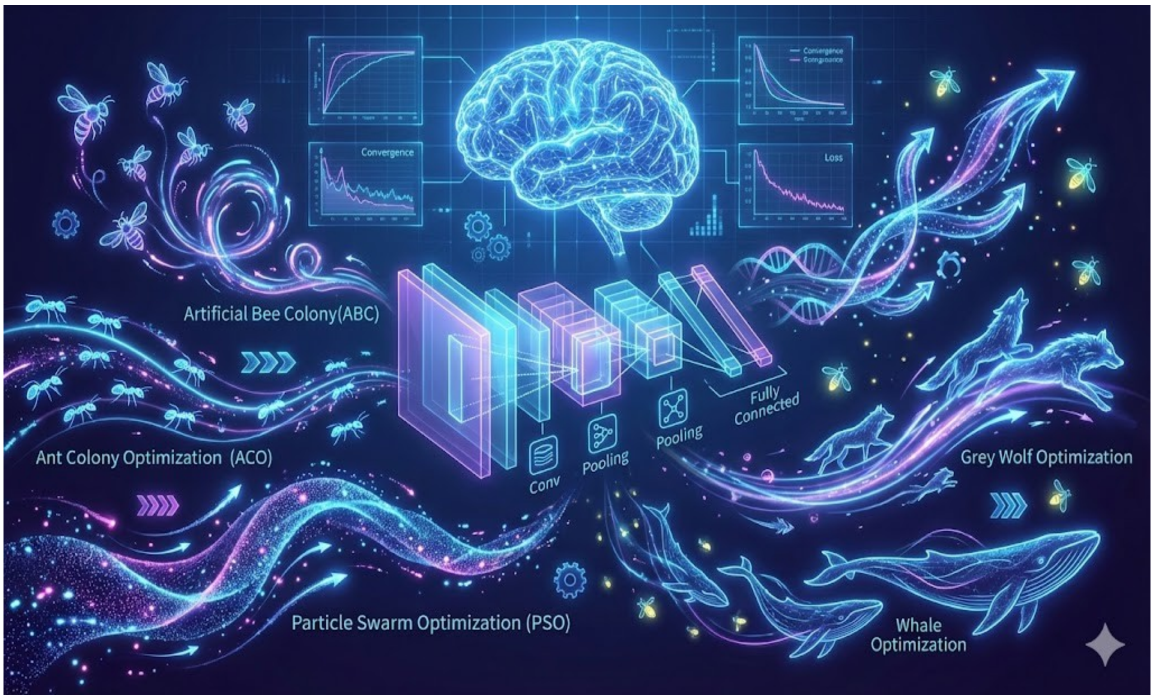
# DeepEvoOpt Project Report

**‘Hyperparameter Optimization for Deep Learning Using Evolutionary Algorithms’**




## AIvolution Team

# Team Members:

1. **Ahmed Salah Fouda**
2. **Ahmed Salah Ali**
3. **Emad Anwer Naguib**
4. **Marwan Mohamed Rashad**
5. **Youssef Fady Gamil**

# ⭐ 1. Project Overview

The **DeepEvoOpt** project is a complete optimization framework designed to improve the performance of deep learning models through **meta-heuristic optimization algorithms**. The project evaluates and compares nine different evolutionary and swarm-based optimization techniques to tune hyperparameters for:

- **Convolutional Neural Network (CNN)**
- **Feed-Forward Neural Network (MLP)**

Both models are trained on the **Fashion-MNIST** dataset.

The system is fully modular, with separate components for:

- Data loading
- Model definitions
- Optimization algorithms
- Objective evaluation
- Experiment execution
- Results logging and visualization

The repository follows a clean structure:

```
DeepEvoOpt/
|— data/
|— notebooks/
|— results/
|— src/
|   ├— models/
|   ├— optimizers/
|   ├— utils/
|   ├— train.py
|   └— run_experiments.py
```

# ⭐ 2. Models Used

**Convolutional Neural Network (CNN)**

The CNN architecture used in this project includes:

- Two convolutional layers
- ReLU activations
- MaxPooling layers
- Dropout for regularization
- Fully Connected (FC) layers for classification

All hyperparameters such as filter sizes, kernel sizes, dropout rates, and FC dimensions are tunable by the optimizers.

**Feed-Forward Neural Network (MLP)**

The MLP model consists of:

- Multiple fully connected layers
- ReLU activation
- Dropout layers
- Configurable hidden layer sizes

This makes it suitable for optimization using meta-heuristic algorithms.

## ⭐ 3. Optimization Algorithms Applied

The following **9 optimization algorithms** were implemented:

1. **Genetic Algorithm (GA)**
2. **Particle Swarm Optimization (PSO)**
3. **Grey Wolf Optimizer (GWO)**
4. **Ant Colony Optimization (ACO)**
5. **Firefly Algorithm**
6. **Artificial Bee Colony (ABC)**
7. **Opposition-Based Chaotic Whale Optimization Algorithm (OBC-WOA)**
8. **Fitness-Centered Recombination (FCR)**
9. **Fuzzy-Controlled Grey Wolf Optimizer (FCGWO)**

Each optimizer:

- Generates a population of hyperparameter sets
- Evaluates them using the unified objective function
- Trains the model for a small number of epochs
- Updates solutions iteratively
- Logs validation loss over time

## ⭐ 4. Workflow Summary

Step 1 — Define Search Space

Each hyperparameter (learning rate, dropout, filter sizes, optimizer type, etc.) has a defined search interval.

Step 2 — Run Optimization

Each meta-heuristic algorithm runs for:

- **POP_SIZE = 6** individuals
- **MAX_ITER = 5** iterations

Step 3 — Objective Function

Each candidate solution:

- Builds the CNN model
- Trains for a few epochs
- Returns validation loss

Step 4 — Logging & Visualization

All results are saved into:

`results/logs/`

`results/figures/`

The notebook plots convergence curves and compares all optimizes.

# ⭐ 5. Results Comparison (CNN Model)

Below is a summary of the **best validation loss** achieved by each optimizer as recorded in the project notebook:

| Optimizer | Best Validation Loss | Notes |
|---|---|---|
| **ACO** | **0.2314** | Best overall performance – fast convergence |
| **GWO** | **0.2453** | Very strong and stable |
| **FCGWO** | **0.2456** | Improved exploration via fuzzy-controlled parameter adaptation |
| **GA** | **0.2616** | Good start then plateaued early |
| **Firefly** | **0.2616** | Similar pattern to GA |
| **OBC-WOA** | **0.2665** | Gradual improvement, moderate performance |
| **PSO** | **0.2833** | Slower convergence, moderate accuracy |
| **FCR** | **0.2951** | Weaker performance, needs tuning |
| **ABC** | **0.3216** | Weakest performance in this experiment |

# ⭐ 6. Analysis of Results

**Why ACO performed best**

- Strong exploitation of promising solutions
- Efficient pheromone-based reinforcement
- Fast and stable convergence patterns

**GWO & FCGWO advantages**

- GWO provides a solid balance between exploration and exploitation
- FCGWO enhances GWO by adapting the parameter "a" using fuzzy logic → improved convergence quality

**GA & Firefly performance**

- Both show early improvements but struggle to escape local minima
- Require larger populations or more iterations to outperform GWO/ACO

**Why ABC scored worst**

- Sensitive to initialization
- Requires larger populations for stable results

# ⭐ 7. Conclusion

The **DeepEvoOpt** framework successfully demonstrates how meta-heuristic algorithms can be used to optimize deep learning models. Through this experiment:

- **ACO** achieved the best overall performance
- **GWO** and **FCGWO** provided strong and consistent results
- Some algorithms (ABC, FCR) require more tuning or more iterations

The project is modular, extensible, and can be further developed to:

- Train models on larger datasets
- Add more optimization algorithms
- Improve GPU utilization
- Automate full training of best-found hyperparameters