Machine Learning and Data Mining II
Final Project - Captcha recognition using convolutional neural networks

Nguyen Hai Dang - 22BI13073
Nguyen Quang Huy - 22BI13195
Nguyen Minh Tuan - 22BI13447
Bui Dang Quang - 22BI13378
Nguyen Tuan Khai - 22BI13202

University of Science and Technology of Hanoi

Abstract

In this paper, we present the research results on the recognition of algorithmically generated text - based CAPTCHA tests using Convolutional Neural Networks (CNN). Text - based CAPTCHA's are used as a security measure used on websites to differentiate between human users and automated bots. However, their effectiveness depends on their resistance to sophisticated recognition techniques, meaning that for many cases CAPTCHA tests can be bypassed using a model that can capture the patterns of the text images. To this end, we first performed various image processing techniques to extract the necessary data, followed by using CNN as the base model for the learning process and to make predictions. To optimize the model accuracy, we also performed hyperband tuning on the model using the TensorFlow framework, and devised a form of the Ensemble Voting Method to aggregate the predictions from multiple CNN models.

1 Introduction

CAPTCHA

CAPTCHAs (Completely Automated Public Turing test to Tell Computers and Humans Apart) are algorithmically generated security measures designed to prevent automated access to web services. In the context of text - based CAPTCHAs, the user is presented with a picture containing a string of text with varying degrees of introduced noise and distortion (in the form of blur/dashed line/missing strokes) intended to make it easy for humans to interpret and solve but difficult for machines. However, as per the case presented in this research, recent advancements in artificial intelligence have rendered traditional CAPTCHAs increasingly susceptible to automated system bypassing.

CNN

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms generally used for processing and analyzing visual data such as images or videos. Inspired by the organization of visual cortex in the human brain, their distinctive architecture leverages spatial hierarchies and local patterns within the data, enabling the automatic learning of complex and abstract features. The use of CNNs to recognize distorted characters in CAPTCHAs has revealed a vulnerability in the security protocol, emphasizing a need for more sophisticated methods that can be resilient to pattern recognition.[1]

Hyperband tuning

Hyperband tuning is a hyperparameter optimization method that combines random search and early stopping to efficiently find the most optimal set of hyperparameters for the model. It maximizes the utilization of computational resources as well as accelerates the searching process. The use of hyperband tuning entails iteratively allocating resources to different subsets of hyperparam configurations, beginning with a large number of configurations evaluated for very few resource allocations. Based on their performance, it selects the top - performing configurations and dispense more computational power for them. This process is repeated until only one possible combination of hyperparameters remain or the computational resources are exhausted.

Voting Ensemble

Ensemble methods in general comprise a collection of machine learning techniques focusing on integrating multiple models to enhance predictive capabilities that would not otherwise be achieved by using individual models. The core concept of ensembling is to utilize the diversity of the various models employed, which can help to clear bias and increase overall stability while reducing prediction error. It would also help to reduce the limitations of individual models, resulting in more accurate and robust outputs. For our research, we will be employing the technique of using voting ensemble across different learners of the same model.

1.1 Image Processing + Data Analysis

The first figure illustrates a random sample element in our dataset.

Figure 1: Sample images from the dataset

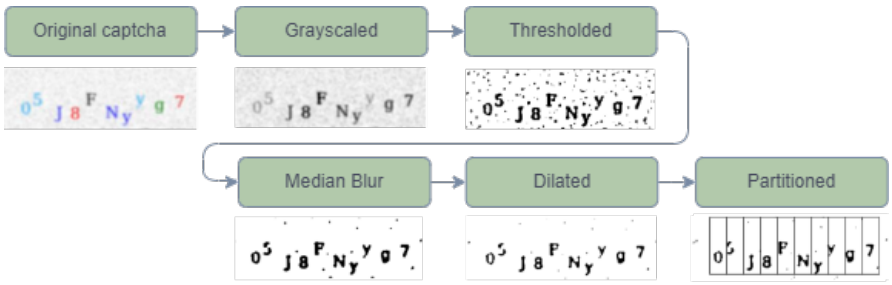
As seen in the figure above, the CAPTCHA has a lot of blur and background noise, and the characters are in random alignment and varying colors. These configurations lend themselves to the added complexity for the automated generators, consequently augmenting the security of the system. The diversity in distortion and noise levels creates a challenge for the model's adaptability. Another unique problem presents itself in the length of the CAPTCHA, as each image contains 10 characters. In order to achieve a correct output, the model has to correctly predict all 10 characters individually.

The original dataset contains a total of 10000 images in the JPG format [2]. For the purposes of our work we'll only be working on 1500 of these images from the dataset, extracted at random using Python. The images are all of the same dimension, 282 pixels in width and 90 in length, with 3 channels of color. They contain 10 alphanumeric characters that may include both upper and lower case letters of the English lexicon as well as numbers.

1.1.1 Image Processing Pipeline

Each image of both the training and testing set follow the same processing procedure:

1. Convert image from the BGR color space to grayscale.
2. Perform adaptive thresholding to create a binary image.
3. Apply median blur to remove noise.
4. Perform dilation.
5. Partition the image into 10 sections of the same size, each assigned a label corresponding to its character.



As seen in Figure 2, after the procedure some of the character images have become disfigured and illegible. However, most of the data instances were still preserved so our model should be able to learn and adapt to this data without issue.

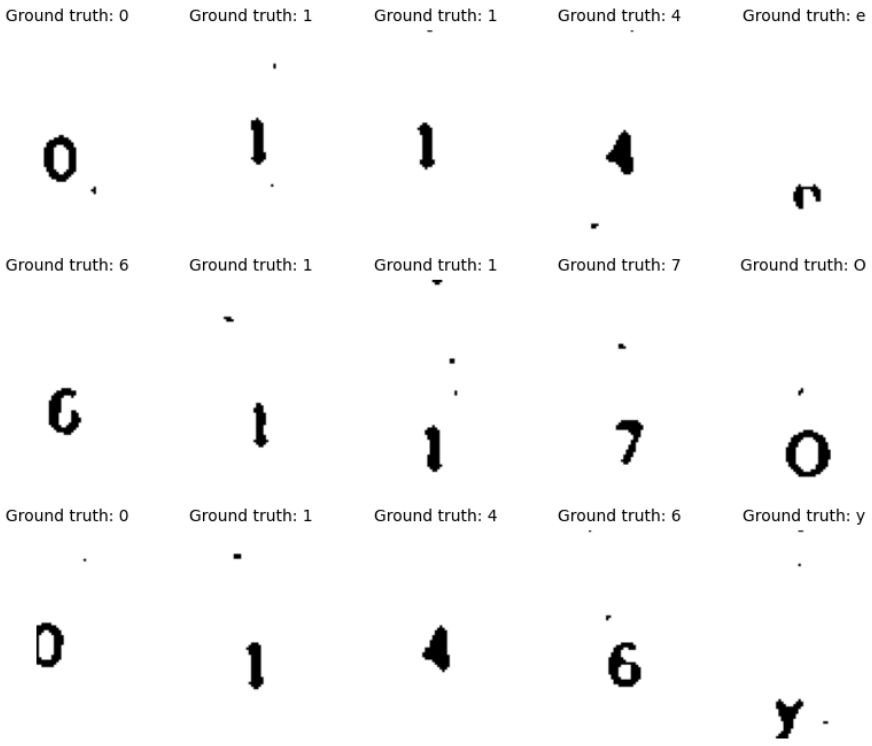


Figure 2: Partitioned captcha with their respective labels

1.1.2 Data Analysis + Handling Imbalanced data

As it stands, the distribution of labels in our dataset is imbalanced, with certain characters appearing a lot more in both the training and testing set. Figure 3 illustrates this problem more clearly.

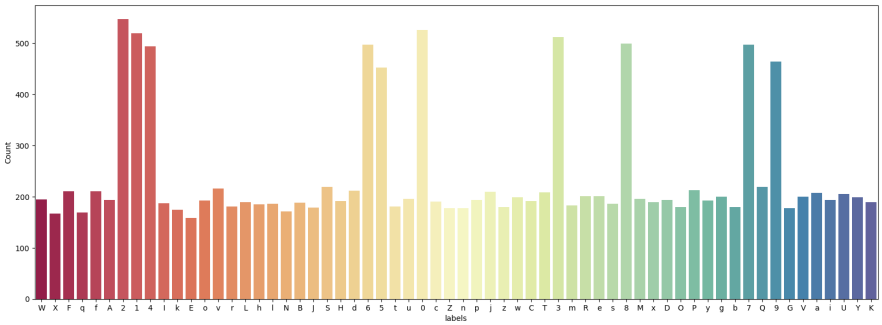


Figure 3: Label distribution in captcha dataset

To handle the imbalance, we implement a data augmentation technique called SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic examples of the minority class. Developed by Chawla et al. [4], the process involves generating new training data points via interpolation between 2 existing instances with the same ground truth label.

Algorithm 1 SMOTE

1: **Input**: Minority class samples, number of desired synthetic instances (N), number of nearest neighbors (K)
2: Compute Euclidean distance between each minority class sample and all other minority class samples to identify the K nearest neighbors for each sample
3: For each sample: Randomly select 1 of K neighbors. Compute difference vector between the minority class sample and neighbor. Multiply the distance by a random $x \in [0, 1]$. Add the resulting vector to the minority class sample to create new synthetic instance.
4: Repeat step 3 N times for N synthetic instances.
5: **Output**: Original minority class now has N synthetic instances.

By generating new samples, we can balance out the label distribution and improve the overall model performance. After applying SMOTE we arrive at a training set with 500 samples for each character class.

1.2 Model

Figure 4 shows the architecture of the model that we utilized for the CAPTCHA prediction, with an input shape of 85x25x1 for the partitioned pictures.

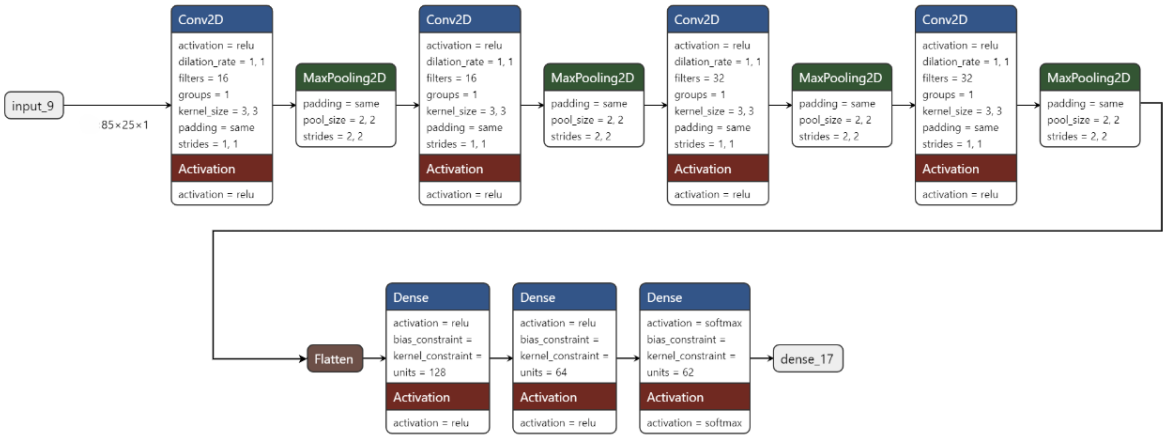


Figure 4: Representation of the layers

The model features 4 Conv2D layers responsible for the convolution operation, which is central to the concept of CNN. They detect feature maps and patterns in the input data, and each layer is followed by a MaxPooling2D layer for feature extraction and dimensionality reduction. The output is flattened and channeled into a series of Dense layers responsible for the actual learning task of the model. The last Dense layer uses the softmax activation function, which calculates probabilities for each possible character. This is the model architecture we use for the initial fitting, without bias or prior knowledge of the data to get an apt first impression. Later iterations of the model also included BatchNormalization layers for improved stability and Dropout layers to prevent overfitting.

Figure 5 illustrates the input data through each layer.

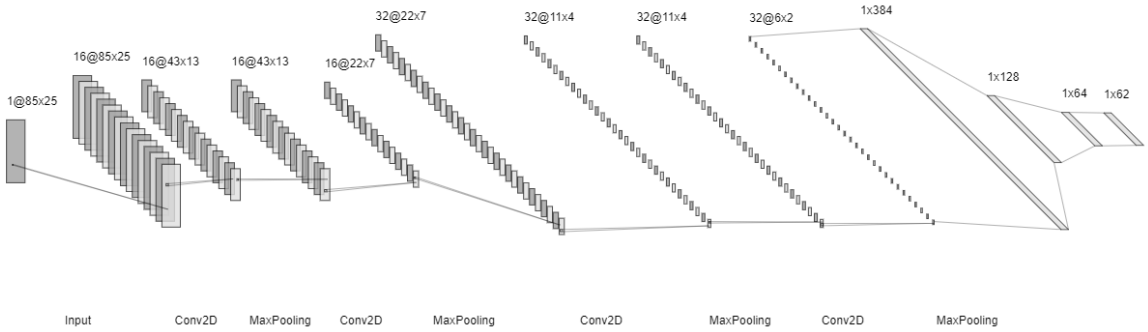


Figure 5: Representation of the input data through each layer

The model's performance was evaluated through multiple learning sessions, and during each of them we introduced 2 callback mechanisms to prevent overfitting and preserve the best performance for that particular session:

- ModelCheckpoint(): allows us to save the model's weights during training, essentially preserving the model at specific intervals or when certain conditions are met. For our purposes we'll be saving the model that achieved the highest validation accuracy.
- ReduceLROnPlateau(): dynamically reduces the learning rate when the model's performance reaches a plateau, which can help the model overcome small oscillations or avoid getting stuck in suboptimal solutions.

1.3 Voting Method

One of the most intuitive and most prominent ensemble methods is the plurality voting method [3]. In this approach, multiple model instances are trained on the same dataset. These trained models are then inferred to new data points to make predictions, and the final output is achieved by aggregating the individual model's predictions using specific voting schemes. For our multi - label classification task, plurality voting simply means the final prediction is the class (or character) that receives the most votes from the individual models.

2 Research process and results

2.1 Performance evaluation for initial fitting

Due to the use of the model checkpoint mechanism, the accuracy of the model on the validation set appear to have a steep dropoff at certain intervals. This detail however is of no consequence to our training process, and when observing the plot, we see that we were able to reach a 95.65% accuracy on the initial fit without any alterations to the model.

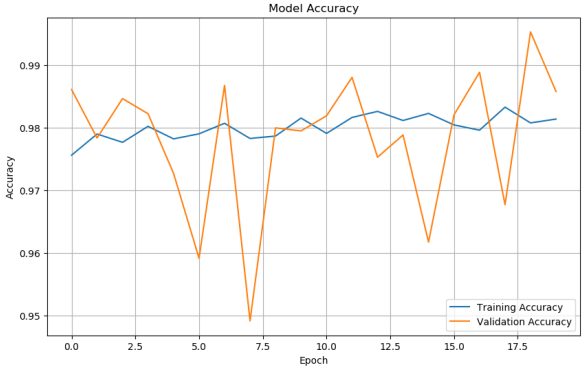


Figure 6: Accuracy by Epoch for the initial fit

2.2 Hyperband tuning

To ensure that our CNN model can attain the best accuracy, we conducted an extensive hyperparameter optimization process by employing the hyperband tuning technique via the keras_tuner function from the tensorflow.keras module. This structure allows us to systematically explore the hyperparameter space we defined, and identify the best combination for our model. The following hyperparameters and their respective candidate values were included in our search space:

- Batch size: 32
- Epoch: 40
- Activation function for the convolutional layers: ReLU
- Number of neurons used in the first dense layer: [200, 750], step = 200
- Number of neurons used in the second dense layer: [30, 200], step = 20
- Dropout rate of the convolutional layers: 0.1, 0.3, 0.5
- Dropout rate of the dense layers: 0.1, 0.3, 0.5
- Optimizer: Adam, RMSprop, SGD, Adamax
- Learning rate: 1e-2, 1e-3, 1e-4

The hyperband tuning process was conducted over 90 trials, and included additional dropout and batch normalization layers to ensure a reliable estimate of the model's performance as well as reduce the risk of overfitting. Table 1 shows the best performing hyperparameter combination, which we will implement for all of our models moving forward.

Optimizer	Conv Dropout	Dense Dropout	Neurons first Dense	Neurons second Dense	Learning rate	Score
Adam	0.1	0.1	350	30	0.001	0.979

Table 1: The best model with the hyperparam combinations using hyperband tuning

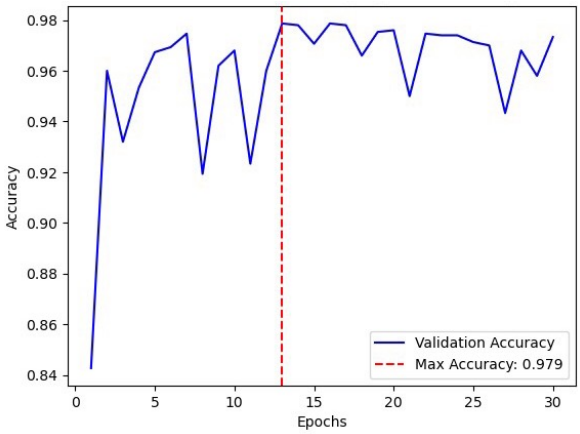


Figure 7: Model Accuracy by Epoch using optimal hyperparameter combination

2.3 Voting Ensemble performance

By employing the plurality voting scheme, the ensemble approach effectively reduces classification error, increase model stability and enhances generalization capabilities. To this end we trained 20 separate models that do not differ from one another in the model architecture or the hyperparameter combination; instead the differences between them arise from their training data and thus the individual

training process they undergo. We split the training dataset into 20 equal subsets, and trained each of the model on 19 of the 20 subsets (which comes out to about 29450 images). Table 2 presents the results of aggregating the 20 models' predictions, where for each iteration the number of models used incremented by 2. The models' predictions were based on the same dataset.

From the analysis of Figure 8, it can be observed that there is a noticeable improvement on the model performance after ensembling; whereas before we were only achieving 97.9% accuracy by character which converts to a 80.87% accuracy on the entire CAPTCHA series, now that figure has been raised to 85.7%. This clearly indicates that the ensemble of models can effectively leverage the strengths of weaker individual models to gain better results. But as the number of model increases beyond 10 - 12 we can see a steep dropoff. suggesting that beyond the designated point there may be diminishing returns even to the point of harming the effectiveness. This phenomenon is more commonly known as "ensemble divergence" or "ensemble disagreement", occuring when there is lack of complementary strength for larger number of models. It may be the case that as more and more learners are ensembled, similar or correlated mistakes appear more often. Therefore for utility purposes we would only perform ensembling on a lower number of models.

Models	4	6	8	10	12	14	16	18	20
Score by each character	0.984	0.983	0.984	0.985	0.985	0.984	0.984	0.984	0.983
Score across entire CAPTCHA	0.851	0.845	0.851	0.857	0.857	0.851	0.851	0.851	0.845

Table 2: Performance comparison of plurality voting function across varying number of models

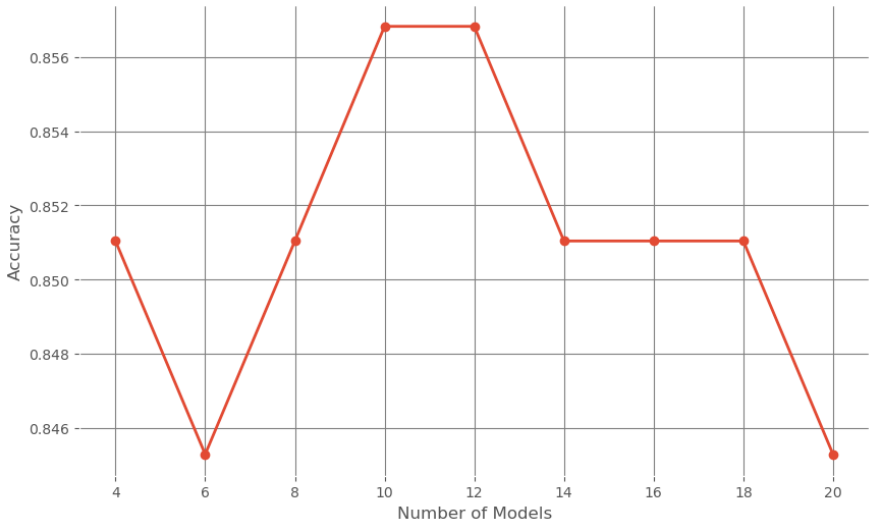


Figure 8: Model accuracy on full CAPTCHA series after ensembling

3 Conclusions

In conclusion, our research results suggested that with proper employment of batch size, number of neurons in the penultimate dense layers (prior to softmax activation) and dropout rate, we can reliably improve the performance of our CNN model. We were able to identify these values by utilizing the hyperband tuning technique, leading to better and more robust model performance. Furthermore, the research also clearly illustrates the significance of voting ensemble schemes in improving model stability and reaching high accuracy altogether. The results are very promising, and a clear indication that text - based CAPTCHAs are no longer machine proof, necessitating a more complex security measure to ensure website safety from unwanted trespass.

References

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. (2018). *Recent advances in convolutional neural networks.*

[2] Aadhav Vignesh: CAPTCHA Images. URL: https://www.kaggle.com/datasets/aadhavvignesh/captcha-images/data

[3] Florin Leon, Sabina-Adriana Floria and Costin Bădică. (2017). *Evaluating the Effect of Voting Methods on Ensemble-Based Classification*

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*