Project Proposal

Computer Vision SS 25

by

Luca Burghard

Matriculation Number:	307
Course:	Class of Oct. 2024
Program:	Robotics and Artificial Intelligence
University:	University of Technology Nuremberg
Submission Date:	11.08.2025

Auxiliary means

This document was written in LaTeX. Formulations were generated with the support of language models such as OpenAI GPT-4o. The software *DeepL Write* was used to check spelling and grammar. *Zotero* was used to manage all cited sources.

All referenced papers were independently searched, read, and reviewed by the me, Luca Burghard. Full-text PDFs of each citation are stored in Zotero. Some LaTeX formulas, especially those involving formatting or mathematical expressions, were generated with assistance from ChatGPT due to limited prior experience with LaTeX equation syntax.

Contents

1 Literature Review

1.1 Preprocessing Techniques

Several studies applied multi-step preprocessing pipelines to simplify input data and improve model performance. Arivazhagan et al. and Khan et al. used grayscale conversion, Otsu's thresholding, and segmentation to localize characters [1, 2]. Noury and Rezaei added image downscaling and median filtering to reduce noise while preserving accuracy [3]. Tang et al. incorporated normalization, resizing, and advanced augmentation (e.g., rotation, flipping, color jitter) to improve generalization [4].

Across all works, data augmentation consistently improved robustness. These techniques are computationally efficient and widely supported. The proposed pipeline will therefore incorporate similar preprocessing, optionally Otsu's method and augmentation strategies.

1.2 Rule-Based and Classical Methods

Rule-based and segmentation-driven methods provide interpretable, low-resource alternatives. Mori and Malik's early shape-context matcher on Gimpy CAPTCHAs laid foundational work for spatial template matching [5]. Tigas developed a pixel-difference-based sliding window matcher using templates [6].

Arivazhagan et al. combined Otsu's thresholding, bounding-box segmentation, and SVM classification with handcrafted features like SIFT and KAZE, achieving 96.7% on clean datasets [1]. Khan et al. explored failure cases for these systems by introducing distortions based on Gestalt principles [2].

While outdated for complex CAPTCHAs, these approaches remain valuable as hybrid components or baselines, especially in clean, low-noise scenarios relevant to the project.

1.3 CNN-based Approaches

CNN-based models dominate recent CAPTCHA solving literature. Wan et al. proposed a CRNN with adaptive filtering and residual blocks, achieving 99% accuracy on a challenging dataset while maintaining CPU efficiency [7]. Arivazhagan et al. tested both a custom CNN and a modified LeNet-5, achieving 99.6% on simple CAPTCHAs and over 82% on more complex cases [1].

Noury and Rezaei's Deep-CAPTCHA used a three-block CNN architecture and handled severely distorted inputs, reaching 98.94% accuracy [3]. Tang et al. employed MobileNetV1, achieving 99.97% accuracy and 2491 FPS, making it ideal for real-time applications [4].

In the context of the Computer Vision Course at UTN SS 2025, we also revisited the fundamentals of CNN architectures, as covered in Learning Unit 2 of the course material [8]. Based on this material, the ResNet-18 will be selected as the backbone for both classification and regression in the Method selection. It strikes a balance between depth, efficiency (1.8 GFLOPs), and generalisation.
ResNet-18 consists of a single convolution stem followed by four residual stages with increasing channel counts (64 to 512), and it ends in a linear projection head. Its robustness and modularity make it ideal for our classification data — i.e., CAPTCHA-like glyphs with moderate distortion but consistent semantic structure. We will use it both as a classifier and as a bounding box regressor with task-specific output layers.

1.4 Transformer- and LLM-based Approaches

Transformer-based models have recently shown strong performance in CAPTCHA recognition. Wu et al. introduced MCA-Bench, using QwenVL-2.5-7B fine-tuned with LoRA for static visual CAPTCHA tasks, demonstrating that even modest vision-language models can reach strong baselines [9]. Deng et al. proposed OEDIPUS, combining LLMs and a DSL to solve reasoning-based CAPTCHAs via chain-of-thought, achieving 63.5% without training data [10]. While their tasks were more complex, simpler datasets like ours would likely yield higher scores.
Benchmarks show that general-purpose VLMs such as GPT-4o and Gemini Pro match or outperform traditional OCR tools like Tesseract and TrOCR on static visual text tasks. Among classical tools, only EasyOCR remains competitive.
Basson trained a Vision Transformer (ViT) on digit-only CAPTCHAs, reaching 98.7% accuracy without OCR post-processing [11].
Although large pretrained VLMs are out of scope due to project constraints, ViT models offer a viable transformer-based alternative that can be implemented from scratch using PyTorch.

2 Method selection

I propose a two–stage, two–network pipeline, both based on the well-established **ResNet-18** architecture:

2.1 ResNet18-BoundingBox (Will be called BoxNet18 from here on)

A ResNet-18-based convolutional network that receives the full CAPTCHA image (preprocessed as described in Data pipeline) and outputs a variable number of bounding boxes (x_1, y_1, x_2, y_2), one per glyph. The number of glyphs per image is not fixed and ranges between the minimum and maximum amount of glyph's per image. Detection is based on a confidence threshold or non-maximum suppression to determine how many glyphs are present.

2.2 ResNet18-GlyphClassifier (Will be called GlyphNet18 from here on)

A ResNet-18 model repurposed as a classifier that receives each predicted glyph crop (resized to 32×32) and outputs a 36-way probability distribution over the character classes $\{A \dots Z, 1 \dots 9\}$.

2.3 Why this method?

This approach separates spatial localization (BoxNet18) from classification (GlyphNet18), allowing for better modularity, interpretability, and targeted error handling. It is especially suited to the case where the number of glyphs varies, but the glyphs themselves are well-separated and moderately clean.

2.4 Advantages

- **Simplicity & debuggability** – each model handles a focused task (prediction errors can be traced to a specific stage).
- **Flexible length support** – the system accommodates different amount of glyphs per image without assuming a fixed number.
- **Data efficiency** – the classifier can be trained on cropped glyphs independently of full images, enabling better reuse and augmentation.
- **Low resource usage** – ResNet-18 is compact (1.8 GFLOPs)[8], enabling CPU inference on local machines.

2.5 Disadvantages

- **Error propagation** – poor bounding boxes lead to misclassification downstream.
- **No global context** – glyphs are classified in isolation, ignoring potential dependencies between neighboring characters.
- **Bounding box count estimation** – predicting how many glyphs to detect adds complexity compared to fixed-length models.
- **Not comparable to SOTA LLM and ViT** – while suitable for CAPTCHAs, this approach cannot match the accuracy or robustness of large transformer-based models on complex, heavily distorted data.

This method is a pragmatic balance between simplicity and flexibility, aligning with the constraints and goals of this project. Given the dataset properties (mildly distorted glyphs, bounding-box ground truth, clean background), this trade-off is preferable. As written in Literature Review, comparable segmentation-plus-CNN systems already achieve $> 97\%$ full-captcha accuracy on harder corpora, so there is some headroom.

3 Data pipeline

1. **Load & preprocess**

   All PNG images are loaded as grayscale using Pillow. Each image is resized to 128×32 to ensure consistent input dimensions for `BoxNet18`. The glyph crops for `GlyphNet18` are resized to 32×32.

   Additionally, a binary version of the image is computed using Otsu's thresholding (as in Literature Review) to experiment with background–foreground separation. This preprocessing path is kept optional and evaluated during ablation.

2. **Target preparation**

   Ground-truth bounding boxes $\mathbf{b}_i$ are normalized:

   $$\hat{\mathbf{b}}_i = \frac{\mathbf{b}_i}{(W, H, W, H)} \in [0,1]^4, \quad i = 1 \dots N$$

   where N is the number of glyphs per image. Bounding boxes and class labels are stored as tensors.

3. **Augmentation**

   To increase robustness against minor distortions, the following augmentations are applied for training (unless already covered by the dataset):

   - RandomRotation randomly between $\pm 20°$
   - Brightness/contrast randomly between $\pm 30\%$

   These transformations simulate typical CAPTCHA noise patterns, consistent with distortions reported in Literature Review.

If the dataset already includes sufficient variation in glyph rotations, brightness, and contrast, only minimal augmentation will be applied. Otherwise, the pipeline integrates affine transforms and brightness jitter to enhance robustness.

4 Training

4.1 Architectures

Both `BoxNet18` and `GlyphNet18` are based on the canonical **ResNet-18** architecture [12]. The standard implementation from `torchvision.models` will be used, pretrained weights are not used.

- **BoxNet18**:
  The final linear layer is replaced with a regression head: `nn.Linear(512,` $4 \cdot N$`)` where N is the maximum number of glyphs. A sigmoid activation is applied to output normalized box coordinates.
- **GlyphNet18**:
  The final layer is replaced with:

  ```
  nn.Linear(512, 36)
  ```

  which outputs unnormalized logits over the 36 character classes.

4.2 Schedule

The following training schedule uses a shared ResNet-18 backbone for both tasks (BoxNet18 and GlyphNet18), with separate output heads for regression and classification. The configuration reflects practical settings derived from prior work [13, 14], though exact hyperparameters may still require empirical validation.

- **Stage A – GlyphNet18 (20 epochs)**
  Train the classification head on cropped glyphs.
  Optimizer: SGD; learning rate = 0.001; momentum = 0.9 [14]; batch size = 256 glyph crops [13].
- **Stage B – BoxNet18 (20 epochs)**
  Freeze GlyphNet18 and train the bounding box regression head on full CAPTCHA images.
  Optimizer: SGD; learning rate = 0.001; momentum = 0.9; batch size = 64 full images.
- **Stage C – Joint fine-tune (5 epochs)**
  Unfreeze all layers and fine-tune end-to-end. Reduce learning rate by factor 10 to keep models stable.

Though both models share the ResNet-18 architecture, their input data differs: BoxNet18 operates on full images (128×32), while GlyphNet18processes small crops (32×32). The smaller crop size enables a larger batch size for GlyphNet18without exceeding memory limits.

5 Loss function

The training is performed in two distinct stages:

- **Stage 1 – BoxNet18:** train ResNet18-BoundingBox independently to predict bounding boxes using Intersection-over-Union (IoU) loss.
- **Stage 2 – GlyphNet18:** use predicted or ground-truth glyph crops to train ResNet18-GlyphClassifier using cross-entropy loss.

5.1 Loss definitions

$$\mathcal{L}_{\text{box}} = \frac{1}{N_{\text{box}}} \sum_{i=1}^{N_{\text{box}}} \Big(1 - \text{IoU}(\mathbf{b}_i, \hat{\mathbf{b}}_i)\Big) \tag{1}$$

$$\mathcal{L}_{\text{cls}} = \frac{1}{N_{\text{cls}}} \sum_{i=1}^{N_{\text{cls}}} \text{CE}(\mathbf{y}_i, \hat{\mathbf{p}}_i) \tag{2}$$

Where the Intersection over Union (IoU) is defined as:

$$\text{IoU} = \frac{\text{Area}_{\text{overlap}}}{\text{Area}_g + \text{Area}_d - \text{Area}_{\text{overlap}}} \tag{3}$$

Here:

- $\mathbf{b}_i$ and $\hat{\mathbf{b}}_i$ are the ground truth and predicted bounding boxes,
- Area_g is the area of the ground truth box,
- Area_d is the area of the predicted (detected) box,
- $\text{Area}_{\text{overlap}}$ is the intersection area between both boxes,
- CE is the cross-entropy loss used for classification.

5.2 Explanation

IoU loss is scale-invariant and directly optimizes the spatial overlap between predicted and ground-truth boxes making it preferable for this task. **Cross-entropy** is standard for multiclass classification and effective when training GlyphNet18 independently on glyph crops. Decoupling training allows each model to specialise in its own task without interference or gradient entanglement. This two-phase training strategy simplifies debugging, improves convergence stability, and aligns with the pipeline modularity discussed in Method selection.

6 Bonus: Strategies

The bonus tasks aim to assess robustness against severe distortions and distractors (Part 3) and either complex degradations or oriented detection (Part 4). The following strategies propose simple yet effective modifications to my existing architecture:

6.1 Part 3 – Robustness to Degradations

To handle line distractors, noise, and larger transformations:

- **Augmentation strategy extension**: Integrate stronger augmentations into training, such as:
  - Random Perspective warping
  - Synthetic strike-through and occlusion lines
  - Gaussian blur

  This forces the model to generalize to distortion patterns seen in the test set.

- **Hard-negative mining**: Misclassified glyphs from the validation set could be recycled as hard examples for re-training the classifier head (GlyphNet18).

6.2 Part 4 – Strategy: Oriented Bounding Box Detection

To support rotated glyph detection and meet the evaluation requirement for oriented bounding boxes, I will extend the BoxNet18 architecture to predict glyph rotation as well.
The modified approach includes:

- **Extend the regression head of BoxNet18** to predict a fifth parameter θ per glyph, representing the rotation angle (in radians or degrees).

- Replace the output layer with:

$$\texttt{nn.Linear(512, 5} \cdot \texttt{N)}$$

where N is the maximum number of glyphs (e.g., 10), and each glyph is now represented by $(x_1, y_1, x_2, y_2, \theta)$.

- Apply a rotation IoU.
- Since the dataset does not include explicit rotation angles, I will simulate rotated glyphs through affine augmentation (see Data pipeline). This enables BoxNet18 to generalize to rotated inputs without requiring ground-truth θ.

This design allows BoxNet18 to generate oriented boxes directly, avoiding overfitting to axis-aligned glyphs and improving accuracy on tilted characters. It integrates cleanly into the existing pipeline and can be trained using the same optimizer and scheduling settings as the original model.

In case the model fails to converge stably with angle regression, a fallback to angle classification or anchor-based rotated boxes will be considered.

References

[1] Arivazhagan S et al. “CAPTCHA RECOGNITION USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES”. In: *ICTACT Journal on Data Science and Machine Learning* 5.3 (June 1, 2024), pp. 641–649. ISSN: 30490197, 25839292. DOI: `10.21917/ijdsml.2024.0135`. URL: `https://ictactjournals.in/ArticleDetails.aspx?id=FN406W` (visited on 08/05/2025).

[2] Imran Moez Khan. “EVALUATING A SEGMENTATION-RESISTANT CAPTCHA INSPIRED BY THE HUMAN VISUAL SYSTEM MODEL”. In: *IIUM Engineering Journal* 12.2 (Oct. 18, 2011), pp. 145–154. ISSN: 2289-7860, 1511-788X. DOI: `10.31436/iiumej.v12i2.127`. URL: `https://journals.iium.edu.my/ejournal/index.php/iiumej/article/view/127` (visited on 08/05/2025).

[3] Zahra Noury and Mahdi Rezaei. *Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment.* June 24, 2020. DOI: `10.48550/arXiv.2006.08296`. arXiv: `2006.08296[cs]`. URL: `http://arxiv.org/abs/2006.08296` (visited on 08/05/2025).

[4] Shengyuan Tang. “Research on CAPTCHA recognition technology based on deep learning”. In: *Applied and Computational Engineering* 81.1 (Nov. 8, 2024), pp. 41–46. ISSN: 2755-2721, 2755-273X. DOI: `10.54254/2755-2721/81/20240967`. URL: `https://www.ewadirect.com/proceedings/ace/article/view/16621` (visited on 08/05/2025).

[5] G. Mori and J. Malik. “Recognizing objects in adversarial clutter: breaking a visual CAPTCHA”. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* CVPR 2003: Computer Vision and Pattern Recognition Conference. Madison, WI, USA: IEEE Comput. Soc, 2003, pp. I–134–I–141. ISBN: 978-0-7695-1900-5. DOI: `10.1109/CVPR.2003.1211347`. URL: `http://ieeexplore.ieee.org/document/1211347/` (visited on 08/05/2025).

[6] Panagiotis Tigas. *ptigas/simple-captcha-solver.* original-date: 2011-02-18T00:09:43Z. July 27, 2025. URL: `https://github.com/ptigas/simple-captcha-solver` (visited on 08/05/2025).

[7] Xing Wan, Juliana Johari, and Fazlina Ahmat Ruslan. “Adaptive CAPTCHA: A CRNN-Based Text CAPTCHA Solver with Adaptive Fusion Filter Networks”. In: *Applied Sciences* 14.12 (June 8, 2024), p. 5016. ISSN: 2076-3417. DOI: `10.3390/app14125016`. URL: `https://www.mdpi.com/2076-3417/14/12/5016` (visited on 08/05/2025).

[8] Justin Johnson. *498_FA2019_lecture08*. Sept. 30, 2019. URL: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture08.pdf (visited on 08/05/2025).

[9] Zonglin Wu et al. *MCA-Bench: A Multimodal Benchmark for Evaluating CAPTCHA Robustness Against VLM-based Attacks*. Aug. 2, 2025. DOI: 10.48550/arXiv.2506.05982. arXiv: 2506.05982[cs]. URL: http://arxiv.org/abs/2506.05982 (visited on 08/05/2025).

[10] Gelei Deng et al. *Oedipus: LLM-enchanced Reasoning CAPTCHA Solver*. May 13, 2024. DOI: 10.48550/arXiv.2405.07496. arXiv: 2405.07496[cs]. URL: http://arxiv.org/abs/2405.07496 (visited on 08/05/2025).

[11] OfekBasson. *OfekBasson/VisionTransformerCaptchaSolver*. original-date: 2023-12-03T21:14:19Z. Dec. 15, 2023. URL: https://github.com/OfekBasson (visited on 08/05/2025).

[12] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 10, 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385[cs]. URL: http://arxiv.org/abs/1512.03385 (visited on 08/05/2025).

[13] Sovit Ranjan Rath. *Training ResNet18 from Scratch using PyTorch*. DebuggerCafe. Sept. 26, 2022. URL: https://debuggercafe.com/training-resnet18-from-scratch-using-pytorch/ (visited on 08/05/2025).

[14] Suvadeep Hajra. *Weights & Biases*. W&B. Sept. 13, 2024. URL: httpss://wandb.ai/suvadeep/pytorch/reports/Finetuning-of-ResNet-18-on-CIFAR-10-Dataset--VmlldzoxMDE2NjQ1 (visited on 08/05/2025).