1ST MANDATORY ASSIGNMENT: NEURAL NETWORKS AND DEEP LEARNING

Georgios Lazaridis

November 2025

Aristotle University of Thessaloniki

1 Introduction and Implementation

This project implemented two **Feedforward Neural Network (FNN)** architectures—the **MLP** and the **CNN**—to solve the **Multi-Class Classification** problem on the **CIFAR-10** dataset. The goal was to compare their effectiveness against baseline methods and analyze the impact of architectural choices.

1.1 Task Definition and Data Pre-processing

- **Task:** Multi-class classification of 32×32 color images into 10 categories (e.g., dog, ship, frog).
- **Normalization:** All pixel values were scaled (normalized) to the range $[0, 1]$.
- **Target Encoding:** Labels were converted to the `One-Hot Encoding` format, which is required for the `categorical_crossentropy` loss function used in the NNs.
- **Learning Algorithm:** Both networks were trained using the fundamental **Back-propagation** algorithm, which computes the gradient of the loss function with respect to the network weights.
- **Optimizer:** The **Adam Optimizer** was selected due to its efficient stochastic gradient descent optimization, which adapts the learning rate for each network weight.
- **Loss Function:** `Categorical Cross-Entropy` was used as the loss function, suitable for evaluating the distance between the Softmax probability output and the One-Hot Encoded targets.

2 Comparative Experimentation and Results

2.1 MLP Architecture (Feature Engineering)

The MLP served as the first FNN baseline. Since it cannot process 2D images directly, an external feature extraction method was required:

- **Dimensionality Reduction:** The $32 \times 32 \times 3$ input (3072 features) was flattened and then subjected to **Principal Component Analysis (PCA)**, reducing the dimension to **100** components.

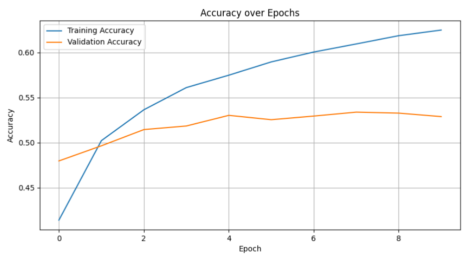
2.2 MLP (PCA) Performance Comparison

The MLP experiments focused on the impact of hidden neuron count (**H1/H2**) and epochs on performance, using PCA (100 components). The experiments using PCA achieved a peak Test Accuracy of 52.88%. However, the MLP(64/32) model (smallest capacity) performed best, suggesting larger models overfitted the limited PCA features. Furthermore, attempting to double the training time worsened the Test Accuracy, confirming severe overfitting across all MLP architectures.

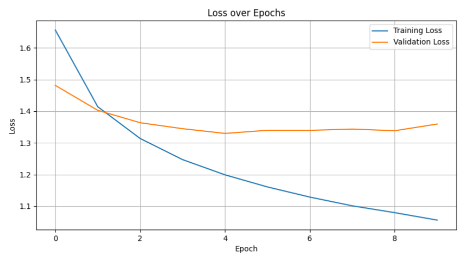
Table 1: **MLP (PCA) Comparative Results**

	Architecture (H1/H2)	**Epochs**	**Train Time (s)**	**Training Acc**	**Test Acc**
	64 / 32	10	22.83	65.94%	**52.88%**
s	128 / 64	10	22.21	64.20%	51.95%
	256 / 128	10	22.56	64.71%	52.52%
	128 / 64	20	43.22	70.04%	51.75%

Figure 1: **Training Curves for MLP (64/32): Accuracy and Loss**

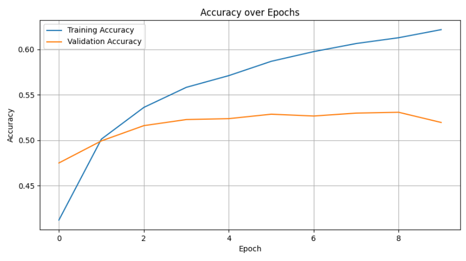


(a) MLP (64/32) Accuracy

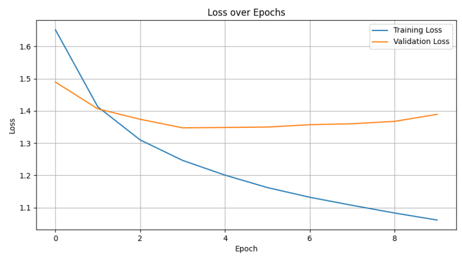


(b) MLP (64/32) Loss

Figure 2: **Training Curves for MLP (128/64): Accuracy and Loss**

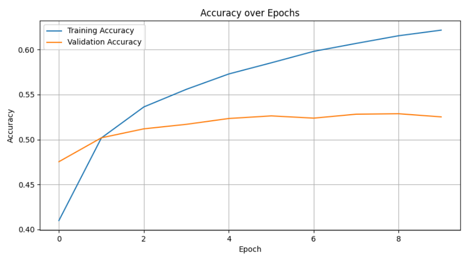


(a) MLP (128/64) Accuracy

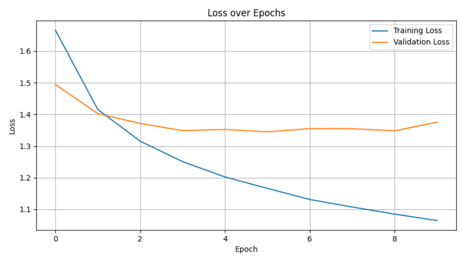


(b) MLP (128/64) Loss

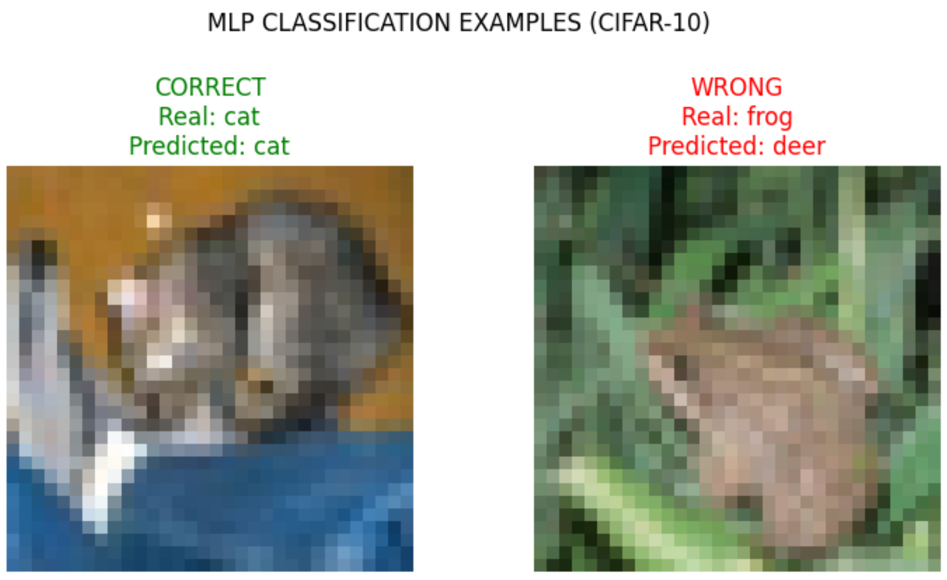
Figure 3: **Training Curves for MLP (256/128): Accuracy and Loss**



(a) MLP (256/128) Accuracy



(b) MLP (256/128) Loss



2.3 CNN Architecture (Feature Learning)

The CNN utilized internal layer components to perform feature extraction, making it highly effective for images:

- **Convolutional Layers (Conv2D):** These layers apply filters (kernels) to the input image to extract local features (e.g., edges, textures) , learning their optimal values through training.
- **Pooling Layers (MaxPooling2D):** Applied after each convolutional block, these layers down-sample the feature maps (e.g., 2×2 pooling) to reduce dimensionality, computational load, and increase robustness to minor variations .

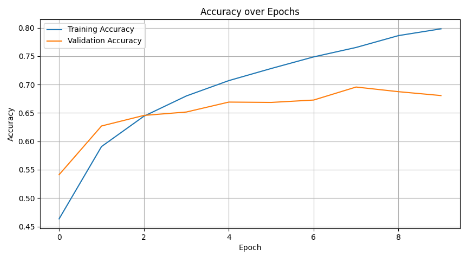
2.4 CNN Performance Comparison

The CNN experiments focused on the capacity of the convolutional layers (filters). The CNN architecture achieved significantly higher performance, peaking at 70.00% Test Accuracy with the CNN(64,128) model. The increased network capacity correlated positively with the Test Accuracy, as the largest model achieved the highest score. However, all CNN models displayed overfitting: training the CNN(32,64) for 20 epochs increased Training Accuracy to 95.87% but caused the Test Accuracy to drop (from 69.21% to 68.79%), confirming the network was memorizing noise after 10 epochs.

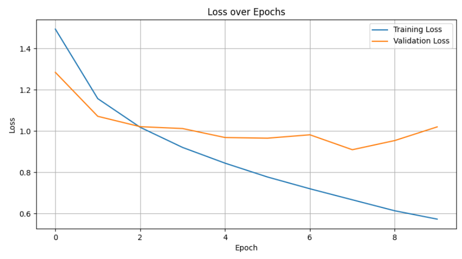
Table 2: **CNN Comparative Results**

Architecture	Epochs	Train Time (s)	Parameters	Training Acc	Test Acc
CNN(16, 32) - Dense128	10	187.41	153,962	81.57%	68.06%
CNN(32, 64) - Dense128	10	295.07	315,722	90.12%	69.21%
CNN(64, 128) - Dense128	10	704.47	666,890	**91.90%**	**70.00%**
CNN(32, 64) - Dense128	20	670.46	315,722	95.87%	68.79%

Figure 4: **Training Curves for CNN (16/32): Accuracy and Loss**

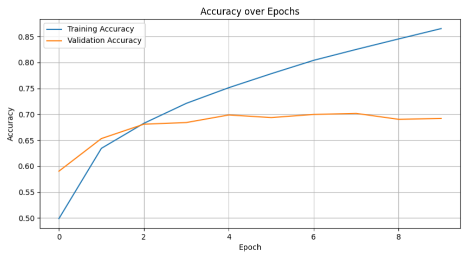


(a) CNN (16/32) Accuracy

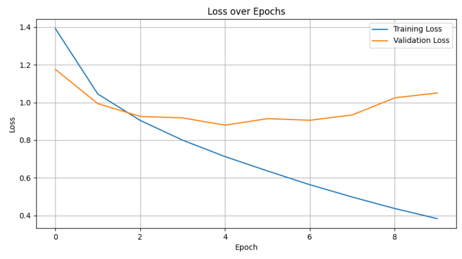


(b) CNN (16/32) Loss

Figure 5: **Training Curves for CNN (32/64): Accuracy and Loss**

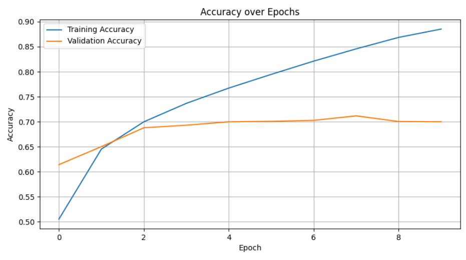


(a) CNN (32/64) Accuracy

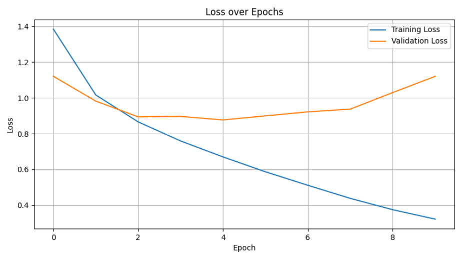


(b) CNN (32/64) Loss

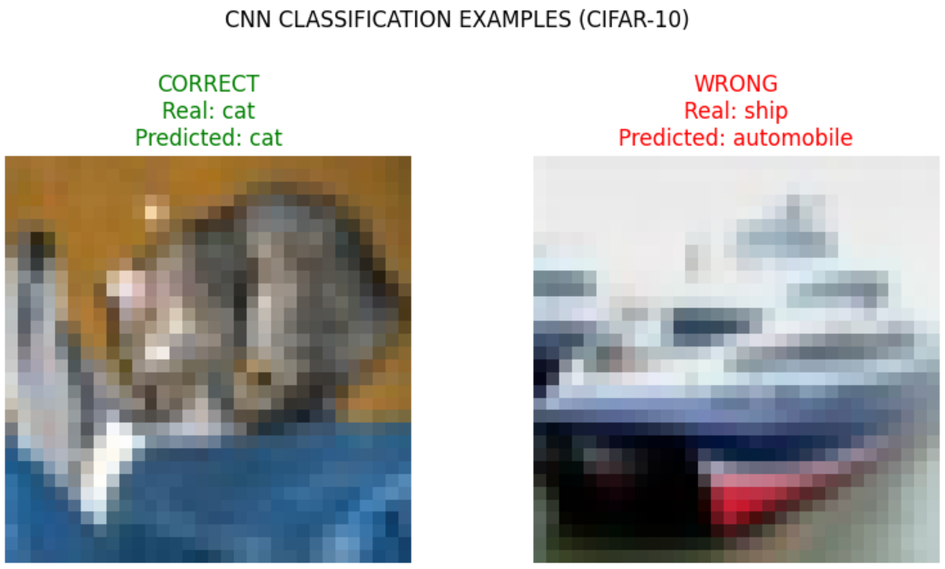
Figure 6: **Training Curves for CNN (64/128): Accuracy and Loss**



(a) CNN (64/128) Accuracy



(b) CNN (64/128) Loss



3 Analysis and Discussion

3.1 CNN vs. MLP vs. KNN vs. NCC

The CNN (70.00% Test Accuracy) achieved the highest performance by far, validating that learned feature extraction through Conv2D layers is essential for complex image classification tasks. The KNN (K=1) (37.38% Test Accuracy) and Nearest Centroid (27.75% Test Accuracy) models performed the poorest, even with PCA applied. This demonstrates that simple distance-based metrics are highly ineffective on this type of image data, as they cannot capture the non-linear, high-level features necessary for accurate classification.

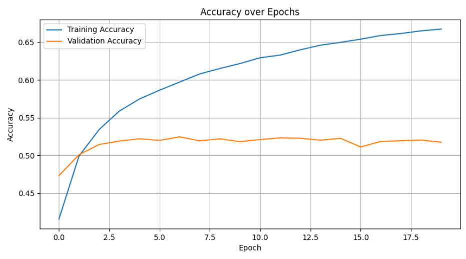
Table 3: **Final Peak Performance Comparison**

Model Type	Pre-processing	Peak Test Accuracy
CNN (64, 128)	None (Image Input)	**70.00%**
MLP (64/32)	PCA (100 Components)	52.88%
KNN (K=1)	PCA	37.38%
KNN (K=3)	PCA	35.34%
Nearest Centroid (NC)	PCA	27.75%

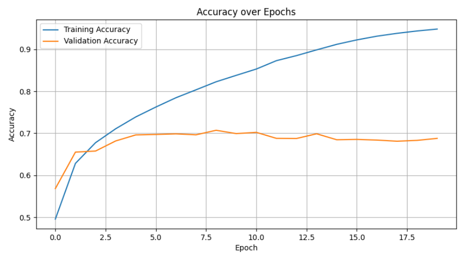
3.2 Overfitting and Epoch Limitation (Both Architectures)

- **MLP Overfitting:** In the MLP experiments, doubling the epochs (from 10 to 20) significantly increased Training Accuracy (from 64.20% to 70.04%) but caused the Test Accuracy to drop (from 51.95% to 51.75%), confirming overfitting.
- **CNN Overfitting:** The CNN displayed the same behavior: training for 20 epochs led to **95.87**% Training Accuracy but a drop in Test Accuracy (68.79%).

Figure 7: **Training Curves to Show Overfitting: Accuracy**



(a) MLP (128/64) Accuracy



(b) CNN (32/64) Accuracy

3.3 Conclusion

The experiments confirmed that for image classification, the **CNN architecture** is necessary for competitive performance. Both FNN types exhibited severe overfitting, underscoring the necessity of implementing **Dropout** regularization and **Early Stopping** in future deep learning models.

3.4 Code Implementation

- **Efficiency:** The codebase utilizes a **Load or Fit** logic, which checks for saved model files (e.g., `mlp_64_32_cifar10_trained_model.keras`) to optimize experimental time.
- **Documentation:** All scripts (`mlp.py`, `cnn.py`) automatically generate visual evidence, including **Accuracy** and **Loss** curves over epochs and display characteristic examples of correct/incorrect classification, fully documenting the network's behavior for the purposes of the report.

3.5 GitHub Repository

The complete source code and necessary configuration files for reproducing the experiments are publicly available on the GitHub repository:

- `https://github.com/lazoulios/neural-network-image-recognition`
- **Contents:** The repository includes the `mlp.py` and `cnn.py` execution scripts, the `utils.py` helper file, the `requirements.txt` dependency list, and the `README.md` setup instructions.