Report_Cnn

Enhanced Stamp Detection (StaVer) - Project Report

Author: Kandil Safouane
Course: II-CCN3
Specialization: Advanced Deep Learning for Document Analysis
Date: November 25, 2025

Executive Summary

This project implements an advanced deep learning solution for automated stamp detection in scanned documents using a custom Enhanced U-Net architecture. The system achieves high-precision pixel-level segmentation of stamps from 200 DPI document scans, demonstrating significant improvements over baseline approaches through architectural enhancements and sophisticated training strategies.

1. Project Overview

1.1 Objective

Develop a robust convolutional neural network capable of accurately detecting and segmenting stamps in scanned documents at the pixel level, suitable for document verification and authentication workflows.

1.2 Dataset

- **Source:** StaVer Dataset (Stamp Verification)
- **Size:** 400 annotated document scans
- **Resolution:** 200 DPI
- **Annotations:** Pixel-level binary masks
- **Split Ratio:** 80% training, 10% validation, 10% testing
  - Training: 320 samples
  - Validation: 40 samples
  - Testing: 40 samples

1.3 Key Features & Innovations

- **High-Resolution Processing:** 512×512 input images (2× baseline resolution)
- **Data Augmentation:** Geometric transformations and photometric adjustments
- **Deep Architecture:** 3-level encoder-decoder with bottleneck
- **Regularization:** Batch normalization and dropout layers

- **Advanced Training:** Learning rate scheduling and early stopping
- **Multiple Metrics:** Comprehensive evaluation using Dice coefficient and IoU

2. Technical Architecture

2.1 Model Architecture: Enhanced U-Net

The core of this project is a modified U-Net architecture optimized for document image segmentation:

Encoder Path (Downsampling)

- **Block 1:** 64 filters → MaxPooling
- **Block 2:** 128 filters → MaxPooling
- **Block 3:** 256 filters → MaxPooling
- Each block contains:
  - Two 3×3 convolutional layers (ReLU activation)
  - Batch normalization after each convolution
  - Dropout for regularization (rates: 0.2, 0.3, 0.4)

Bottleneck

- **Filters:** 512
- Highest abstraction level with maximum dropout (0.5)
- Captures global contextual information

Decoder Path (Upsampling)

- **Block 1:** Transpose convolution → Concatenate with encoder Block 3 → 256 filters
- **Block 2:** Transpose convolution → Concatenate with encoder Block 2 → 128 filters
- **Block 3:** Transpose convolution → Concatenate with encoder Block 1 → 64 filters
- Skip connections preserve spatial details from encoder

Output Layer

- 1×1 convolution with sigmoid activation
- Produces binary segmentation mask

Total Parameters: ~7.7 million
Model Size: ~93 MB

2.2 Data Augmentation Strategy

Real-time augmentation applied to training samples:

- **Horizontal Flip:** 50% probability
- **Vertical Flip:** 50% probability
- **Rotation:** ±15° with 70% probability
- **Brightness Adjustment:** ±20% with 50% probability

All transformations maintain spatial consistency between images and masks.

2.3 Training Configuration

Optimizer: Adam (learning rate: 1e-4)
Loss Function: Binary cross-entropy
Batch Size: 8
Epochs: 20 (with early stopping)
Hardware: GPU-accelerated training (TensorFlow)

Callbacks:

1. **Early Stopping:** Halts training if validation loss doesn't improve for 5 epochs
2. **ReduceLROnPlateau:** Reduces learning rate by 50% after 3 stagnant epochs
3. **ModelCheckpoint:** Saves best model based on validation IoU

3. Evaluation Metrics

3.1 Metric Definitions

Intersection over Union (IoU):

```
IoU = |Prediction ∩ Ground Truth| / |Prediction ∪ Ground Truth|
```

Measures overlap ratio between predicted and actual stamp regions.

Dice Coefficient (F1 Score):

```
Dice = 2 × |Prediction ∩ Ground Truth| / (|Prediction| + |Ground Truth|)
```

Harmonic mean of precision and recall, more sensitive to small objects.

Pixel Accuracy:

```
Accuracy = Correct Pixels / Total Pixels
```

Percentage of correctly classified pixels.

3.2 Performance Results

Test Set Performance:

- **Accuracy:** 99.2%
- **IoU:** 0.87
- **Dice Coefficient:** 0.93
- **Loss:** 0.042

These metrics indicate excellent segmentation quality with minimal false positives/negatives.

4. Implementation Details

4.1 Environment Setup

- **Framework:** TensorFlow 2.x with Keras API
- **Compute:** GPU acceleration (CUDA-enabled)
- **Dependencies:**
  - `tensorflow` - Deep learning framework
  - `opencv-python` - Image processing operations
  - `numpy` - Numerical computations
  - `matplotlib` - Visualization
  - `scikit-learn` - Evaluation metrics
  - `kagglehub` - Dataset acquisition

4.2 Data Pipeline

1. **Download:** Automated retrieval from Kaggle using `kagglehub`
2. **Preprocessing:** Folder structure normalization to fix nested directories
3. **Loading:** Batch loading with on-the-fly resizing to 512×512
4. **Normalization:** Pixel values scaled to [0, 1] range
5. **Augmentation:** Applied during training data generation
6. **Batching:** TensorFlow Dataset API with prefetching for efficiency

4.3 Training Workflow

1. Initialize model with random weights
2. Load training and validation datasets
3. Configure callbacks for adaptive training
4. Train for up to 20 epochs
5. Restore best weights based on validation IoU
6. Evaluate on held-out test set
7. Save final model in H5 format

5. Results & Analysis

5.1 Training Convergence

The model demonstrated stable training with consistent improvement across all metrics:

- Loss decreased from ~0.4 to ~0.04 over 15 epochs
- Validation metrics closely tracked training metrics (no overfitting)
- Learning rate reduction triggered twice, enabling fine-tuning
- Early stopping activated after epoch 18

5.2 Qualitative Analysis

Visual inspection of predictions reveals:

- **Strengths:**
  - Accurate boundary detection even with irregular stamp shapes
  - Robust to document background noise and text occlusion
  - Handles partial stamps at document edges
- **Limitations:**
  - Occasional over-segmentation with very faint stamps
  - Some confusion with dark handwritten signatures
  - Performance degrades slightly with stamp rotations >30°

5.3 Comparison with Baseline

Improvements over standard U-Net (256×256, no augmentation):

- **IoU:** +0.12 (0.75 → 0.87)
- **Dice:** +0.08 (0.85 → 0.93)
- **Training Stability:** Reduced variance in validation metrics
- **Inference Quality:** Fewer false positives on complex backgrounds

6. Key Technical Decisions

6.1 Why 512×512 Resolution?

Higher resolution preserves fine stamp details (text, intricate borders) that are lost at 256×256. The trade-off in training speed is acceptable given the limited dataset size.

6.2 Why Batch Normalization?

- Accelerates convergence by stabilizing activations
- Acts as implicit regularization

- Enables higher learning rates

6.3 Why Multiple Metrics?

- **IoU:** Standard benchmark for comparison
- **Dice:** Better for class imbalance (stamps are small relative to background)
- **Accuracy:** Interpretable for non-technical stakeholders

6.4 Data Augmentation Rationale

With only 320 training samples, augmentation is critical to:

- Prevent overfitting
- Improve generalization to unseen document variations
- Simulate real-world scanning conditions (rotation, lighting)

7. Deployment Considerations

7.1 Inference Performance

- **Processing Time:** ~50-100ms per image on GPU
- **Batch Processing:** Supports multiple documents simultaneously
- **Model Format:** Standard H5 (compatible with TensorFlow Serving)

7.2 Integration Points

The trained model can be integrated into:

- Document management systems
- Automated verification workflows
- Mobile scanning applications (with optimization)
- Cloud-based API services

7.3 Optimization Opportunities

For production deployment:

- **TensorFlow Lite:** Convert for mobile/edge devices
- **TensorRT:** Optimize for NVIDIA GPUs
- **Quantization:** Reduce model size (93MB → <25MB)
- **ONNX Export:** Enable cross-framework compatibility

8. Future Enhancements

8.1 Model Improvements

- **Attention Mechanisms:** Add spatial attention to focus on stamp regions
- **Multi-Scale Processing:** Process images at multiple resolutions
- **Ensemble Methods:** Combine predictions from multiple models
- **Transfer Learning:** Fine-tune from pre-trained ImageNet weights

8.2 Dataset Expansion

- Collect more training samples (target: 1,000+)
- Include stamps from different countries and eras
- Add challenging cases: colored stamps, overlapping text, damaged documents
- Create synthetic training data using domain randomization

8.3 Additional Features

- **Multi-Class Segmentation:** Distinguish different stamp types
- **Stamp Classification:** Identify specific stamp categories
- **Confidence Scores:** Output uncertainty estimates
- **Post-Processing:** Apply morphological operations to refine masks

9. Conclusion

This project successfully developed a high-performance stamp detection system using deep learning. The Enhanced U-Net architecture, combined with careful preprocessing and training strategies, achieves 87% IoU and 93% Dice coefficient on the test set—suitable for real-world document processing applications.

The implementation demonstrates best practices in computer vision engineering:

- Systematic data handling
- Robust augmentation strategies
- Principled architecture design
- Comprehensive evaluation methodology

The modular codebase and documented approach provide a solid foundation for further research and deployment in production environments.

10. References & Resources

Dataset:

- StaVer (Stamp Verification) Dataset: https://www.kaggle.com/datasets/rtatman/stamp-verification-staver-dataset

Architecture:

- Original U-Net Paper: Ronneberger et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation" (2015)

Framework Documentation:

- TensorFlow: https://www.tensorflow.org/
- Keras API: https://keras.io/

Evaluation Metrics:

- IoU/Jaccard Index: Standard semantic segmentation metric
- Dice Coefficient: Medical imaging segmentation benchmark

Appendix: Model Summary

```
Model: "Enhanced_UNet"
_________________________________________________________________
Total params: 7,767,809
Trainable params: 7,761,665
Non-trainable params: 6,144
_________________________________________________________________
Input Shape: (None, 512, 512, 3)
Output Shape: (None, 512, 512, 1)
```

Training Hardware: GPU-accelerated (CUDA-enabled)
Training Duration: ~30 minutes
Final Model File: `enhanced_stamp_model.h5` (93 MB)