Computer Vision Project Winter 2024/2025

Task Description

The final project for this course will require you to implement a CAPTCHA-Solver using the computer vision techniques discussed in the lecture. Your solver will take a CAPTCHA image as shown in Figure 1 and output the predicted string (along with any other intermediate outputs required for evaluation). The project is divided into the following four parts:

- Part 1: *Project Proposal (50 Points + 4 Bonus Points)*
- Part 2: *Primary Solution (90 Points)*
- Part 3: *Degradations (7 Points Bonus)*
- Part 4: *Challenging Degradations (7 Points Bonus)* or *Oriented Detection (7 Points Bonus)*

Part 1: Project Proposal

This initial phase will require you to write-up a min. 5-page project proposal (excluding references). The proposal is written ***individually***. The proposal essentially states the methodology and approach that you will use to create your primary solution for a Captchav solver (part 2). Given that there may be several ways to approach this problem, we leave the choice of method entirely unrestricted. You may follow existing techniques, however, **it is required that you implement your approach from scratch**, where you may only use generic machine learning and vision libraries (see last section). Your written proposal must contain the following sections:

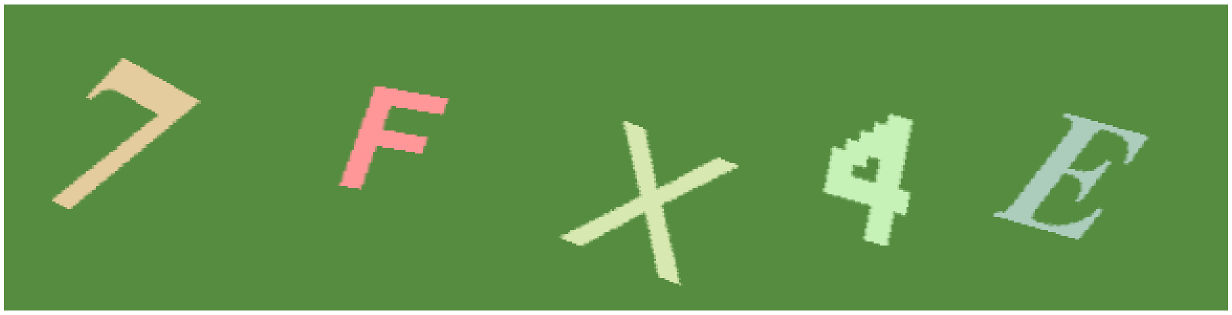


Figure 1: Sample captcha image from training set of part 1, with rotation, shear, multi-color backgrounds and multiple-fonts. This example has a total of 5 characters. The expected output in this case is: $7FX4E$.

- **Literature review of related methods (15 Points)**: Please provide a detailed overview of related approaches in the literature. For each approach, discuss how this approach relates to the given problem.
- **Method selection (6 Points)**: Please provide a detailed explanation of which approach you want to use and why you choose this method. Include a discussion of the advantages and disadvantages of selecting this method.
- **Data pipeline including augmentation strategies (9 Points)**: Please describe how your approach will process the training data and any augmentation strategies that you plan to apply.
- **Training including architecture and optimizer (6 Points)**: Please describe which architecture you will use and how you plan to train it, including the optimizer and hyperparameters.
- **Detailed description of the loss function(s) (6 Points)**: Please describe the loss function that you will apply to train your method.
- **Part 3 and part 4 strategies (4 Bonus Points)**: Here, you can earn bonus points by proposing strategies on how to implement solutions for parts 3 and 4.

Part 2: Primary Solution

In this part, you will be implementing your primary solution for an end-to-end Captcha solver. As shown in Figure 1, the goal is to predict the final Captcha string for an input image containing characters with random degradations. Here are the available characters that your Captcha solver will encounter:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

To facilitate the training of your method, we provide a dataset. It consists of a total of 100k images and annotations. We split the dataset into *train*, *val* and *test* splits. The training set is composed of 60k images and ground-truth Captcha pairs, while the validation set only contain 20k pairs. The test set contains 20k images without associated ground truth. Finally, you will have to run your method on this test set and submit the results to us, which we will then evaluate.
We use the Levenshtein Distance (please click on the link for details) to assess the quality of each prediction. Specificially, we use the Levenshtein Error Rate to measure over-all performance. Below is a precise definition of how the evaluation works:

Let D_{test} be the test dataset, where each sample consists of an image and its corresponding ground-truth CAPTCHA text. Denoting the dataset as:

$$D_{\text{test}} = \{(I_i, T_i) \mid i = 1, 2, \ldots, N\},$$

where:

- I_i is the i-th image in the test dataset,
- T_i is the ground-truth CAPTCHA text for the i-th image and
- N is the total number of samples in D_{test}.

For each image I_i you need to predict a CAPTCHA text $\hat{T}_i$. The **Levenshtein Distance** between the ground-truth text T_i and the predicted text $\hat{T}_i$ is defined as:

$$\text{Levenshtein Distance}(T_i, \hat{T}_i) = \min(\text{Insertions}, \text{Deletions}, \text{Substitutions})$$

where Insertions, Deletions, and Substitutions represent the minimal single-character edits required to transform T_i into $\hat{T}_i$. The **Levenshtein Error Rate (LER)** computes the average error rate across all samples in D_{test} and is given by:

$$LER(D_{\text{test}}, \hat{T}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{Levenshtein Distance}(T_i, \hat{T}_i)}{\text{Length}(T_i)}$$

where:

- $\hat{T} = \{\hat{T}_i \mid i = 1, 2, \ldots, N\}$ is the set of predicted CAPTCHA texts corresponding to the test dataset,
- $\text{Length}(T_i)$ is the number of characters in the ground-truth CAPTCHA T_i and
- $\text{Levenshtein Distance}(T_i, \hat{T}_i)$ is the distance between the ground truth CAPTCHA and the predicted CAPTCHA for the i-th sample.

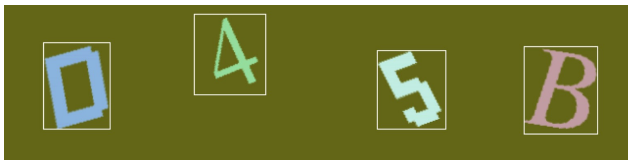


Figure 2: Upright Bounding Boxes visualized around the characters

For the training and validation sets we additionally provide bounding box coordinates for each character within the image in the dataset (X-center, Y-center, Width, Height), visualized in Figure 2. In the table below, we provide a summary of specifications for the dataset:

Splits	**Train**	**Val**	**Test**
Size	60k	20k	20k
Resolution	640x160		
Captcha Length	Variable		
Ground Truth Captcha String	✓	✓	×
Ground Truth Bounding Box	✓	✓	×

Table 1: Summary of dataset characteristics for Train, Val, and Test sets.

Part 3: *Added Degradations*

In this bonus section, your task will be to improve your Captcha solver to make it more robust to larger degradations. The updated test set will evaluate your solver for larger rotations, line distractors and image noise as shown in Figure 3. Please note that no training set is provided for these cases.

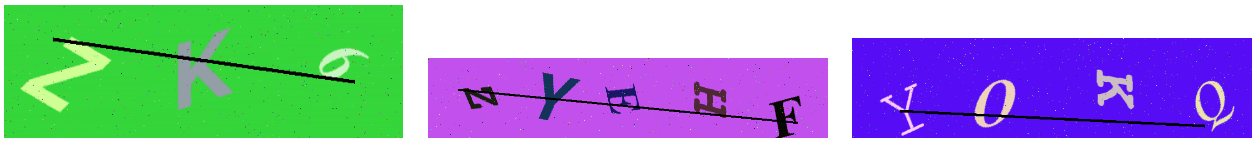


Figure 3: Samples from the testing set for part 3, with line distractors, larger rotations, noise, and complex backgrounds.

Part 4

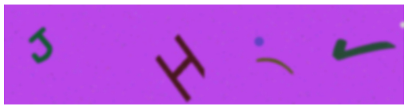
For this final bonus section, we provide two options and you may select one of the two.

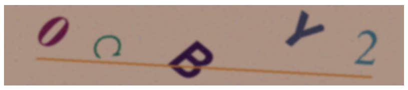
Option 1: Challenging degradations and distractors

This pathway extends the challenge of Phase 2 and adds further degradations in the test set. These include:

- Circular distractors
- Non-ASCII character distractors (a set of characters will be provided)
- Added challenging fonts
- Blur
- Overlap between characters

Below are examples with some of the above degradations. You may visually inspect the test set (when available) to get familiar with all different types.


(a) CAPTCHA ground truth: **'JH'**. The two other characters are distractors/background.


(b) CAPTCHA ground truth: **'OCBY2'**. Without any distractors.

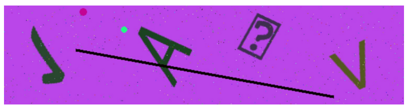

(c) CAPTCHA ground truth: **'JAV'**. The distractor symbol is **'?'**.

Figure 4: Examples of CAPTCHA images with distractor characters.

Option 2: Oriented Bounding Box Implementation

The alternative pathway you may choose to implement is detectecting oriented bounding boxes around the individual characters. We will use the *mAP@0.5:0.95* metric to evaluate your method for bounding box accuracy. *The dataset will not contain any further degradations than part 3.*

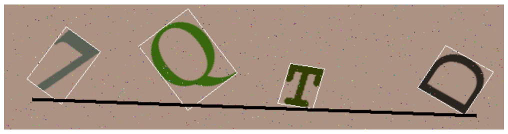


Figure 5: A sample from the test set of part 3 with oriented bounding boxes visualized.

Important Note

Group Work

The project proposals (part 1) are to be done individually. After this part the course will split into two teams and each team will implement a solution. Your implementation is independent of your own proposals and can use different ideas.

Implementation Note

The implementations must use only the toolboxes provided by the Python libraries below:

- pytorch
- numpy
- PILLOW
- torchvision
- matplotlib
- sklearn

If you would like to use a library outside of the listed ones, you need to obtain permission first.