CAPTCHA Recognition Using Deep Learning

Karthik Venkat Malavathula

1 Tasks Done

I've successfully completed all the tasks. I did not leave any part. However, there are always possibilities to try to improve my results, by trying different architectures and ideas.

2 Introduction

This Precog task was an incredible learning experience! I got to dive into the fascinating world of CAPTCHA recognition. It was like learning a new language. Not only did I learn how to read and analyze research papers, but I also had the opportunity to turn theory into practice by applying what I learned to real-world problems.

At the start, I found myself learning up information from a variety of sources—research papers, tutorials, and youtube videos to get a deeper understanding of the problem. Once I had a grasp of the concepts, I took the plunge and started coding. There were a lot of challenges along the way, but it was all part of the process. By learning from both successes and failures, I was able to change approaches.

I'll begin by sharing some key insights from the research papers that guided my approach. After that, I'll talk aobut the step-by-step process I followed to build the solution, from data generation and preprocessing to model training and evaluation. Along the way, I'll highlight the justification behind each decision I made, explaining why certain methods were chosen and what I learned from each stage.

In the end, I'll present my results, and of course, provide my reflections on what worked well, what didn't, and the lessons I've learned. It's been a nice opportunity to apply theory in a meaningful way and see the results. With my learning, I wish to apply all this knowledge in future projects!

3 Summary of Research Papers

- **Paper 1: A Review on Text-Based CAPTCHA Breaking using Deep Learning [1]**
  - CAPTCHA recognition methods: CV, Random Guessing, Human Solver Relay.
  - CV-based methods: Image Operators, ML, DL.
  - CNNs outperform SVMs in CAPTCHA recognition.
  - Recognition Techniques: CNN, RNN, Attention Mechanisms, Capsule Networks, DeepCaptcha.
  - Object detection: YOLO, SSD, Faster R-CNN.
  - GANs used for synthetic data generation.
  - Evaluation Metrics: Single Character Recognition Accuracy, Model Parameter Quantity, Attack Success Rate.

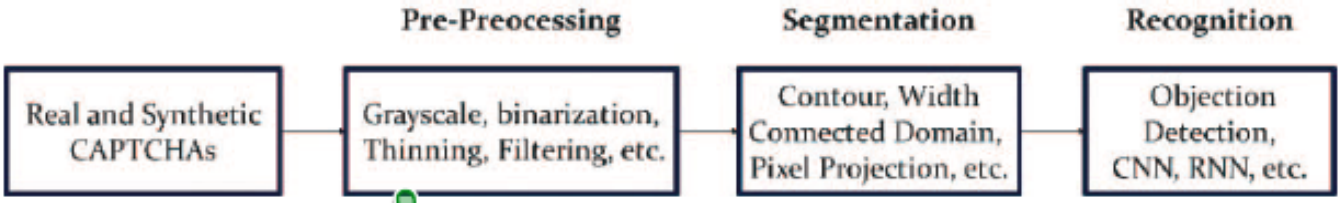

Figure 1: CAPTCHA Graphs of the Model

- **Paper 2: Adaptive CAPTCHA - A CRNN-Based Text CAPTCHA Solver**
  - CNN+RNN-based model with Adaptive Fusion Filtering Networks (AFFN).
  - RNNs and LSTMs handle CAPTCHAs with connecting characters.
  - CRNN with CTC loss function enhances recognition ability.
  - CNN+RNN architecture provides high accuracy with low complexity.

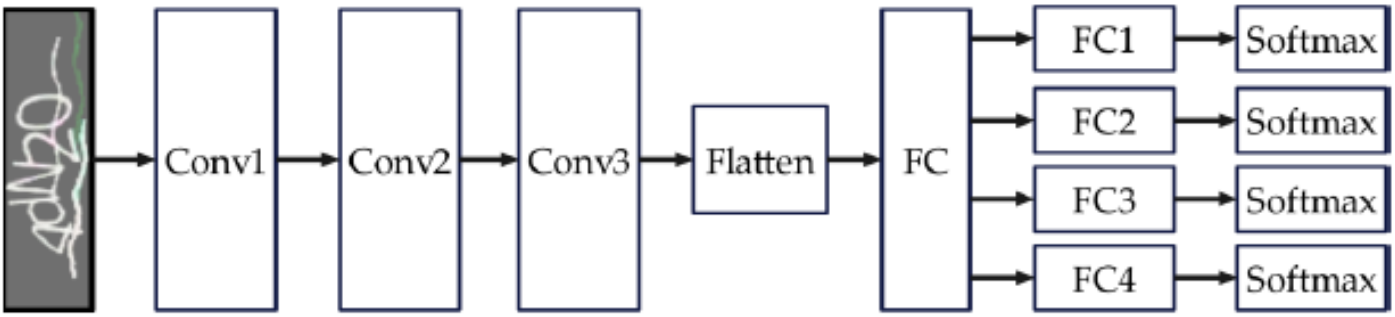

Figure 2: CAPTCHA Graphs of the Model

  - Dataset: P-CAPTCHA from Python ImageCaptcha library.
  - Loss Functions: Focal, MSE, BCE.

- **Paper 3: Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy**
  - CAPTCHA recognition methods: segmentation-based and segmentation-free.
  - Segmentation-free models exhibit higher accuracy and efficiency.

- **Paper 4: Image-Based CAPTCHA Recognition Using Deep Learning Models**
  - Batch size: 32, Learning rate: 0.001, Adam optimizer.
  - 50 epochs, early stopping.

- **Paper 5: Unveiling CAPTCHAs - Advanced CNN Techniques for High-Accuracy Recognition**
  - CNNs effectively recognize text-based CAPTCHAs.
  - Encoder: CNN, Decoder: RNN.
  - ResNet+LSTM achieves high accuracy.

4 Plan of Action

1. **Create synthetic data manually.**
   - Initially, I attempted to manually create or photograph CAPTCHA images. This method was time-consuming and inconsistent due to lighting and angle variations, making it unfeasible for large datasets.
2. **Generate synthetic data using Python ImageCaptcha library.**
   - I discovered the Python ImageCaptcha library, which allowed me to easily generate a wide variety of CAPTCHA images with different distortions, noise, and text. It was flexible, fast, and provided a large dataset.
3. **Pre-processing: Scaling, Greyscale conversion, Normalization.**
   - Preprocessing was key. I scaled images to a fixed size, converted them to greyscale to simplify the model, and normalized pixel values to improve convergence speed and accuracy.
4. **Train/Validation/Test split.**
   - I split the data into 80% for training, 10% for validation, and 10% for testing. This helped avoid overfitting and ensured the model would generalize to unseen data.
5. **Model definitions: CNN, RNN, CNN+RNN.**
   - I used CNNs for feature extraction and RNNs (LSTMs) for handling sequences. Combining both models allowed the system to efficiently recognize text in CAPTCHAs with varying distortions.
6. **Training with 100 samples.**
   - I trained the model with 100 labeled samples, which proved valuable for testing the model's ability to generalize, even though the accuracy was initially low. This small dataset helped identify areas for improvement.
7. **Classification.**
   - After training, the model was tested on new CAPTCHA images. The classification accuracy was reasonable, but edge cases revealed areas for further tuning and data augmentation.

8. **Data generation.**
    - I augmented the dataset with variations in fonts, distortions, and noise to improve generalization. The more diverse the data, the better the model's performance on unseen images.
9. **Loss Function Tuning.**
    - I tested several loss functions (e.g., Cross-Entropy, CTC Loss) to find the most effective one for my model. I found that Cross-Entropy loss, often used for sequence recognition, worked best for handling the CAPTCHA text.
10. **Bonus task.**
    - I explored creating mirror-flipped CAPTCHAs, reversed text, and added various types of noise. This task helped make the model more resilient to different transformations.
11. **Evaluation Metrics: Accuracy, Params, Precision, FPS, Inference Speed, Failure Case Analysis.**
    - Accuracy was the main metric, but I also measured precision, inference speed, and failure cases. These helped refine the model and ensure it was efficient and robust.
12. **Model Optimization and Testing.**
    - To optimize the model, I tested various architectures, dropout rates, and regularization techniques to avoid overfitting and improve generalization. It was crucial to balance model complexity with efficiency.
13. **Error Analysis.**
    - I carefully analyzed failure cases where the model misclassified CAPTCHAs, paying attention to distortions or noisy images. This provided valuable insights for further data generation and model adjustments.
14. **Hyperparameter Tuning.**
    - I experimented with learning rates, batch sizes, and optimizer choices (Adam, SGD) to fine-tune the model's performance. Small changes here had significant effects on convergence and accuracy.
15. **Model Optimization and Testing.**
    - To optimize the model, I tested various architectures, dropout rates, and regularization techniques to avoid overfitting and improve generalization. It was crucial to balance model complexity with efficiency.

5 Experimentation

5.1 Case 1: Parameter Count - 16921700

Train/Val/Test	Model	Loss Function	Epochs	Description	Accuracy (Train/Val/Test)	Observations
70/15/15	CNN	Cross Entropy	10	4 Conv → ReLU → Max Pool, 2 FC	8/0/0	Low accuracy, further adjustments needed
70/15/15	CNN	Cross Entropy	30	4 Conv → ReLU → Max Pool, 2 FC	61/0/6	Model underperforming, possibly needs more training or tweaks
70/15/15	CNN	Cross Entropy	30	4 Conv → ReLU → Max Pool, 2 FC	60/0/0	Batch Normalization applied, low accuracy observed

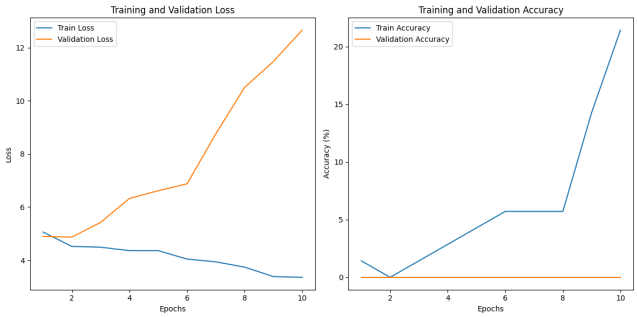

Figure 3: Graphs of the Model

```
Epoch 1/10 - Train Loss: 5.0594, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 1/10 - Val Loss: 4.8958, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 2/10 - Train Loss: 4.5194, Train Accuracy: 0.00% - Train Correct Predictions: 0/70
Epoch 2/10 - Val Loss: 4.8681, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 3/10 - Train Loss: 4.4899, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 3/10 - Val Loss: 5.4247, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 4/10 - Train Loss: 4.3600, Train Accuracy: 2.86% - Train Correct Predictions: 2/70
Epoch 4/10 - Val Loss: 6.3233, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 5/10 - Train Loss: 4.3592, Train Accuracy: 4.29% - Train Correct Predictions: 3/70
Epoch 5/10 - Val Loss: 6.6152, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 6/10 - Train Loss: 4.0436, Train Accuracy: 5.71% - Train Correct Predictions: 4/70
Epoch 6/10 - Val Loss: 6.8738, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 7/10 - Train Loss: 3.9385, Train Accuracy: 5.71% - Train Correct Predictions: 4/70
Epoch 7/10 - Val Loss: 8.7535, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 8/10 - Train Loss: 3.7409, Train Accuracy: 5.71% - Train Correct Predictions: 4/70
Epoch 8/10 - Val Loss: 10.5018, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 9/10 - Train Loss: 3.3836, Train Accuracy: 14.29% - Train Correct Predictions: 10/70
Epoch 9/10 - Val Loss: 11.4770, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 10/10 - Train Loss: 3.3563, Train Accuracy: 21.43% - Train Correct Predictions: 15/70
Epoch 10/10 - Val Loss: 12.6560, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
```

Figure 4: Numerical Results of the Model

5.1.1 Inferences

- **Low Accuracy**: Despite increasing epochs, accuracy remained low, suggesting insufficient learning or model limitations.
- **Batch Normalization Impact**: Batch normalization didn't improve results, possibly indicating its limited effect with this architecture.
- **Training Duration**: Even with 30 epochs, the model showed minimal improvement, hinting at the need for longer training or adjustments.

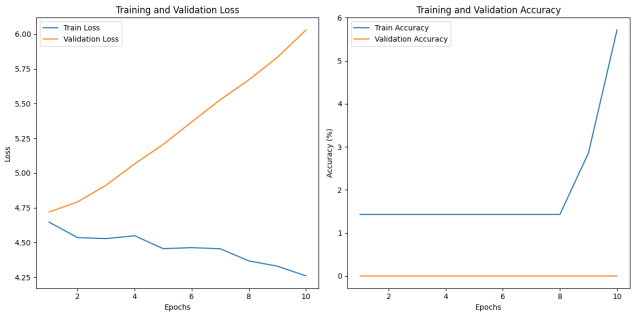

Figure 5: Graphs of the Model

```
Epoch 1/10 - Train Loss: 4.6465, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 1/10 - Val Loss: 4.7191, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 2/10 - Train Loss: 4.5351, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 2/10 - Val Loss: 4.7922, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 3/10 - Train Loss: 4.5278, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 3/10 - Val Loss: 4.9129, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 4/10 - Train Loss: 4.5484, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 4/10 - Val Loss: 5.0662, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 5/10 - Train Loss: 4.4552, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 5/10 - Val Loss: 5.2064, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 6/10 - Train Loss: 4.4627, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 6/10 - Val Loss: 5.3685, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 7/10 - Train Loss: 4.4546, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 7/10 - Val Loss: 5.5281, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 8/10 - Train Loss: 4.3672, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 8/10 - Val Loss: 5.6714, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 9/10 - Train Loss: 4.3290, Train Accuracy: 2.86% - Train Correct Predictions: 2/70
Epoch 9/10 - Val Loss: 5.8332, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 10/10 - Train Loss: 4.2593, Train Accuracy: 5.71% - Train Correct Predictions: 4/70
Epoch 10/10 - Val Loss: 6.0297, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
```

Figure 6: Numerical Results of the Model

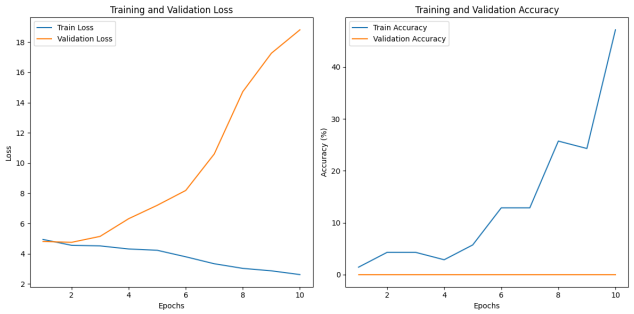

Figure 7: Graphs of the Model

```
Epoch 1/10 - Train Loss: 4.9457, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 1/10 - Val Loss: 4.8124, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 2/10 - Train Loss: 4.5581, Train Accuracy: 4.29% - Train Correct Predictions: 3/70
Epoch 2/10 - Val Loss: 4.7545, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 3/10 - Train Loss: 4.5194, Train Accuracy: 4.29% - Train Correct Predictions: 3/70
Epoch 3/10 - Val Loss: 5.1487, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 4/10 - Train Loss: 4.3145, Train Accuracy: 2.86% - Train Correct Predictions: 2/70
Epoch 4/10 - Val Loss: 6.3173, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 5/10 - Train Loss: 4.2293, Train Accuracy: 5.71% - Train Correct Predictions: 4/70
Epoch 5/10 - Val Loss: 7.2033, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 6/10 - Train Loss: 3.7986, Train Accuracy: 12.86% - Train Correct Predictions: 9/70
Epoch 6/10 - Val Loss: 8.1860, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 7/10 - Train Loss: 3.3382, Train Accuracy: 12.86% - Train Correct Predictions: 9/70
Epoch 7/10 - Val Loss: 10.6006, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 8/10 - Train Loss: 3.0318, Train Accuracy: 25.71% - Train Correct Predictions: 18/70
Epoch 8/10 - Val Loss: 14.7249, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 9/10 - Train Loss: 2.8702, Train Accuracy: 24.29% - Train Correct Predictions: 17/70
Epoch 9/10 - Val Loss: 17.2607, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 10/10 - Train Loss: 2.6208, Train Accuracy: 47.14% - Train Correct Predictions: 33/70
Epoch 10/10 - Val Loss: 18.8110, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
```

Figure 8: Numerical Results of the Model

5.1.2 Justification and Next Steps

- **CNN Architecture**: A basic CNN was chosen for feature extraction, as it is widely effective for image tasks. This structure served as a baseline.
- **Cross Entropy Loss**: Standard choice for classification tasks, and I felt it was useful for this based on reserach papers
- **Epochs**: 10-30 epochs were used to check for convergence, though the model didn't reach optimal performance.

Next Steps: - **Explore Advanced Architectures**: Investigate CNN+RNN or deeper CNNs to improve performance on complex CAPTCHA texts.
- **Tune Hyperparameters**: Test different learning rates, and regularization techniques like dropout.
- **Expand Training Data**: Use more varied CAPTCHA data to help the model generalize better to unseen data.

5.1.3 Failure Case Analysis

- **Edge Cases**: The model struggled with noisy or distorted CAPTCHAs, highlighting its inability to generalize to diverse CAPTCHA styles.

5.2 Case 2: Parameter Count - 8828260

Train/Val/Test	Model	Loss Function	Epochs	Description	Accuracy (Train/Val/Test)	Observations
70/15/15	CNN	Cross Entropy	10	3 Conv → ReLU → Max Pool, 2 FC	21.43/0/0	Loss is high, LR too high
70/15/15	CNN	Cross Entropy	10	3 Conv → ReLU → Max Pool, 2 FC	5.71/0/0	Adjusted LR – 0.0001 → Poor performance
70/15/15	CNN	Cross Entropy	30	3 Conv → ReLU → Max Pool, 2 FC	77.14/0/0	Increased epochs → Overfitting to training data
70/15/15	CNN	Cross Entropy	30	3 Conv → ReLU → Max Pool, 2 FC	5.71/0/0	Model is simple, or Dropout = 0.3, Dropout = 0.4, Dropout = 0.5
70/15/15	CNN	Cross Entropy	30	3 Conv → ReLU → Max Pool, 2 FC	90/0/0	Batch Normalization

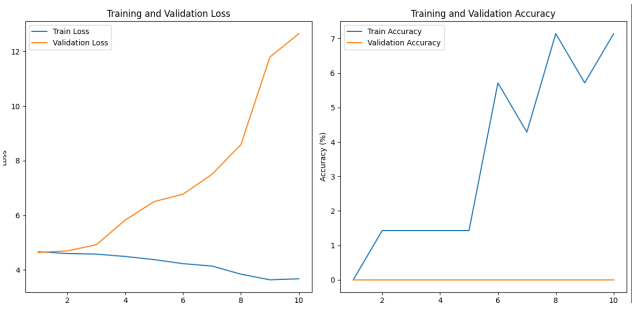

Figure 9: Graphs of the Model

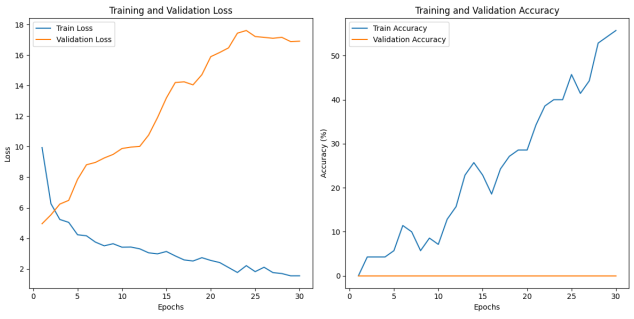

Figure 10: Graphs of the Model

5.2.1 Inferences

- **Edge Cases**: The model struggled with noisy and distorted CAPTCHAs, showing poor generalization to diverse styles.
- **Underfitting**: Despite training for 30 epochs, the model failed to capture key features, indicating underfitting.
- **Noisy Data Challenge**: The model's failure with noisy images suggests it needs better handling of variations.

5.2.2 Justification and Next Steps

- **Epochs**: Started with 30 epochs to give the model enough time to learn, though results showed the need for further adjustments.

Next Steps:

- **Use CNN+RNN Models**: Explore combining CNNs with RNNs for better sequential learning, especially for distorted or connected characters.
- **Increase CNN Layers**: Experiment with increasing the number of CNN layers to extract deeper feature representations and improve model performance.
- **Hyperparameter Tuning**: Adjust learning rates, batch sizes, and experiment with different optimizers to improve model training and convergence.

5.2.3 Improvement Strategies

- **Data Augmentation**: Introduce more realistic distortions and noise to better simulate real-world CAPTCHAs.
- **Transfer Learning**: Use pre-trained models like ResNet to leverage advanced feature extraction and improve performance.
- **Regularization**: Add techniques like dropout or early stopping to prevent overfitting and speed up convergence.

5.3 Case 3: Parameter Count - 5865416

Train/Val/Test	Model	Loss Function	Epochs	Description	Accuracy (Train/Val/Test)	Observations
70/15/15	CNN	Cross Entropy	10	5 Conv → ReLU → Max Pool, 2 FC	1.43/0/0	
70/15/15	CNN	Cross Entropy	30	5 Conv → ReLU → Max Pool, 2 FC	31.4/0/0	Increased epochs
70/15/15	CNN	Cross Entropy	30	5 Conv → ReLU → Max Pool, 2 FC	55.71/0/0	Batch Normalization applied

```
Epoch 1/10 - Train Loss: 4.6382, Train Accuracy: 0.00% - Train Correct Predictions: 0/70
Epoch 1/10 - Val Loss: 4.6302, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 2/10 - Train Loss: 4.5940, Train Accuracy: 0.00% - Train Correct Predictions: 0/70
Epoch 2/10 - Val Loss: 4.7037, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 3/10 - Train Loss: 4.5491, Train Accuracy: 0.00% - Train Correct Predictions: 0/70
Epoch 3/10 - Val Loss: 5.6379, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 4/10 - Train Loss: 4.6546, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 4/10 - Val Loss: 5.4745, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 5/10 - Train Loss: 4.4458, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 5/10 - Val Loss: 5.0258, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 6/10 - Train Loss: 4.4783, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 6/10 - Val Loss: 5.1885, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 7/10 - Train Loss: 4.4341, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 7/10 - Val Loss: 6.0322, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 8/10 - Train Loss: 4.3761, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 8/10 - Val Loss: 7.8165, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 9/10 - Train Loss: 4.3038, Train Accuracy: 0.00% - Train Correct Predictions: 0/70
Epoch 9/10 - Val Loss: 7.4969, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
Epoch 10/10 - Train Loss: 4.3529, Train Accuracy: 1.43% - Train Correct Predictions: 1/70
Epoch 10/10 - Val Loss: 7.1231, Val Accuracy: 0.00% - Val Correct Predictions: 0/15
```

Figure 11: Numerical Results of the Model

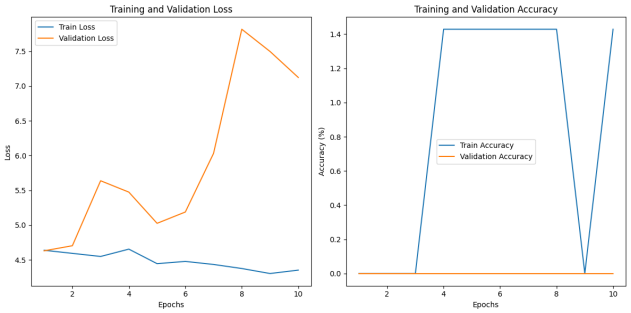

Figure 12: Graphs of the Model

Figure 13: Graphs of the Model

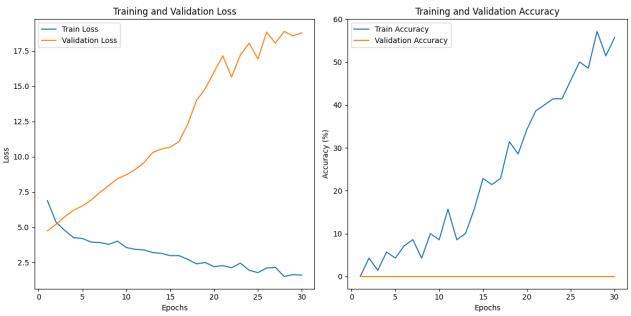

Figure 14: Graphs of the Model

5.3.1 Inferences

- **Low Initial Accuracy**: The model's accuracy started very low, even after 10 epochs, indicating that the architecture wasn't fully suited to the task.
- **Slow Improvement**: Accuracy remained stuck at low levels even after 30 epochs, suggesting the model wasn't learning effectively.
- **Batch Normalization Effect**: Applying batch normalization resulted in a slight improvement, but still, the model showed underwhelming performance.

5.3.2 Justification and Next Steps

- **Epochs**: Trained for 30 epochs based on typical training times, but it was clear more epochs or a different approach was needed for better results.

5.3.3 Improvement Strategies

- **Optimize Architecture**: Test with deeper CNNs or add LSTM layers to capture sequential dependencies in CAPTCHA text.
- **Data Augmentation**: Increase data diversity by applying transformations (noise, rotation, distortion) to enhance generalization.
- **Learning Rate Adjustment**: Experiment with different learning rates and optimizers to speed up convergence and improve performance.

5.4 Case 4

Train/Val/Test	Model	Loss Function	Epochs	Description	Accuracy (Train/Val/Test)	Observations
140/15/15	CNN	Cross Entropy	10	5 Conv → ReLU → Max Pool, 2 FC	0.71/0/0	
140/15/15	CNN	Cross Entropy	30	5 Conv → ReLU → Max Pool, 2 FC	45.71/0/0	Increased epochs
140/15/15	CNN	Cross Entropy	30	5 Conv → ReLU → Max Pool, 2 FC	54.29/0/0	Batch Normalization applied

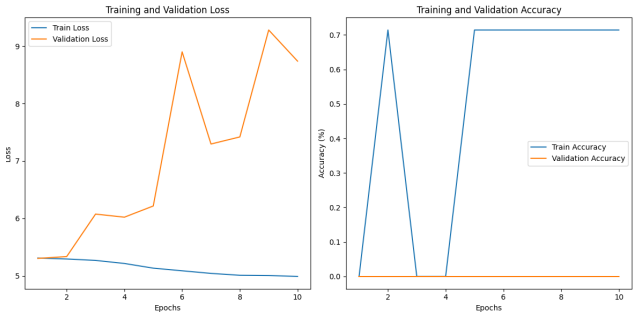

Figure 15: Graphs of the Model

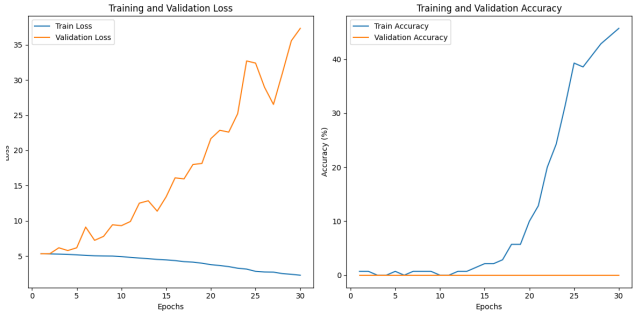

Figure 16: Graphs of the Model

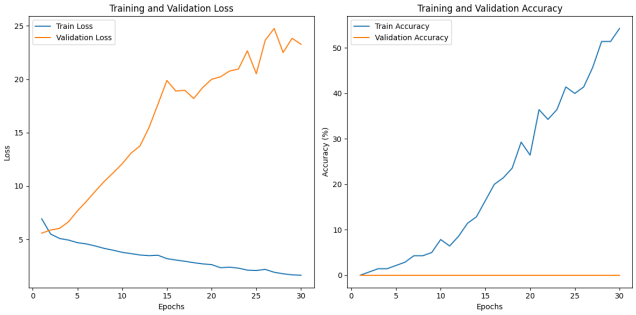

Figure 17: Graphs of the Model

Inferences:

- **Model is not Generalizing:** The validation and test accuracy remain at 0 across all experiments, indicating severe overfitting or a failure to learn meaningful patterns.
- **Training Accuracy Improvement:** Increasing the number of epochs (from 10 to 30) helped the model improve training accuracy, but this did not translate to validation or test accuracy gains.

- **Batch Normalization Impact:** Applying batch normalization increased training accuracy further (from 45.71% to 54.29%), but validation and test accuracy remained at 0, suggesting the issue isn't just training convergence.

- **Potential Issues:**
  - **Data Leakage or Label Mismatch:** If validation/test accuracy is always 0, check if labels are misassigned or if there's an issue with how data is split.
  - **Class Imbalance:** If the dataset is imbalanced, the model may be biased toward dominant classes.
  - **Learning Rate or Optimization Issue:** The model might be overfitting too early or struggling with vanishing gradients.

Next Steps:

- **Debug Data Processing:** Ensure that the dataset splits and labels are correctly assigned.
- **Check for Class Imbalance:** If present, try weighted loss functions or oversampling.
- **Try Different Optimizers & Learning Rates:** Experiment with Adam, RMSprop, or a different learning rate to stabilize learning.
- **Try Data Augmentation:** Introduce transformations like rotation, flipping, and noise to improve generalization.

5.5 Case 5: Data Augmentation

Train/Val/Test	Model	Loss Function	Epochs	Description	Accuracy (Train/Val/Test)	Observations
140/15/15	CNN	Cross Entropy	10	3 Conv → ReLU → Max Pool, 2 FC	61.43/26.67/30	
140/15/15	CNN	Cross Entropy	30	3 Conv → ReLU → Max Pool, 2 FC	95/50/60	Increased epochs
140/15/15	CNN	Cross Entropy	30	3 Conv → ReLU → Max Pool, 2 FC	98.57/46.67/63.33	Batch Normalization applied
140/15/15	CNN	Cross Entropy	Early Stop	3 Conv → ReLU → Max Pool, 2 FC	63.57/23.67/20	Batch Normalization applied

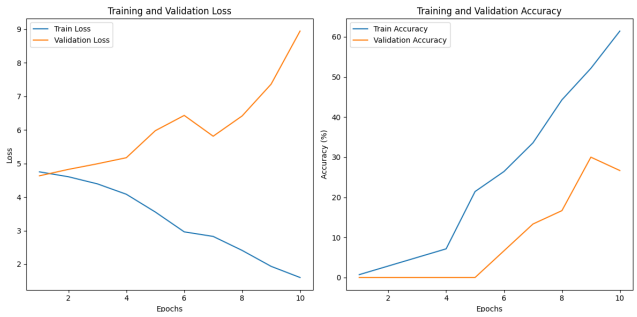

Figure 18: Graphs of the Model

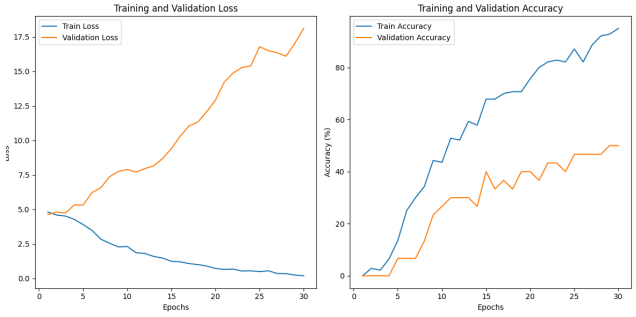

Figure 19: Graphs of the Model

5.6 Case 6: Data Augmentation

4 Conv → ReLU → Max Pool + 2 FC

Loss Function	Epochs	Accuracy (Train/Val/Test)	Parameter Count	Inference Speed (FPS)	Observation / Steps Taken
Cross Entropy	30	97.62/55.56/77.78	8828260	0.0116	Increased epochs
Cross Entropy	30	100/77.78/82.22	16922148	0.0100	Batch Normalization applied

5.7 Case 7: Data Augmentation

5 Conv → ReLU → Max Pool + 2 FC

Loss Function	Epochs	Accuracy (Train/Val/Test)	Parameter Count	Inference Speed (FPS)	Observation / Steps Taken
Cross Entropy	30	71.90/41.00/51.11	5814116	0.0124	Increased epochs
Cross Entropy	30	93.33/64.44/68.89	5816100	0.0120	Batch Normalization applied

Inferences:

- **Improved Generalization:** Applying data augmentation significantly boosted validation and test accuracy compared to previous cases, suggesting better generalization.

- **Effect of Batch Normalization:** Batch normalization further improved training accuracy (from 71.90% to 93.33%) and validation/test accuracy (from 41.00%/51.11% to 64.44%/68.89%). This indicates that normalizing activations helped stabilize training.
- **Inference Speed:** A slight increase in parameter count (from 5,814,116 to 5,816,100) led to a small decrease in inference speed (from 0.0124 FPS to 0.0120 FPS). This trade-off suggests that batch normalization adds slight computational overhead but is beneficial overall.
- **Potential for Further Improvement:** While performance improved significantly, there is still a noticeable gap between training and validation/test accuracy, indicating potential overfitting.

Next Steps:

- **Adjust Network Architecture:**
  - Increase/decrease convolutional layers or kernel sizes.
  - Experiment with dropout to prevent overfitting.
  - Try different activation functions (Leaky ReLU, Swish).
- **Visualize Feature Maps:** Examine intermediate layers to understand what the CNN is learning.
- **Train on a Subset & Verify:** Train on a smaller dataset and check if the model can at least memorize it. If not, there's a fundamental issue with the architecture or data processing.

6 Task - 3: CAPTCHA Generation

- Evaluation Metrics: Character and Word Accuracy.

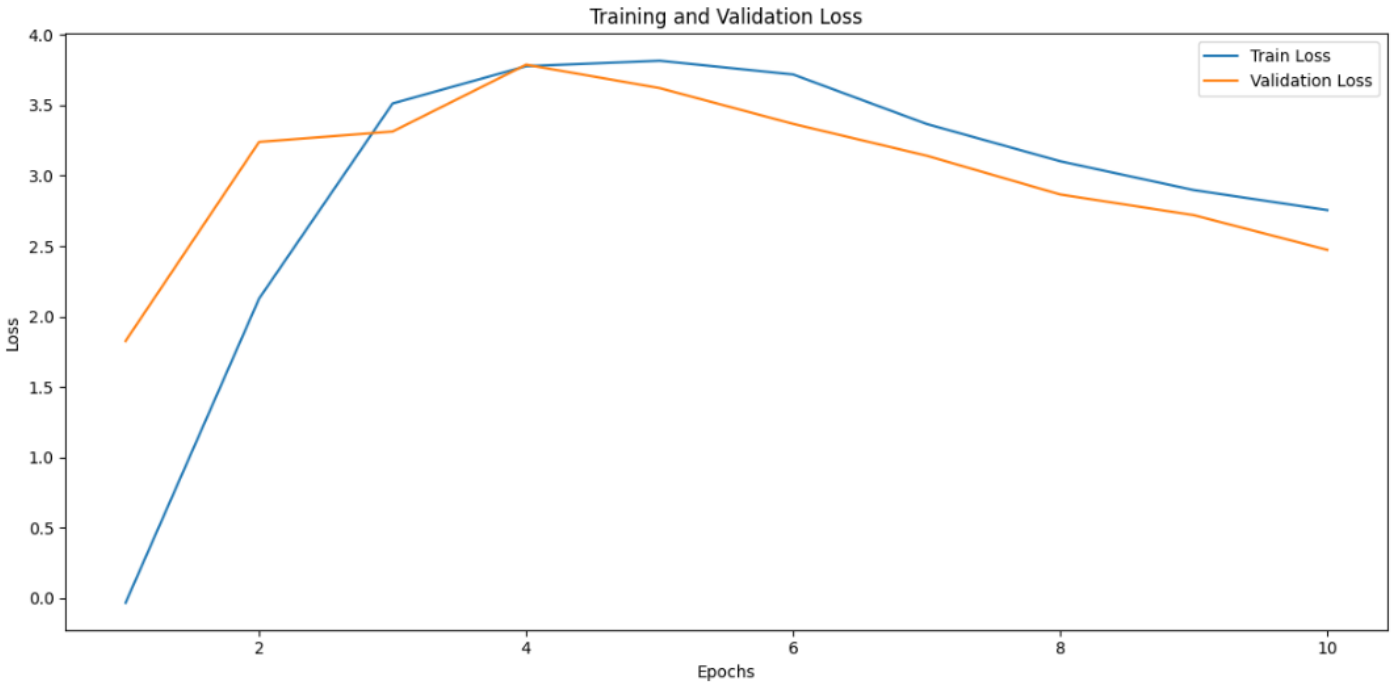

Figure 20: CAPTCHA Graphs of Model (Graph 1)

Inferences for Graph 1: Training and Validation Loss (15 Epochs)

- **Initial Increase in Loss:** Both training and validation loss exhibit an initial rise before reaching a peak around the 4th epoch. This indicates the model is adjusting weights while possibly overfitting on early epochs.
- **Stabilization Post-Peak:** After the 4th epoch, training and validation loss decrease consistently, reflecting effective optimization and reduction in overfitting.

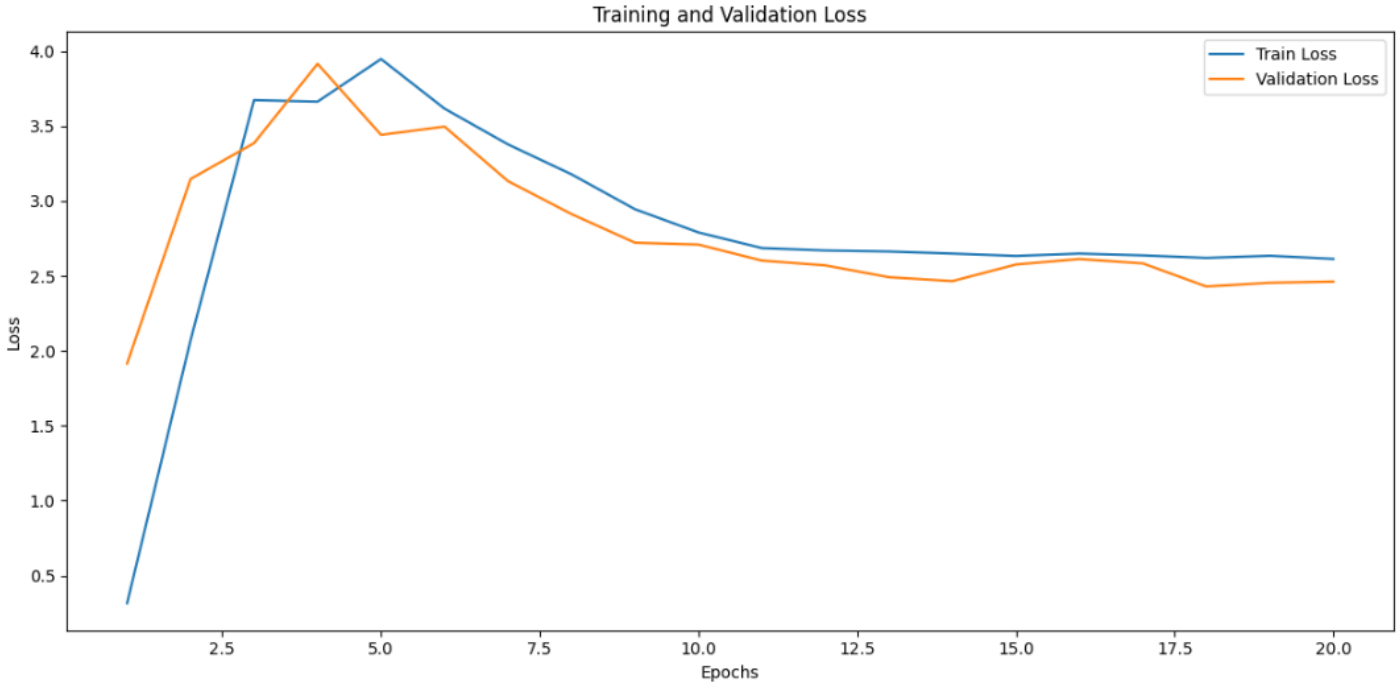

Figure 21: CAPTCHA of the Model (Graph 2)

- **Convergence:** By the 15th epoch, the gap between training and validation loss has reduced, suggesting improved generalization.

Next Steps for Graph 1:

- **Extended Training:** Train for additional epochs to ensure further convergence and monitor for signs of overfitting.
- **Learning Rate Tuning:** Experiment with learning rate decay to smooth the loss trajectory.
- **Augmentation and Regularization:** Consider adding more data augmentation and dropout to further reduce the initial divergence.

Inferences for Graph 2: Training and Validation Loss (10 Epochs)

- **Peak at 4th Epoch:** Similar to Graph 1, both training and validation loss peak around the 4th epoch, indicating this is a critical point in the training process.
- **Faster Convergence:** By the 10th epoch, both losses are declining smoothly, suggesting the model's performance is improving.
- **Validation Lag:** Validation loss remains consistently higher than training loss, reflecting some level of overfitting or an inherent complexity in the validation data.

Next Steps for Graph 2:

- **Regularization Techniques:** Employ L2 regularization or dropout to address the training-validation gap.
- **Extended Validation Monitoring:** Analyze validation performance on individual data batches to identify specific challenges.
- **Optimizer Adjustments:** Test optimizers like Adam or RMSprop to observe their impact on convergence.

References

[1] BuiltWith. Captcha usage distribution in the top 1 million sites. Online. Accessed: Oct. 15, 2023.

[2] Z. Derea, B. Zou, A. A. Al-Shargabi, A. Thobhani, and A. Abdussalam. Deep learning based captcha recognition network with grouping strategy. *Sensors*, 23(23):9487, 2023.

[3] Huthaifa Mohammed Kanoosh, A. F. Abbas, N. N. Kamal, Zainab Mejeed Khadim, D. A. Majeed, and Sameer Algburi. Image-based captcha recognition using deep learning models. In *Proceedings of the Conference on Cognitive Models and Artificial Intelligence*, pages 273–278, May 2024.

[4] Huthaifa Mohammed Kanoosh, A. F. Abbas, N. N. Kamal, Zainab Mejeed Khadim, D. A. Majeed, and Sameer Algburi. Image-based captcha recognition using deep learning models. In *Proceedings of the Cognitive Models and Artificial Intelligence Conference*, 2024.

[5] Poonam Shourie, V. Anand, D. Upadhyay, Swati Devliyal, S. Gupta, and Gunjan Shandilya. Unveiling captchas: Advanced cnn techniques for high-accuracy recognition. In *Proceedings of the International Conference on Computing and Communications Systems (IICCCS)*, pages 1–6, Sep 2024.

[6] X. Wan, J. Johari, and F. A. Ruslan. Adaptive captcha: A crnn-based text captcha solver with adaptive fusion filter networks. *Applied Sciences*, 14(12):5016, 2024.

[7] Y. Wang, Y. Wei, M. Zhang, Y. Liu, and B. Wang. Make complex captchas simple: A fast text captcha solver based on a small number of samples. *Information Sciences*, 578:181–194, Nov 2021.

[8] W. Xing, M. R. S. Mohd, J. Johari, and F. A. Ruslan. A review on text-based captcha breaking based on deep learning methods. In *2023 International Conference on Computer Engineering and Distance Learning (CEDL)*, pages 171–175, Shanghai, China, 2023.