# CS4243 Mini Project

Ng Chuan Xin, Yeoh Zhong Han Ervin, Goh Ler Xuan, Huang Yixin, Anton Tan Hong Zhi

## INTRODUCTION

This project aims to develop a CAPTCHA Recognition System that can accurately identify and convert CAPTCHA images into readable text. Leveraging techniques such as image preprocessing and deep learning, the system learns to overcome distortions, noise, and varying fonts commonly used to fool automated recognition.

Two Step Approach:

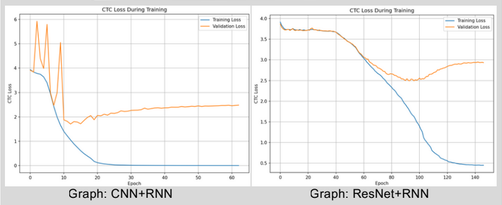
1. Data Preprocessing to localise alphanumeric characters within CAPTCHA images
2. Training various models to classify extracted regions before putting together the results for a final captcha output

Our final model then uses ensemble classification so that the framework can handle variable-length CAPTCHA effectively and achieve high accuracy in solving different kinds of CAPTCHAs

## NAIVE APPROACH – WHOLE CAPTCHA RECOGNITION

Our initial experiments explored three architectures for whole CAPTCHA recognition – a CNN+RNN(CTC) hybrid model, a ResNet+RNN(CTC) model, and TrOCR(CER), a transformer-based optical character recognition model. These architectures were designed to capture both the spatial features of CAPTCHA images and the sequential nature of text recognition.

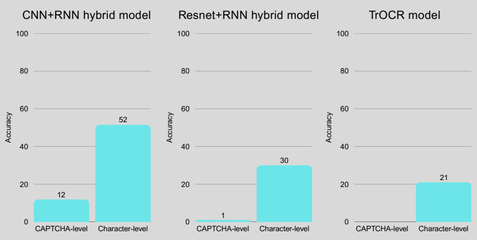
## RESULTS




Graph: CNN+RNN




Graph: ResNet+RNN




Early results showed poor overall CAPTCHA-level accuracy, primarily due to compounded errors across multiple character predictions.

To address this, we shifted our focus toward character-level accuracy, enabling the models to learn more effectively. We restructured our dataset by isolating individual characters from CAPTCHA images and retrained each model on these segmented datasets. This formed our 2 step approach aimed at improving recognition consistency and providing a clearer understanding of each model's strengths in character classification.

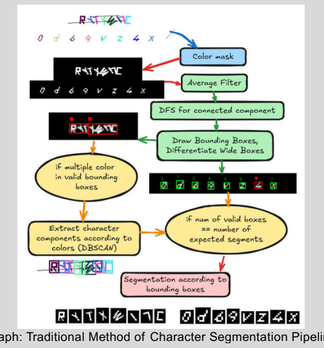
## DATA PREPROCESSING

1. **Cleanup of data**: Manually check individual datasets to fix mislabeled training data and corrupted pictures


Unidentifiable characters

Ambiguous characters (could be labeled as I or i)

2. **Isolation** of characters within CAPTCHA images:
   a. YOLO (Achieved 37.4% correct segmentations)
   b. Traditional Method of Image Segmentation (Achieved 91.7% correct segmentations)




Graph: Traditional Method of Character Segmentation Pipeline

## TRAINING BASELINE MODELS

With training based on our new character dataset, we utilised the character entropy loss as heuristic while training our models, and concluded that this character loss enabled more stable and effective training → resulting in higher accuracy. Consequently, subsequent training was based on this loss function to enhance model convergence.

| Baseline Model | Character-level Accuracy | CAPTCHA-level Accuracy |
|---|---|---|
| CNN | 0.8797 | 0.5586 |
| **ResNet50** | **0.9178** | **0.6654** |
| MLP | 0.8080 | 0.4141 |
| VGG16 | 0.8243 | 0.4514 |

For this dataset, ResNet had resulted in the best character recognition accuracy, likely due to its ability to learn deeper and more robust feature representations. We also experimented with other models such as Multi-Layer Perceptron (MLP) and Convolutional Neural Networks (CNN) to see whether more customizable and flexible parameter configurations would yield similar results while reducing training time and model size.

| MLP Layers | Input Size | Output Size | Dropout Rate |
|---|---|---|---|
| Layer 1 | 1024 | 512 | 0.5 |
| Layer 2 | 512 | 256 | 0.5 |
| Layer 3 | 256 | 128 | 0.3 |

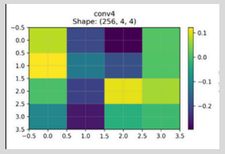
Graph: Multi-Layer Perception Model Architecture

| Layer Type | Input Channels | Output Channels | Kernel Size | Stride | Padding | Output Size (WxH) |
|---|---|---|---|---|---|---|
| Conv1 + MaxPool | 1 | 32 | 3 × 3 | 1 | 1 | 16 × 16 |
| Conv2 + MaxPool | 32 | 64 | 3 × 3 | 1 | 1 | 8 × 8 |
| Conv3 + MaxPool | 64 | 128 | 3 × 3 | 1 | 1 | 4 × 4 |
| Conv4 + MaxPool | 128 | 256 | 3 × 3 | 1 | 1 | 2 × 2 |
| Flatten | - | - | - | - | - | 1024 |
| FC1 | 1024 | 512 | - | - | - | - |
| FC2 | 512 | 128 | - | - | - | - |
| FC3 (Output) | 128 | 36 | - | - | - | - |

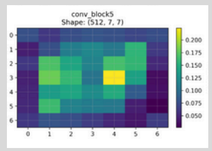
Graph: Convolutional Neural Network Model Architecture

## BASELINE MODELS RESULTS ANALYSIS

We concluded that ResNet had resulted in the best accuracy due to its ability to learn deeper and more robust features. For a deeper look at this, we can extract and visualise the learnt representations in the last layer of each model. (e.g. CNN and ResNet)




CNN

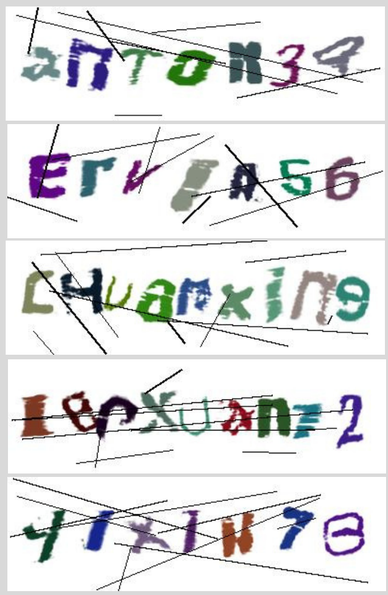



ResNet

## CAPTCHA GENERATION – GAN

Generative Adversarial Networks (GANs) consist of two competing neural networks – a generator that creates synthetic CAPTCHA images and a discriminator that distinguishes between real and generated samples. Through this adversarial training, the generator gradually learns to produce realistic and varied CAPTCHA images that mimic the patterns found in real datasets.

| | Method A: "End-to-End" (Failed) | Method B: "Per-Character" (Successful) |
|---|---|---|
| **Dataset** | Full CAPTCHA images (e.g., abc123.png). | Pre-segmented single characters (e.g., a.png). |
| **Task** | Model must learn shapes, layout, spacing, rotation, and noise all at once. | Model learns one simple task: distinct character shapes. |
| **Result** | **Training Failed.** The task was too complex. The model suffered from training collapse and produced unusable, noisy images. | **Training Succeeded.** The model learned to generate all 36 classes with high fidelity. |



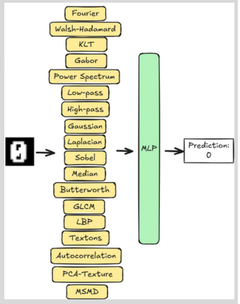
## HANDCRAFTED FEATURE EXTRACTION

Building upon our class material, we built our own feature maps incorporating techniques covered: a single-layer model and a multi-layer model.

Model 1 Architecture:

A single-layer feature extraction where features are extracted from the input images using various filters. These features are combined with raw pixel statistics to form a single feature vector for each image.

The extracted features are then fed into a Multi-Layer Perceptron (MLP) with three hidden layers for classification




Graph: Single Layer Feature Extraction

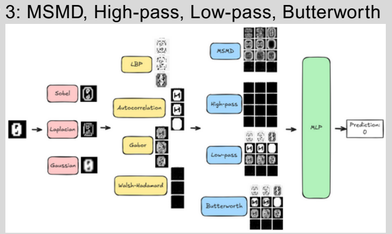
Model 2 Architecture:

A multi-layer feature extraction consisting of 3 layers.

Layer 1: Sobel, Laplacian, Gaussian

Layer 2: LBP, Autocorrelation, Gabor, Walsh-Hadamard

Layer 3: MSMD, High-pass, Low-pass, Butterworth




Graph: Multi-layer Feature Extraction

| Model | Character-level Accuracy | CAPTCHA-level Accuracy |
|---|---|---|
| Single-layer | 0.4283 | 0.0317 |
| Multi-layer | 0.8128 | 0.4241 |
| **ResNet50** | **0.9178** | **0.6654** |

**Single-layer Method:**

The single-layer method extracts features directly from raw images using only one set of filters. This limits its ability to capture complex or abstract patterns, resulting in shallow and less expressive features. Consequently, it struggles to distinguish similar characters and is more prone to overfitting and poor generalization.

**Multi-layer Method:**

The multi-layer method builds features hierarchically, similar to deep learning architectures, with each layer progressively refining and enriching the information.

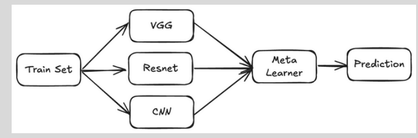
- Layer 1: Basic edges and structures
- Layer 2: Textures and orientations
- Layer 3: Multi-scale and frequency details

By capturing both local and global features, it produces a more discriminative and robust representation, leading to significantly better accuracy and generalization compared to the single-layer method.

**Comparison:**

The multi-layer method outperforms the single-layer approach by extracting features hierarchically, allowing it to capture more complex patterns and focus on important filters. However, ResNet surpasses both methods as it learns these hierarchical features automatically through deep residual learning, rather than relying on handcrafted filters. This enables ResNet to achieve higher accuracy and stronger generalization across diverse data.

## FINAL MODEL – ENSEMBLE




Graph: Ensemble Method

For our final model, we used ensemble classification, combining our multiple models to create a more accurate and robust one.

This approach trains individual CNN, ResNet50, and VGG16 models separately, then uses their validation set predictions as features to train a meta-learner.

**Simple Average Ensemble:** The most straightforward approach that assigns equal weights (1/3, 1/3, 1/3) to all three models and averages their softmax probability distributions before making the final prediction.

**Weighted Average Ensemble:** This method assigns weights to each model based on their individual validation accuracies, giving higher influence to better-performing models. The weights are normalized.

**Learned Weights Ensemble:** Uses numerical optimization (scipy.minimize) to find the optimal combination weights that maximize ensemble accuracy on the validation set. The optimization is constrained so weights sum to 1 and remain non-negative, essentially learning the best linear combination of model predictions.

## RESULTS ANALYSIS

| Meta Learner | Character-level Accuracy | CAPTCHA-level Accuracy |
|---|---|---|
| Logistic Regression | 0.9096 | 0.6437 |
| Simple Average | 0.9196 | 0.6709 |
| Weighted Average | 0.9195 | 0.6693 |
| **Learned Weight** | **0.9198** | **0.6715** |

Our final model utilised the Learned Weights Ensemble method to achieve the **highest accuracy of 0.9198 (Character-level) and 0.6715 (CAPTCHA-level)**, outperforming ResNet. This shows the principle that for this task of CAPTCHA recognition, a collective of models does indeed produce better predictions than any single model by compensating for individual errors and biases.

## REFERENCES


- TrOCR: Train OCR model on captcha dataset - https://www.kaggle.com/code/chenkuanchung/train-ocr-model-on-captcha-dataset#Inference
- Handcrafted Feature Extraction: Gonzalez and Woods, Digital Image Processing, 4th ed., Pearson, 2018. (Chapter 3, 4, 7)
- GAN Model: Kwon, Hyun & KIM, Yongchul & YOON, Hyunsoo & Choi, Daeseon. (2018). CAPTCHA Image Generation Systems Using Generative Adversarial Networks. IEICE Transactions on Information and Systems. E101.D. 543-546. 10.1587/transinf.2017EDL8175.


## TEAM CONTRIBUTIONS

- Ng Chuan Xin: Data preprocessing, handcrafted feature extraction
- Yeoh Zhong Han Ervin: Naive approach, baseline model training, ensemble
- Goh Ler Xuan: Baseline model training, poster creation
- Huang Yixin: GAN captcha generation
- Anton Tan Hong Zhi: Data preprocessing, baseline model training

Link to GitHub Repo: https://github.com/Ervinoreo/CS4243-miniproject