# Computer Vision Project Proposal

Kengo Kaneda

August 2025

## 1 Literature Review of Related Methods

### 1.1 DeepCAPTCHA

**Reference:** *Deep-CAPTCHA: A Deep Learning Based CAPTCHA Solver for Vulnerability Assessment*

**Overview**

DeepCAPTCHA is a method that uses a large number of automatically generated CAPTCHA images to train a convolutional neural network. It typically creates synthetic data with variations in fonts, noise, and shapes to mimic real CAPTCHAs.

**Technical Details**

- A standard CNN is trained end-to-end to predict the full text from the image.
- Each output character is treated as a separate classification target.
- Data diversity is controlled during synthetic image generation.

**Relation to the Problem**

Since the dataset is already provided for this project, synthetic data generation is not necessary. However, the main idea of training a CNN end-to-end on CAPTCHA images remains valuable. We can directly apply the DeepCAPTCHA model design to the given dataset.

### 1.2 Residual CNN + CTC Loss

**Reference:** *Segmentation-free CTC-based OCR for Text CAPTCHA Classification*

**Overview**

This method combines a Residual CNN with Connectionist Temporal Classification (CTC) loss to recognize text in CAPTCHA images without the need for character-level segmentation. It supports variable-length outputs and operates directly on full images.

**Technical Details**

- Based on the ResNet architecture, using residual blocks to enable deeper learning and better pattern extraction.
- Trained end-to-end on full CAPTCHA images.
- Uses CTC loss to align predicted sequences with target strings, enabling recognition without predefined character positions.
- Does not require segmentation or fixed-length assumptions.

**Relation to the Problem**

This method is well-suited for the project because it enables recognition of variable-length CAPTCHA strings without any segmentation or pre-processing. The combination of deep residual learning and CTC loss allows the model to learn complex spatial features while producing sequential outputs.

### 1.3 Attention-Based Image Captioning

**Reference:** *A Novel CAPTCHA Recognition System Based on Refined Visual Attention*

**Overview**

This approach models CAPTCHA recognition as a sequence generation problem, similar to image captioning. The model attends to different image regions and generates one character at a time.

**Technical Details**

- A CNN encoder extracts features from the image.
- An RNN decoder (e.g., LSTM) generates the output string character by character.
- An attention mechanism focuses on different image regions for each character.

**Relation to the Problem**

This method is particularly beneficial for complex CAPTCHAs or those with variable layouts. It requires no image segmentation and learns to focus on relevant areas of the image. Given the availability of labeled data, this model can be trained directly without additional data preparation.

## 2 Model Selection

### Selected Method: Residual CNN + CTC Loss

For this project, I chose to implement a Residual CNN model combined with Connectionist Temporal Classification (CTC) loss. This approach is segmentation-free and supports variable-length character sequences. The model takes a full CAPTCHA image as input and predicts the complete character string directly, without requiring pre-processing or manual segmentation.

### Why I Chose This Method

The Residual CNN + CTC approach is well suited to the CAPTCHA-Solver task for several reasons:

- **Supports Variable-Length Strings:** Unlike fixed-output models, this method can handle different character lengths, which matches the structure of the given dataset.
- **Segmentation-Free Learning:** The model does not require explicit character segmentation. It learns the alignment between image features and text automatically through CTC loss.
- **Stable Deep Network Training:** Residual connections in the CNN allow for stable and deep feature learning.
- **Widely Supported:** ResNet architectures and CTC implementations are well supported in major deep learning libraries such as PyTorch, making development and experimentation efficient.

### Advantages of This Method

- Segmentation-free and suitable for variable-length CAPTCHA strings.
- Deep residual architecture allows strong feature extraction.
- Compatible with end-to-end training pipelines using CTC loss.
- More flexible than DeepCAPTCHA, which assumes fixed-length strings and synthetic data.
- Simpler to implement and train than attention-based models that require sequence decoders.

### Disadvantages of This Method

- Higher computational cost due to the depth of the ResNet backbone.
- Requires CTC decoding which adds slight complexity to inference.
- Compared to attention-based models, it lacks explicit modeling of character dependencies.

## 3 Data Processing Pipeline

The model processes the training data as follows:

1. **Image Loading and Resizing:** All CAPTCHA images are loaded and resized to a consistent height, while maintaining or adjusting the width to accommodate different string lengths. Padding may be applied if necessary.
2. **Normalization:** Images are normalized (e.g., pixel values scaled to $[0, 1]$ or standardized using mean and standard deviation) to stabilize training.
3. **Label Encoding:** Text labels are converted into sequences of character indices using a predefined character set (e.g., 0–9, A–Z). These sequences are stored along with their lengths for use in CTC loss.
4. **Batch Collation:** A custom data loader collates images of the same height but variable width, and prepares the corresponding label lengths and input lengths required for CTC training.
5. **Output Format:** The model outputs a sequence of probability distributions over characters, one per time step. CTC loss uses the model's predictions and ground truth sequences for alignment during training.

### Augmentation Strategies

To improve generalization and robustness, the following augmentation strategies are applied during training:

#### Moderate Augmentation

This strategy introduces realistic distortions while preserving readability. It includes:

- Random rotation (up to $\pm 10°$)
- Color jitter (brightness and contrast variation)
- Gaussian blur (to simulate low-resolution or smudged text)
- Additive Gaussian noise (to mimic background interference)
- Normalization (standardizing input pixel values)

**No Augmentation (Baseline)**

This strategy performs no augmentation beyond basic preprocessing:

- Image resizing and normalization only
- No added noise, rotation, or transformations

This configuration serves as a baseline to evaluate the effectiveness of augmentations. It is also useful for debugging or verifying model functionality without external variability.

# 4 Training Including Architecture and Optimizer

## Architecture

I will use a ResNet18-based convolutional neural network combined with Connectionist Temporal Classification (CTC) loss. This architecture is relatively lightweight, easy to train, and widely supported in PyTorch. It provides a good balance between training efficiency and recognition performance.

## Optimizer and Training Plan

The model is trained using the Adam optimizer, which adapts the learning rate for each parameter and typically ensures fast and stable convergence.

The training configuration is as follows:

Table 1: Training Configuration

| **Parameter** | **Value / Description** |
|---|---|
| Optimizer | Adam optimizer with adaptive learning rates for each parameter. |
| Loss Function | Connectionist Temporal Classification (CTC) loss, enabling segmentation-free sequence learning. |
| Initial Learning Rate | 0.001 (decayed during training using scheduler). |
| Batch Size | 32, adjusted with mixed precision or gradient accumulation if needed. |
| Number of Epochs | 50–100, with early stopping if validation LER plateaus. |
| Weight Decay | $1 \times 10^{-5}$ to reduce overfitting. |
| Learning Rate Scheduler | Cosine annealing or ReduceLROnPlateau based on validation metrics. |
| Input Image Size | Height fixed |

# 5 Detailed Description of the Loss Function

The model is trained using the Connectionist Temporal Classification (CTC) loss, which is designed for sequence labeling tasks without requiring pre-segmented training data.

Let the input image be transformed by the CNN into a sequence of feature vectors:

$$\mathbf{x} = (x_1, x_2, \ldots, x_T), \quad x_t \in \mathbb{R}^d$$

where $T$ is the number of time steps after convolutional downsampling.

The network outputs, at each time step $t$, a probability distribution over the set of characters $\mathcal{C}$ plus a special blank symbol $\varnothing$:

$$\mathbf{y}_t = \text{softmax}(W x_t + b), \quad \mathbf{y}_t \in \mathbb{R}^{|\mathcal{C}|+1}.$$

Given a target label sequence $\mathbf{l} = (l_1, l_2, \ldots, l_L)$ with $l_i \in \mathcal{C}$, CTC defines the probability of $\mathbf{l}$ given $\mathbf{x}$ as the sum over all valid alignments $\pi \in \mathcal{B}^{-1}(\mathbf{l})$:

$$P(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} \prod_{t=1}^{T} y_{t,\pi_t}$$

where $\mathcal{B}$ is the collapsing function that removes repeated characters and blanks, and $y_{t,\pi_t}$ denotes the probability assigned at time $t$ to symbol $\pi_t$.

The CTC loss is the negative log-likelihood of the target sequence:

$$\mathcal{L}_{\text{CTC}} = -\log P(\mathbf{l} \mid \mathbf{x}).$$

### Properties and Advantages

- **Alignment-Free:** No explicit segmentation of characters is required; alignments are marginalized via dynamic programming.
- **Variable-Length Support:** The model can handle variable-length inputs and outputs naturally.
- **End-to-End Learning:** Directly optimizes sequence prediction from image features without hand-crafted alignment rules.

# 6 Bonus Task Strategies

The project description offers two optional bonus tasks: introducing additional degradations and handling oriented bounding boxes. My strategies are as follows:

### Part 3: Additional Degradations

- Introduce synthetic distortions beyond the original dataset, such as:
  - Elastic deformation
  - Random occlusions
  - Bézier curve distractors
  - Gaussian blur
- Incorporate these as optional transforms in the data pipeline during training.
- Tune probability and intensity to maintain readability and avoid underfitting.

- Apply progressive augmentation: start mild and increase severity in later epochs.
- Use test-time augmentation (TTA) with prediction voting to improve robustness under heavy degradation.

## Part 4: Oriented Bounding Boxes

- Extend the detector to predict rotated bounding boxes parameterised as $(x, y, w, h, \theta)$.
- Derive regression targets from oriented ground truth boxes.
- Include in the loss function:
  - Smooth L1 loss for position and size
  - Sine–cosine encoding for angle regression to handle periodicity
- Apply rotated non-maximum suppression (NMS) at inference.