Neural CAPTCHA Recognition System

Why Neural OCR?

Classical OCR Limitations:

- Handcrafted features (edge maps, stroke patterns)
- Fail on font variations
- Cannot handle noise or distortions
- No adaptation to conditional rendering

Neural Approach Advantages:

- Learn features automatically
- Generalize across fonts
- Handle noise robustly
- Adapt to complex patterns

Four Core Tasks

1. Dataset Generation

✓ Synthesized 3,000 CAPTCHA images

✓ Three difficulty levels

2. Classification

✓ 100-word vocabulary

✓ CNN architectures

3. Text Extraction

✓ Variable-length OCR

✓ Seq2seq with attention

4. Conditional Rendering

✓ Color-based transformations

✓ Reversed text challenge

Three Complexity Levels

Dataset	Images	Characteristics	Difficulty Score
Easy	1,000	Fixed font, white background	**0.253**
Hard	1,000	6 fonts, noise, distortions	0.780
Bonus	1,000	Conditional: green=normal, red=reversed	**0.869**

Key Innovation:

- Quantitative difficulty scoring system
- Systematic complexity progression
- Complete metadata tracking for analysis

Easy Dataset Examples

Clean, Readable CAPTCHAs

```
Background: White (255,255,255) Font: DejaVu Sans (28-32pt) Noise: None Distortion: None
```

Performance Achieved:

- Classification: **95.5% accuracy**
- OCR: **90.5% exact match**
- CER: **0.041**

Hard Dataset Examples

Complex, Noisy CAPTCHAs

```
Backgrounds: Variable (200-255 RGB) Fonts: 6 families (DejaVu, Liberation) Noise: Gaussian (σ=0.02), Salt-pepper Distortions: Rotation ±5°, Shear ±0.2
```

Performance Achieved:

- Classification: **4.5% accuracy**
- OCR: **4% exact match**
- CER: **0.846**

Conditional Rendering Challenge

The Rule:

- **Green Background**: Display text normally
- **Red Background**: Display text reversed
- **Critical**: Label remains unchanged!

Example:

Background	Display	Label
Green	"hello"	"hello"
Red	"olleh"	"hello"

Performance: 11% exact match

Architecture - Classification

CNN Design Choices

LightweightCNN (Easy Dataset):

```
Conv(3→32) → Conv(32→64) → Conv(64→128) AdaptivePool → FC(512) → FC(256) → FC(100)
```

ImprovedCNN (Hard/Bonus):

```
ResBlock(64→128) → ResBlock(128→256) → ResBlock(256→512) SpatialAttention → GlobalPool → Classifier
```

Key Innovation: Residual connections + Spatial attention mechanism

Architecture - Text Extraction

Sequence-to-Sequence Model

Encoder (CNN):

```
Conv1(3→64) → Pool Conv2(64→128) → Pool Conv3(128→256) Conv4(256→512) AdaptivePool(4,16)
```

Decoder (LSTM + Attention):

```
Embedding(vocab→256) LSTM(256+context→512) Attention(Bahdanau) Output(512→vocab)
```

Training Strategy

Hyperparameter Selection

Parameter	Value	Justification
Learning Rate	1e-3	Standard for Adam optimizer
Batch Size	32	Memory/stability balance
Optimizer	Adam	Adaptive learning rates
Scheduler	ReduceLROnPlateau	Automatic adjustment
Dropout	0.3-0.5	Based on dataset complexity

Total Training Time: ~8 hours on single GPU

Performance Across Tasks

Classification (100-word vocabulary):

Dataset	Accuracy	Training Time
Easy	95.5%	10s
Hard	4.5%	169s
Bonus	5.0%	170s

Text Extraction (OCR):

Dataset	Exact Match	CER	WER
Easy	90.5%	0.041	0.095
Hard	4.0%	0.846	0.960
Bonus	11.0%	0.788	0.890

Key Finding #1

Complexity Barrier

Performance Degradation:

- Easy → Hard: 91% accuracy drop
- Linear complexity increase → Exponential performance decrease

Evidence:

```
Easy: Train=91%, Val=95% (No overfitting) Hard: Train=66%, Val=4.5% (Severe overfitting) Bonus: Train=97%, Val=5% (Extreme overfitting)
```

Insight: Current architectures fail to extract invariant features

Key Finding #2

Attention Mechanism Failure

Observation:

Attention weights remain **uniform** across image regions

Hypothesis:

Noise overwhelms attention's focusing ability

Evidence:

- No performance improvement with attention on hard dataset
- Attention maps show no meaningful patterns
- Model defaults to global features

Implication: Need noise-robust attention mechanisms

Key Finding #3

Conditional Rendering Challenge

Bidirectional Processing Required

Current Limitation:

- Unidirectional LSTM: left-to-right processing
- Reversed text: requires right-to-left understanding

Performance Gap:

- Green (normal): 18% exact match
- Red (reversed): **4% exact match**

Solution: Bidirectional LSTM or Transformer architectures

Why Models Struggle

1. Overfitting (Hard/Bonus)

Train: 97.75%, Val: 5%

Memorization vs. generalization

2. Feature Extraction

6 fonts too diverse for CNN

Loss plateaus at epoch 85

3. Sample Inefficiency

800 training samples insufficient

High-dimensional feature space

4. Architecture Mismatch

CNN+LSTM inadequate for conditional logic

Need specialized architectures

Surprising Discoveries

1. Easy CAPTCHAs are Broken

95.5% accuracy with simple CNN

→ Modern CAPTCHAs must use complexity

2. Training Dynamics Anomaly

Catastrophic forgetting in bonus dataset

Model alternates between patterns

3. Error Patterns

Model defaults to high-frequency words

"elephant" → "freedom" (semantic bias)

Evidence-Based Solutions

1. Curriculum Learning

- Start with easy samples
- Gradually increase difficulty
- Expected: 20-30% improvement

2. Multi-Task Learning

- Auxiliary task: predict background color
- Helps with conditional logic

3. Transformer Architecture

- Better at long-range dependencies
- Bidirectional by design

4. Data Augmentation

- Generate more training samples
- Online augmentation during training

Resource Optimization

Efficiency Strategies Implemented

Computational:

- Adaptive pooling (reduces dimensions)
- Batch normalization (faster convergence)
- Early stopping criteria

Memory:

- Dynamic vocabulary loading
- Gradient accumulation
- Efficient data loaders

Result: Complete pipeline runs on single GPU in ~8 hours

Theoretical Understanding

Architecture Justification:

- **CNN**: Exploits spatial locality in images
- **LSTM**: Handles sequential dependencies in text
- **Attention**: Focuses on relevant regions (theory vs. practice gap)
- **Residuals**: Maintains gradient flow in deep networks

Loss Function Analysis:

- CrossEntropy vs. CTC trade-offs documented
- Teacher forcing implications analyzed

Hyperparameter Impact:

- Dropout correlation with dataset complexity (0.3→0.5)
- Learning rate scheduling effectiveness measured

Key Takeaways

✓ Achievements

- Complete CAPTCHA pipeline implementation
- 95.5% accuracy on simple CAPTCHAs
- Comprehensive failure analysis
- Novel conditional rendering dataset

📊 Insights

- 91% complexity barrier identified
- Attention mechanisms need noise robustness
- Conditional logic requires specialized architectures

🔬 Research Value

- Benchmark for CAPTCHA difficulty
- Evidence of neural OCR limitations
- Actionable improvement strategies

Thank You

Questions & Discussion

Repository: Available with all code and results

Key Metrics:

- 3,000 images generated
- 6 models trained
- 8 hours total training
- 95.5% best accuracy