Comparative Analysis of Support Vector Machines and Convolutional Neural Networks for Digit Classification

Biruk Gebru Jember

August 13, 2025

1 Theoretical Foundation

1.1 Maximal Margin Classifiers

Maximal margin classifiers find the optimal hyperplane that separates two classes with maximum margin. For data points (x_i, y_i) where $y_i \in \{-1, 1\}$, the hyperplane is:

$$w^T x + b = 0 \tag{1}$$

The goal is to maximize $\frac{2}{\|w\|}$ subject to:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i \tag{2}$$

1.2 Support Vector Classifiers (SVC)

SVCs extend maximal margin classifiers for non-separable data using slack variables ξ_i:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \tag{3}$$

subject to:

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad \forall i \tag{4}$$

1.3 Hard vs Soft Margin Classification

Hard Margin: Requires perfect linear separability with strict constraints $y_i(w^T x_i + b) \geq 1 \quad \forall i$.
Limitations:

- Only works with perfectly separable data
- Sensitive to outliers and noise
- May overfit

Soft Margin: Allows misclassifications using slack variables $y_i(w^T x_i + b) \geq 1 - \xi_i$.
Advantages:

- Handles noisy and non-separable data
- More robust to outliers
- Better generalization with parameter C control

Why Soft Margin is Better: Real-world data is rarely perfectly separable, making soft margin more practical and robust.

1.4 Support Vector Machines (SVMs)

SVMs use the kernel trick to map data into higher-dimensional spaces for linear separation. The dual formulation is:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{5}$$

subject to:

$$0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6}$$

where $K(x_i, x_j)$ is the kernel function and α_i are Lagrange multipliers.

1.5 Hyperplanes in Machine Learning

A hyperplane in $\mathbb{R}^n$ is a flat affine subspace defined by $w^T x + b = 0$ where w is the normal vector and b is the bias term.

Properties:

- Divides space into two half-spaces
- Distance from point x: $d = \frac{|w^T x + b|}{\|w\|}$
- Margin width: $\frac{2}{\|w\|}$

SVMs use hyperplanes as decision boundaries, maximizing the margin between classes.

1.6 Kernel Functions

Kernel functions $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ enable SVMs to work in high-dimensional spaces without explicit coordinate computation.

Common Kernels:

- **RBF/Gaussian**: $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$
- **Polynomial**: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$
- **Linear**: $K(x_i, x_j) = x_i^T x_j$

Key Parameters:

- γ: Controls influence of training examples (RBF)
- C: Regularization parameter balancing margin and errors
- d: Polynomial degree

2 Implementation and Methodology

2.1 Experimental Setup

This study compares three approaches for MNIST digit classification:

1. **CNN**: End-to-end deep learning approach
2. **CNN + SVM**: Hybrid approach using CNN for feature extraction and SVM for classification
3. **Pure SVM**: Traditional machine learning approach on raw pixel data

2.2 Dataset and Preprocessing

MNIST dataset: 70,000 handwritten digit images (28×28 pixels) with 60,000 training and 10,000 test samples.

Preprocessing:

- Normalization: Pixel values scaled to [0,1] range
- Reshaping: CNN input to (28, 28, 1) for grayscale
- Flattening: SVM input to 784-dimensional vectors

2.3 CNN Architecture

CNN implemented using TensorFlow/Keras:

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2,2)),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
```

Components:

- Convolutional Layers: Extract hierarchical features
- MaxPooling: Reduce dimensions and provide translation invariance
- Dense Layers: Learn non-linear feature combinations
- Softmax: Output probability distribution

2.4 Feature Extraction and SVM Implementation

For CNN+SVM, 64-dimensional features extracted from penultimate dense layer:

```
feature_extractor = Model(inputs=model.input,
                          outputs=model.layers[-2].output)
X_features = feature_extractor.predict(X_data)
```

SVM implementation using scikit-learn with RBF kernel:

```
svm = SVC(kernel='rbf', C=10, gamma=0.05)
```

Training Strategy:

- CNN: 5 epochs with Adam optimizer
- CNN+SVM: 10,000 samples for SVM training
- Pure SVM: 5,000 samples due to computational constraints

3 Results and Analysis

3.1 Performance Comparison

Table 1: Model Performance Comparison

Metric	CNN	CNN+SVM	Pure SVM
Test Accuracy	0.987	0.989	0.954
Training Time (s)	27.3	139.3	164.4
Model Complexity	High	High	Medium

3.2 Detailed Classification Results

Table 2: Per-Class Performance Metrics

Digit	Model	Precision	Recall	F1-Score
0	CNN	0.99	0.99	0.99
	CNN+SVM	0.99	0.99	0.99
	Pure SVM	0.97	0.99	0.98
1	CNN	0.99	1.00	1.00
	CNN+SVM	0.99	1.00	0.99
	Pure SVM	0.97	0.99	0.98
2	CNN	0.98	0.99	0.98
	CNN+SVM	0.99	0.99	0.99
	Pure SVM	0.95	0.95	0.95
3	CNN	0.98	0.99	0.99
	CNN+SVM	0.99	0.99	0.99
	Pure SVM	0.94	0.95	0.95
4	CNN	1.00	0.99	0.99
	CNN+SVM	0.99	0.99	0.99
	Pure SVM	0.93	0.97	0.95

4 Conclusions

4.1 Key Findings

1. **CNN+SVM achieves highest accuracy** (98.9%), demonstrating effectiveness of combining deep feature extraction with traditional ML classification.

2. **CNN performs excellently** (98.7%) with fastest training time (27.3 seconds), making it most efficient approach.

3. **Pure SVM shows respectable performance** (95.4%) despite using only raw pixel data, highlighting kernel methods effectiveness.

4. **Training time varies significantly**: CNN is 5-6 times faster than SVM-based approaches.

4.2 Recommendations

- **Production systems**: Use CNN for optimal balance of accuracy and speed
- **Interpretability**: Consider pure SVM for clear decision boundaries
- **Maximum accuracy**: Employ CNN+SVM hybrid with sufficient computational resources
- **Real-time applications**: CNN preferred due to fast inference

4.3 Theoretical Insights

Results validate:

- Kernel effectiveness: RBF kernel captures non-linear patterns in raw pixel data
- Feature hierarchy: CNN's learned features provide superior representations
- Hybrid approaches: Combining deep learning with traditional ML yields improvements
- Computational trade-offs: Higher accuracy comes with increased computational cost