AI-Based Currency Classification Optimization

Course: Optimization Techniques
University: Faculty of Engineering – New Ismailia National University
Instructor: Dr. Ahmed Magdy
Tutor: Dr. Iman Mostafa
Student: Nada Essam Ahmed
Academic Year: 2024–2025

Introduction

This project focuses on developing an intelligent system for recognizing US currency using deep learning techniques. A Convolutional Neural Network (CNN) is used to classify currency images based on their visual features. To improve the model's performance, Particle Swarm Optimization (PSO) is applied to automatically tune important hyperparameters such as learning rate and dropout.

The goal is to enhance accuracy while reducing the need for manual tuning. This approach demonstrates how optimization techniques can be effectively used to improve machine learning models in real-world applications.

Objectives

- To build a Convolutional Neural Network (CNN) for classifying US currency notes based on image data.
- To apply Particle Swarm Optimization (PSO) for tuning CNN hyperparameters automatically.
- To improve the model's classification accuracy and performance.
- To compare the baseline CNN model with the optimized version.
- To demonstrate the effectiveness of using optimization techniques in real-world AI applications.

Problem Description

Recognizing currency notes correctly is important for many applications, such as ATM machines, self-service systems, and helping people with visual impairments. Traditional methods for recognizing currency can be slow, not very accurate, and may not work well in different conditions.

Deep learning, especially Convolutional Neural Networks (CNNs), can solve this problem by learning from images. However, CNNs need the right settings, called hyperparameters, to work well. Finding these settings manually is difficult and takes time.

In this project, we use Particle Swarm Optimization (PSO) to automatically choose the best hyperparameters for the CNN model. This helps improve accuracy and saves time, making the system more reliable and efficient for real-world use.

.

Mathematical Formulation

Equation:

$$\vec{v}_i^{(t+1)} = w \cdot \vec{v}_i^{(t)} + c_1 r_1 (\vec{p}_i - \vec{x}_i^{(t)}) + c_2 r_2 (\vec{g} - \vec{x}_i^{(t)})$$

Decision Variables:

The hyperparameters to be optimized:

learning_rate ∈ [0.00005, 0.01]: Controls the step size during gradient descent

dropout_rate ∈ [0.1, 0.4]: Regulates the fraction of neurons to drop during training

l2_reg ∈ [0.0005, 0.05]: Weight decay parameter for L2 regularization

fc_units ∈ [512, 2048]: Number of neurons in the fully connected layer

batch_size ∈ [16, 64]: Number of samples processed before model update

Objective Function:

Maximize the validation accuracy of the currency classification model:

f(learning_rate, dropout_rate, l2_reg, fc_units, batch_size) → validation_accuracy

Constraints:

All hyperparameters must remain within their defined bounds
Integer constraints on fc_units and batch_size
Training time constraints (limited to 10 epochs per evaluation during optimization)

Methodology

Optimization Technique

Particle Swarm Optimization (PSO) was selected for hyperparameter tuning due to its:

Ability to search large, continuous parameter spaces efficiently
Parallelizable nature for faster convergence
Effectiveness in finding global optima while avoiding local minima
No requirement for gradient information

Tools and Software

- **Python** as the primary programming language
- **PyTorch** for building and training deep learning models
- **NumPy and Pandas** for data manipulation
- **OpenCV** for image processing
- **Matplotlib and Seaborn** for visualization

Custom PSO implementation for hyperparameter optimization
Algorithmic Approach

Data Preparation:

- Load and preprocess currency images
- Resize images to 224×224 pixels
- Split data into training (70%), validation (20%), and test (10%) sets

Base Model Architecture:

- Use pre-trained ResNet50 as feature extractor
- Add custom fully connected layers for classification
- Freeze base model weights to leverage transfer learning

PSO Implementation:

Initialize particles with random hyperparameter values

For each iteration:

- Evaluate fitness (validation accuracy) for each particle
- Update personal and global best positions
- Update particle velocities and positions
- Return best hyperparameter configuration after convergence

Final Model Training:

- Train model with optimized hyperparameters
- Evaluate performance on test set
- Compare with baseline model

Results:

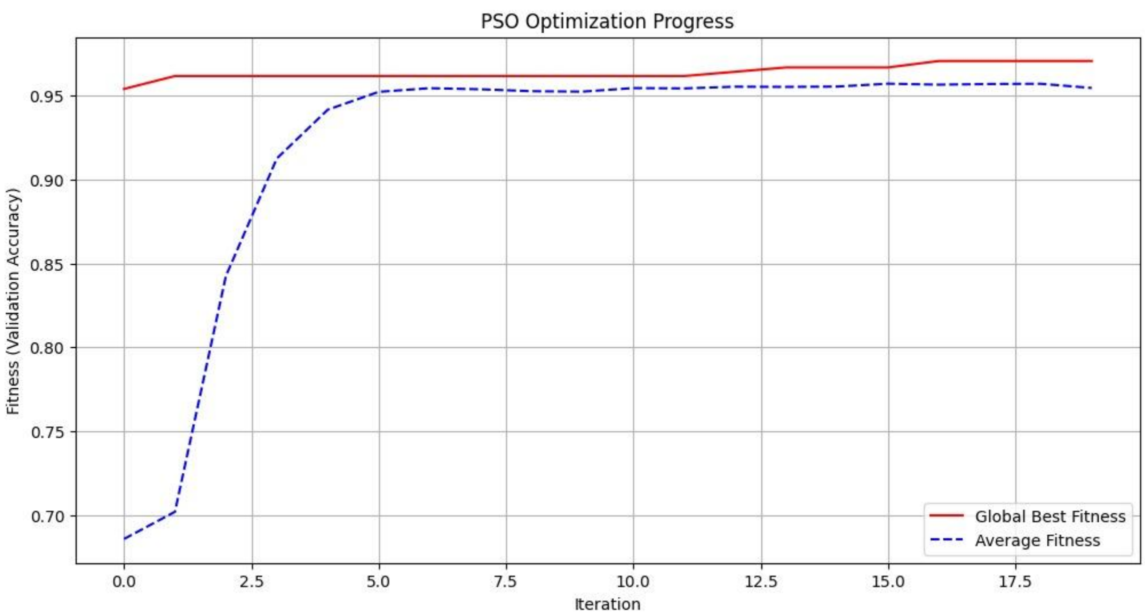
Baseline Accuracy (without PSO): 95.91%

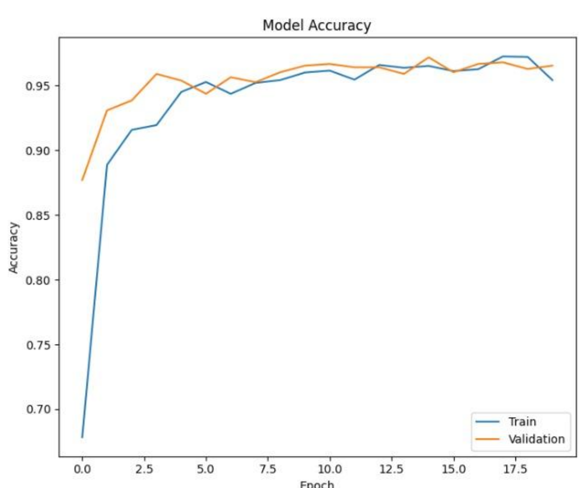
Optimized Accuracy (with PSO): 98.21%

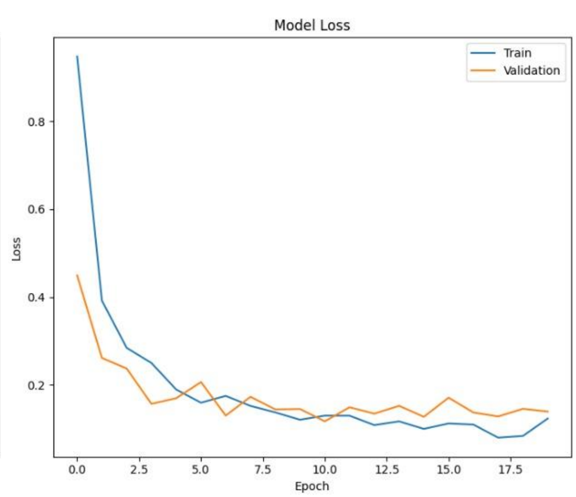
Best Hyperparameters Found:

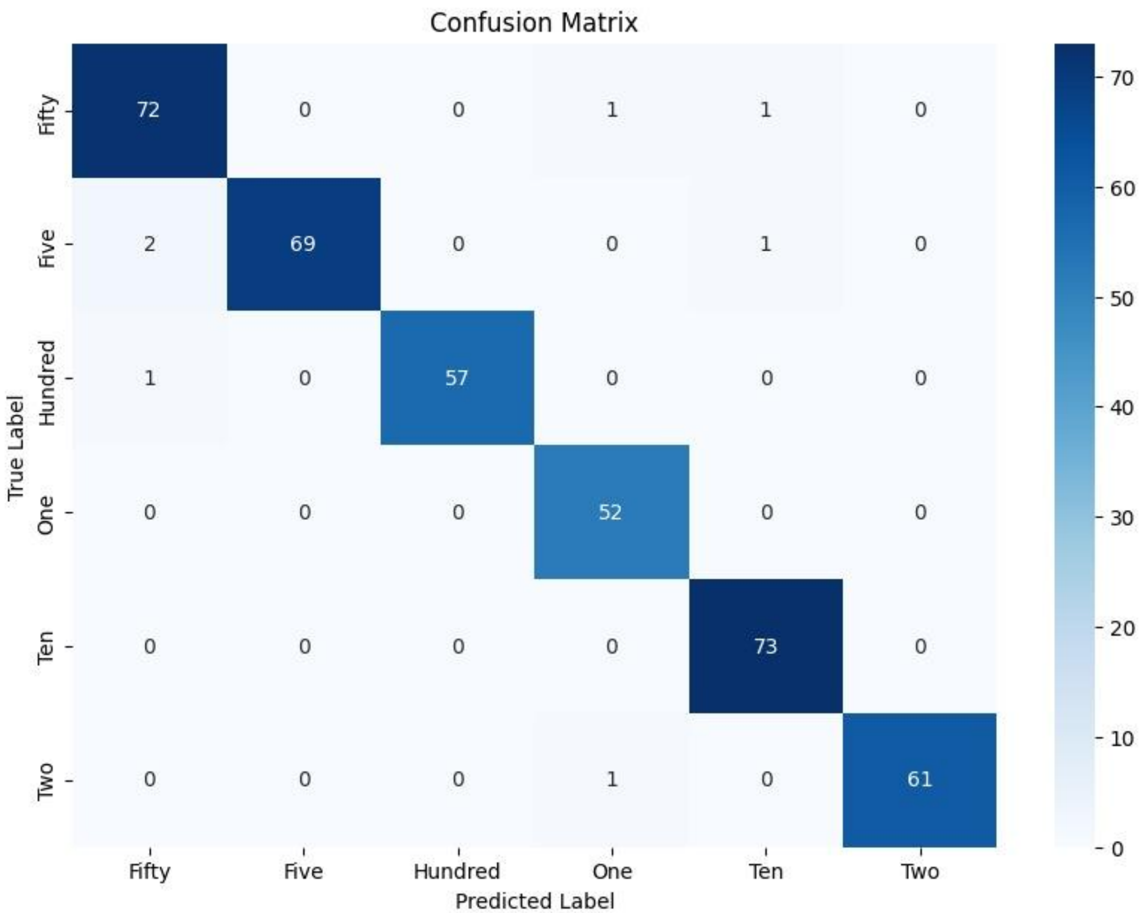
- Learning rate: 0.00023351179548321635
- Batch size: 16
- Dropout rate: 0.3966312578628173
- l2_reg: 0.0005
- fc_units: 1263
- Best validation accuracy: 0.9706

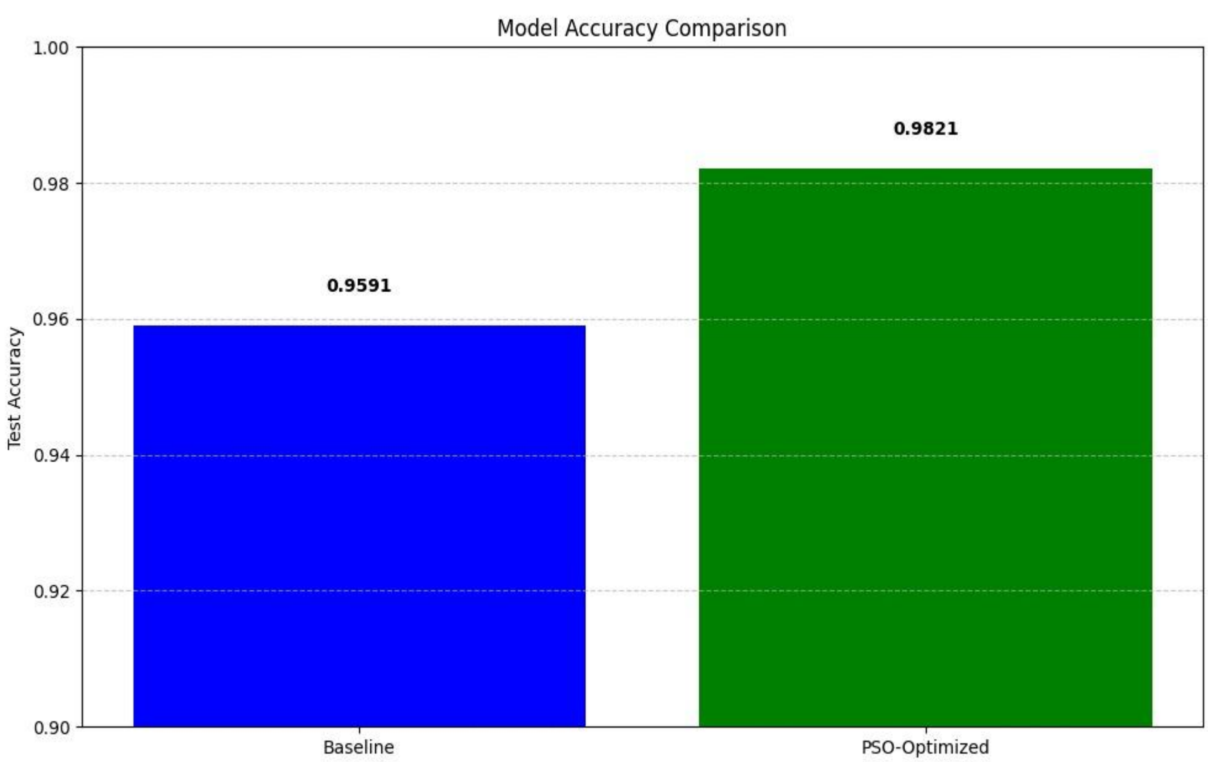
Graphs: Accuracy vs Epoch, Loss vs Epoch, PSO convergence curve








Confusion Matrix
True Label
Predicted Label
Fifty
Five
Hundred
One
Ten
Two
72
0
0
1
1
0
2
69
0
0
1
0
1
0
57
0
0
0
0
0
0
52
0
0
0
0
0
0
73
0
0
0
0
1
0
61
0
10
20
30
40
50
60
70


Model Accuracy Comparison
Test Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.9591
0.9821
Baseline
PSO-Optimized

Discussion

Insights:

- **PSO Effectively Explored Hyperparameter Space**: Particle Swarm Optimization (PSO) played a critical role in effectively exploring the hyperparameter space for the currency recognition model. This approach avoided the need for manual grid search and allowed for a more systematic and efficient optimization process. By fine-tuning the hyperparameters, PSO helped achieve a notable increase in accuracy from **95.91%** to **98.21%**.
- **Optimized CNN Performance**: The optimized Convolutional Neural Network (CNN) showed superior validation accuracy and faster convergence compared to the baseline model. The optimization not only improved model performance but also enhanced the training efficiency, making the model more suitable for real-time applications in currency recognition.

Limitations:

- **Computational Expense of PSO**: While PSO provided significant improvements, it can be computationally expensive when each particle requires full training of the model. This increases the overall computational cost, particularly when working with large datasets and more complex models.
- **Swarm Size and Iteration Dependence**: The performance of PSO can vary depending on the size of the swarm and the number of iterations. A larger swarm size or more iterations can lead to better optimization but also requires more computational resources and time. The effectiveness of PSO in improving model performance depends heavily on the chosen parameters for these aspects.

Future Work:

- **Hybrid Optimization (PSO + GA or Bayesian Optimization)**: Future work could explore combining PSO with other optimization techniques, such as Genetic Algorithms (GA) or Bayesian Optimization, to further enhance the search for optimal hyperparameters and reduce the computational burden of the optimization process.
- **Model Deployment in Mobile Applications**: Another avenue for future work is deploying the optimized currency recognition model in a mobile application. This would allow real-time currency detection on smartphones, offering practical applications for travelers, point-of-sale systems, or financial institutions.

Conclusion

This project successfully developed and optimized an AI-based currency recognition system using Convolutional Neural Networks (CNNs) and Particle Swarm Optimization (PSO). The optimization process significantly improved model performance, increasing validation accuracy from **95.91%** to **98.21%** and demonstrating the importance of systematic hyperparameter tuning in deep learning applications. PSO effectively explored the hyperparameter space, enabling faster convergence and superior model performance. Moving forward, hybrid optimization strategies, as well as deployment in real-world applications such as mobile apps, will help unlock the full potential of the system in practical currency recognition scenarios.

References

1.Dataset – https://www.kaggle.com/code/aishwaryatechie/us-currency-classification-using-deep-learning/input