified several directions for further improvement and exploration:

- Increase Data Volume : To enhance the robustness and accuracy of our model, it is crucial to increase the volume of training data. A larger dataset would allow the model to learn a wider variety of CAPTCHA patterns and improve its generalization capabilities.
- Dynamic Character Count Recognition : Currently, our model is designed to recognize CAPTCHAs with a fixed number of characters. Future work should focus on developing a more flexible model capable of recognizing CAPTCHAs with a variable number of characters. This would involve modifying the architecture to handle sequences of different lengths and potentially implementing a more advanced decoding mechanism.

- Training on Different Types of CAPTCHA Images : To make our model more versatile, we should train it on different types of CAPTCHA images, including those with varying fonts, colors, and backgrounds. This would involve collecting and annotating a diverse set of CAPTCHA images and potentially using data augmentation techniques to simulate different CAPTCHA styles.

- Exploring Alternative Models : While our current model has shown promising results, exploring other model architectures could yield even better performance. This could include experimenting with more advanced neural network architectures, such as transformers or attention-based models, which have shown success in other image recognition tasks.

References

[1] Keras, *Captcha OCR Example*, https://keras.io/examples/vision/captcha_ocr/, Accessed: 2024-06-27.

[2] Paul Fournier, *Captcha Version 2 Images Dataset*, https://www.kaggle.com/datasets/fournierp/captcha-version-2-images, Accessed: 2024-06-27.