Neural Networks Deep Learning

Semester Assignments for the Course of Neural Networks

Koukouletsou Aikaterini
koukoulet@ece.auth.gr
10218

Aristotle University of Thessaloniki
Faculty of Engineering

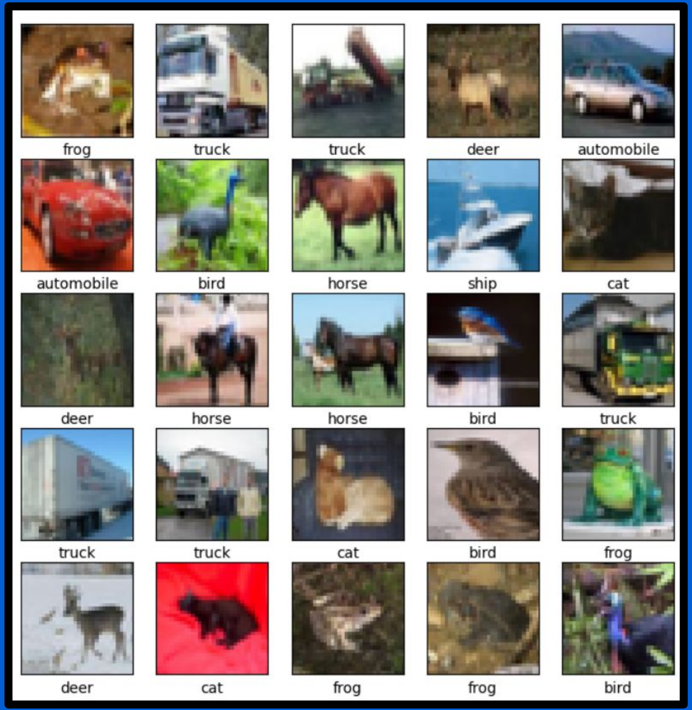
Professor: Anastasios Tefas

JANUARY 2025

TABLE OF CONTENTS

CIFAR-10

The dataset used for this assignment is the CIFAR10 dataset.



Consists of **10 classes**:
airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

Dataset Size:
Training Set **50,000** images
Test Set **10,000** images

Dataset is **Balanced**
6,000 images per Class

Image Dimensions:
32x32 pixels
3 color channels (RGB)

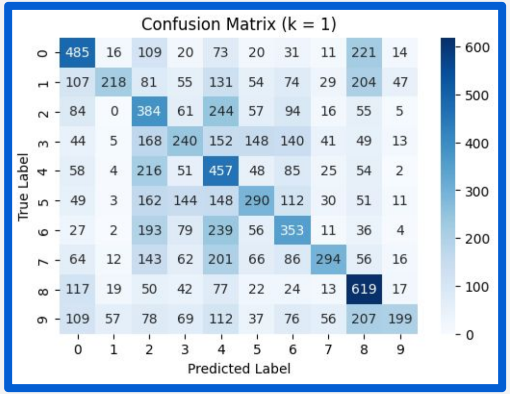
Pixel Values range for **[0, 255]** or **[0, 1]** after **Normalization**

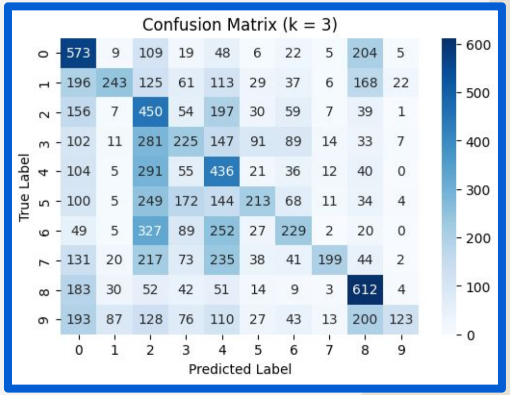
When **flattened** each image is a vector of 32x32x3 = **3072** elements

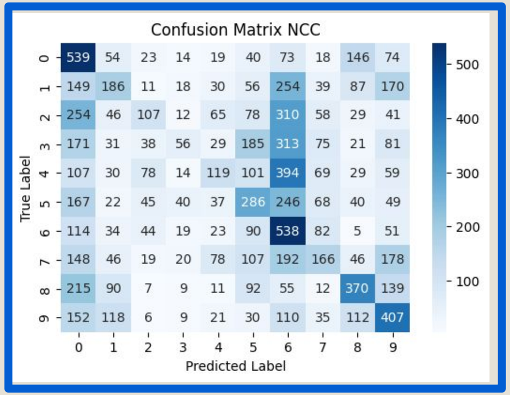
Challenges: Low Resolution

k-Nearest Neighbors + Nearest Centroid

Initial Accuracy Scores:







kNN with k = 1 achieved an accuracy score of 35.39%

kNN with k = 3 achieved an accuracy score of 33.03%

Nearest Centroid achieved an accuracy score of 27.74

k-Nearest Neighbors + Nearest Centroid

Techniques used to for improved Accuracy:

1. **Edge Detection** Algorithms
   a. Sobel
   b. Canny
2. **Color Space** Transformation
   a. HSV
   b. YCbCr
   c. LAB
3. **Distance** Metrics
   a. Cosine Similarity
   b. Minkowsky
   c. Chebyshev
4. Principal Component Analysis (**PCA**)

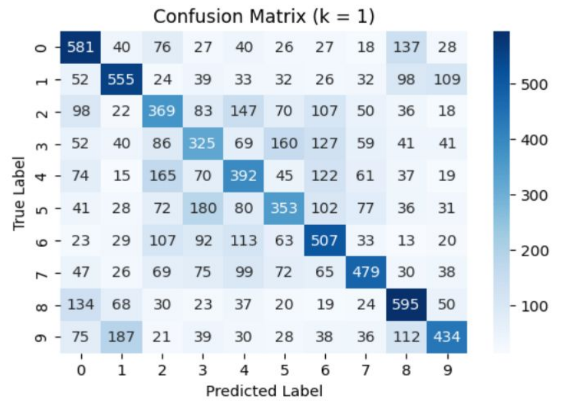
Best Results for LAB + Cosine Similarity + PCA

Method	k-NN (k=1)	k-NN (k=3)	NCC (k=1)
Baseline (Raw Images)	35.39%	33.03%	27.74%
Sobel Edge Detection	32.05%	30.83%	27.56%
Canny Edge Detection	15.89%	14.15%	25.70%
LAB Color Space	38.01%	35.86%	32.86%

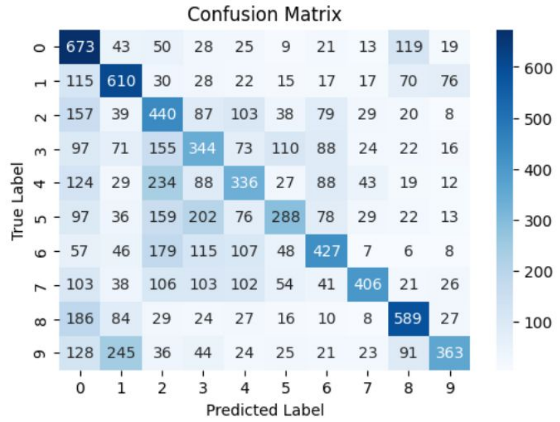
Method	k-NN (k=1)	k-NN (k=3)	NCC (k=1)
PCA + LAB + Cosine Similarity	46.01%	44.76%	32.80%

Metrics for Best Case

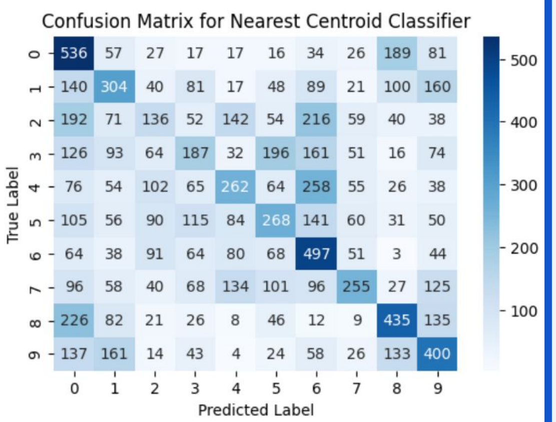
Class	Precision	Recall	F1-Score	Support
Airplane	0.49	0.58	0.53	1000
Automobile	0.55	0.56	0.55	1000
Bird	0.36	0.37	0.37	1000
Cat	0.34	0.33	0.33	1000
Deer	0.38	0.39	0.38	1000
Dog	0.41	0.35	0.38	1000
Frog	0.44	0.51	0.47	1000
Horse	0.55	0.48	0.51	1000
Ship	0.52	0.59	0.56	1000
Truck	0.55	0.43	0.49	1000
Accuracy		0.46		10000
Macro Avg	0.46	0.46	0.46	10000
Weighted Avg	0.46	0.46	0.46	10000



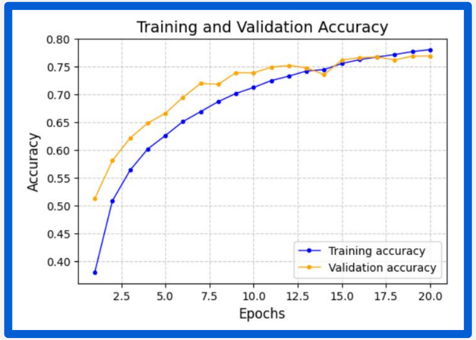
Class	Precision	Recall	F1-Score	Support
Airplane	0.39	0.67	0.49	1000
Automobile	0.49	0.61	0.54	1000
Bird	0.31	0.44	0.36	1000
Cat	0.32	0.34	0.33	1000
Deer	0.38	0.34	0.35	1000
Dog	0.46	0.29	0.35	1000
Frog	0.49	0.43	0.46	1000
Horse	0.68	0.41	0.51	1000
Ship	0.60	0.59	0.60	1000
Truck	0.64	0.36	0.46	1000
Accuracy		0.45		10000
Macro Avg	0.48	0.45	0.45	10000
Weighted Avg	0.48	0.45	0.45	10000

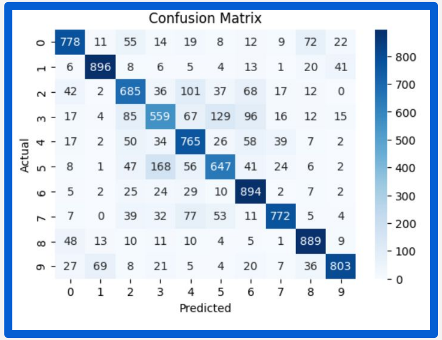


Class	Precision	Recall	F1-Score	Support
Airplane	0.32	0.54	0.40	1000
Automobile	0.31	0.30	0.31	1000
Bird	0.22	0.14	0.17	1000
Cat	0.26	0.19	0.22	1000
Deer	0.34	0.26	0.29	1000
Dog	0.30	0.27	0.28	1000
Frog	0.32	0.50	0.39	1000
Horse	0.42	0.26	0.32	1000
Ship	0.43	0.43	0.43	1000
Truck	0.35	0.40	0.37	1000
Accuracy		0.33		10000
Macro Avg	0.33	0.33	0.32	10000
Weighted Avg	0.33	0.33	0.32	10000



CNN





IMPLEMENTATION APPROACH:

- ➔ A CNN was implemented **from scratch** to gain a deeper understanding of its inner workings and fundamentals.
- ➔ In parallel, **TensorFlow/Keras** was utilized for faster execution. The reduced runtime enabled extensive experimentation, hyperparameter tuning, and the testing of additional techniques to improve accuracy—iterations that would not have been feasible with the scratch implementation alone.

This **dual approach** balanced deep learning and practical efficiency, maximizing both understanding and results.

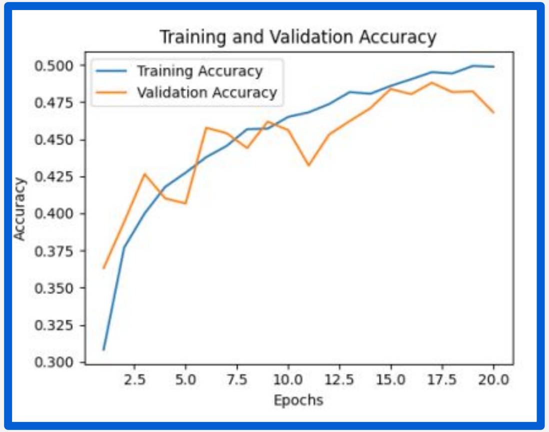
Accuracy: 76.88% - 20 epochs - Time: 643.2s

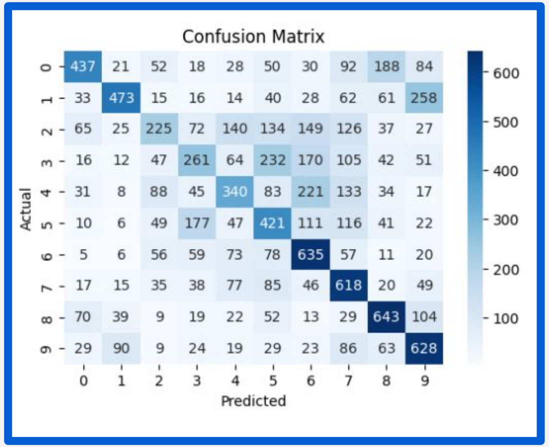
1. Layers:
   1.1. **Three Conv2D layers** (32, 64, 128 filters) with **MaxPooling** and **Dropout.**
   1.2. **Fully connected Dense** layer (1024 units) with Dropout.
   1.3. Output layer: 10 units (**Softmax**).
2. Optimizer: **Adam** with **categorical crossentropy loss**.
3. Features: **Learning rate scheduling** and **data augmentation** to enhance performance.

MLP

Accuracy: 46.81% - 20 epochs - Time: 720.2s

1. Input: 32x32 RGB images, flattened into a 1D vector.
2. Layers:
   - **2.1. Three Dense layers:**
     - 2.1.1. 1024, 2048, and 512 units (**ReLU activation**).
   - 2.2. **Dropout (30%)** added after the second and third Dense layers for regularization.
   - 2.3. Output layer: 10 units (**Softmax**) for classification.
3. Optimizer: **Adam** with **categorical crossentropy loss**.
4. Features: **Data augmentation** (rotation, shifting, flipping, shearing, zooming) to improve generalization.





SVM

OvR Approach

Linear Kernel

Kernel	C parameter	Accuracy
Linear	1.0	0.38

Kernel	C parameter	Accuracy
Linear	10.0	0.36

Polynomial Kernel

Kernel	C parameter	Degree	coef0	Accuracy
Polynomial	1.0	2	1.0	0.52

Kernel	C parameter	Degree	Accuracy
Polynomial	1.0	3	0.52

Kernel	C parameter	Degree	Accuracy
Polynomial	1.0	5	0.50

Kernel	C parameter	Degree	coef0	Accuracy
Polynomial	10.0	2	1.0	0.52

RBF Kernel

Kernel	C parameter	γ parameter	Accuracy
RBF	1.0	0.1	0.22

Kernel	C parameter	γ parameter	Accuracy
RBF	10.0	0.1	0.24

Kernel	C parameter	γ parameter	Accuracy
RBF	1.0	0.01	0.56

Kernel	C parameter	γ parameter	Accuracy
RBF	1.0	scale	0.54

Kernel	C parameter	γ parameter	Accuracy
RBF	10.0	scale	0.57

Sigmoid Kernel

Kernel	C parameter	γ parameter	coef0	Accuracy
Sigmoid	1.0	0.1	0	0.10

Kernel	C parameter	γ parameter	coef0	Accuracy
Sigmoid	10.0	0.1	0	0.10

SVM

Techniques used to for improved Accuracy:

1. **Standard Scaler**
2. **Histogram of Oriented Gradients**

$$\text{Magnitude: } \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}, \text{Orientation: } \theta = \arctan\left(\frac{\partial I}{\partial y} \Big/ \frac{\partial I}{\partial x}\right)$$

3. **PCA**

Best Results for StandardScaler + HOG + PCA

Accuracy:
57% -> 64%

Time: 63min

Confusion Matrix

True Labels \ Predicted Labels	0	1	2	3	4	5	6	7	8	9
0	737	17	60	20	31	10	14	11	80	20
1	28	770	7	22	16	5	17	8	55	72
2	77	7	508	70	108	88	75	32	28	7
3	46	22	94	446	74	162	82	48	9	17
4	43	12	98	74	572	54	64	59	10	14
5	23	9	81	165	70	516	51	61	11	13
6	23	17	47	59	60	45	707	13	20	9
7	18	7	58	64	86	60	15	656	6	30
8	107	55	25	13	13	8	11	13	706	49
9	34	81	13	32	24	13	9	20	35	739

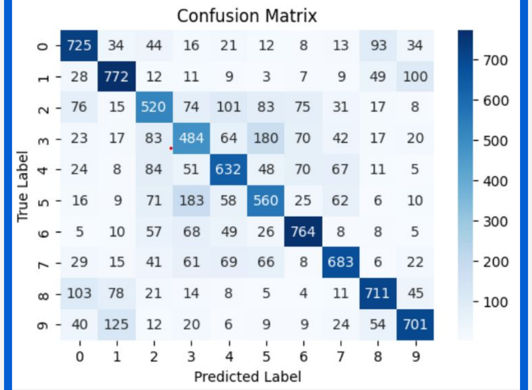
Class	Precision	Recall	F1-Score	Support
Airplane	0.65	0.74	0.69	1000
Automobile	0.77	0.77	0.77	1000
Bird	0.51	0.51	0.51	1000
Cat	0.46	0.45	0.45	1000
Deer	0.54	0.57	0.56	1000
Dog	0.54	0.52	0.53	1000
Frog	0.68	0.71	0.69	1000
Horse	0.71	0.66	0.68	1000
Ship	0.74	0.71	0.72	1000
Truck	0.76	0.74	0.75	1000
Accuracy		0.64		10000
Macro Avg	0.64	0.64	0.64	10000
Weighted Avg	0.64	0.64	0.64	10000

SVM and CNN

Accuracy:
57% -> 65.52%

The **best CNN model** from the first assignment was utilized for **feature extraction**, while the **best SVM model** from this assignment was employed for the **classification** phase after feature extraction.

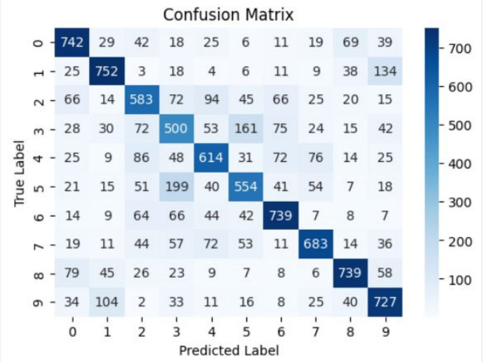
Class	Precision	Recall	F1-Score	Support
Airplane	0.68	0.72	0.70	1000
Automobile	0.71	0.77	0.74	1000
Bird	0.55	0.52	0.53	1000
Cat	0.49	0.48	0.49	1000
Deer	0.62	0.63	0.63	1000
Dog	0.56	0.56	0.56	1000
Frog	0.73	0.76	0.75	1000
Horse	0.72	0.68	0.70	1000
Ship	0.73	0.71	0.72	1000
Truck	0.74	0.70	0.72	1000
Accuracy		0.66		10000
Macro Avg	0.65	0.66	0.65	10000
Weighted Avg	0.65	0.66	0.65	10000



SVM and Efficient Net V2

Accuracy:
57% -> 66.33%

Class	Precision	Recall	F1-Score	Support
Airplane	0.70	0.74	0.72	1000
Automobile	0.74	0.75	0.75	1000
Bird	0.60	0.58	0.59	1000
Cat	0.48	0.50	0.49	1000
Deer	0.64	0.61	0.62	1000
Dog	0.60	0.55	0.58	1000
Frog	0.71	0.74	0.72	1000
Horse	0.74	0.68	0.71	1000
Ship	0.77	0.74	0.75	1000
Truck	0.66	0.73	0.69	1000
Accuracy		0.66		10000
Macro Avg	0.66	0.66	0.66	10000
Weighted Avg	0.66	0.66	0.66	10000



Προσέγγιση	Ακρίβεια (%)
SVM, C=10, γ = scale	57
Με Προεπεξεργαστικά Βήματα (HoG, PCA, Standard Scaler)	64
Με Συνελικτικό Μοντέλο Από την Προηγούμενη Εργασία (Feature Extraction)	65.52
Με EfficientNetV2 για Feature Extraction και SVM για Ταξινόμηση	66.33

RBF

1. **Determining Cluster Centers:** K-Means clustering is applied to group the data and identify the centers of the RBF neurons in the hidden layer.

2. **RBF Kernel Types:** Gaussian, Linear, Quadratic, Inverse Quadratic, Multi Quadratic, Inverse Multiquadric, Matérn 3/2, Matérn 5/2

3. **Adaptive Sigma**

$$\sigma_i = \frac{1}{k}\sum_{j=1}^{k} \|x_j - c_i\|$$

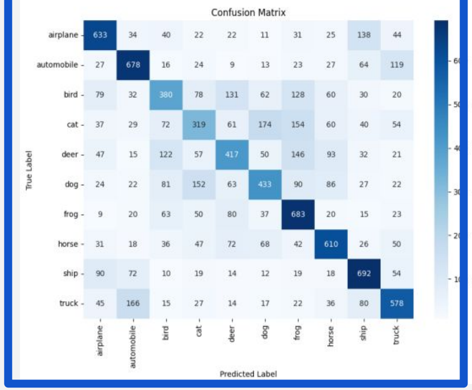
For a constant sigma,

Training Accuracy: 66.98
Testing Accuracy: 54.2

Time: 20 min

n_{hidden}	Training Accuracy (%)	Testing Accuracy (%)
100	45.23	41.50
500	55.12	49.80
1000	60.75	51.40
3000	63.10	53.00
5500	66.98	54.23

Class	Precision	Recall	F1-Score	Support
0	0.62	0.63	0.63	1000
1	0.62	0.68	0.65	1000
2	0.46	0.38	0.41	1000
3	0.40	0.32	0.36	1000
4	0.47	0.42	0.44	1000
5	0.49	0.43	0.46	1000
6	0.51	0.68	0.58	1000
7	0.59	0.61	0.60	1000
8	0.60	0.69	0.65	1000
9	0.59	0.58	0.58	1000
Accuracy		0.54		10000
Macro avg	0.54	0.54	0.54	10000
Weighted avg	0.54	0.54	0.54	10000

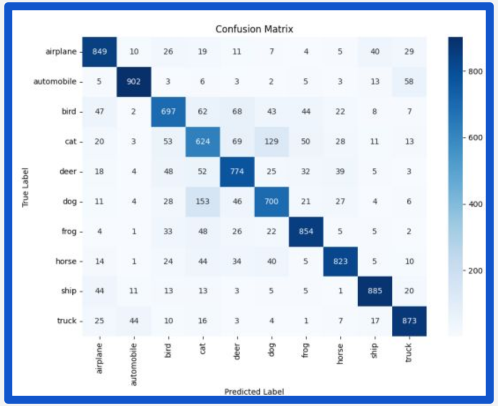


CNN + RBFNN

CNN model from the first Assignment for feature extraction and RBFNN from current Assignment to perform the classification task

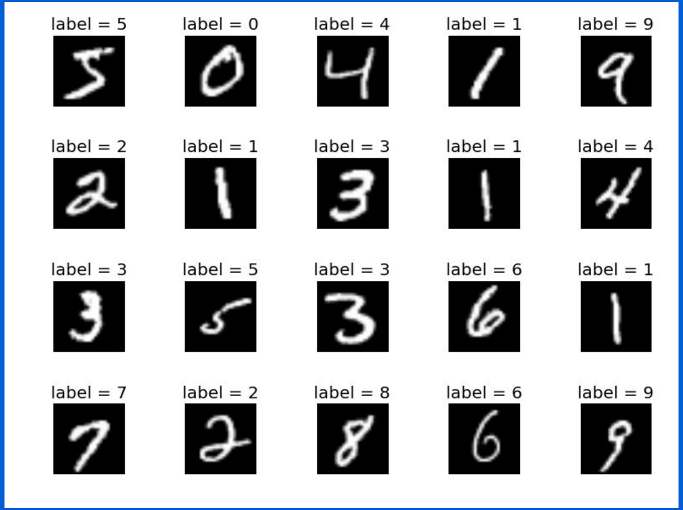
Training Accuracy: 96.31
Test Accuracy: 80.18

Time: 68 min



Class	Precision	Recall	F1-Score	Support
Airplane (0)	0.82	0.85	0.83	1000
Automobile (1)	0.92	0.90	0.91	1000
Bird (2)	0.75	0.70	0.72	1000
Cat (3)	0.60	0.62	0.61	1000
Deer (4)	0.75	0.77	0.76	1000
Dog (5)	0.72	0.70	0.71	1000
Frog (6)	0.84	0.85	0.85	1000
Horse (7)	0.86	0.82	0.84	1000
Ship (8)	0.89	0.89	0.89	1000
Truck (9)	0.86	0.87	0.86	1000
Accuracy		**0.7981**		10000
Macro Avg	0.80	0.80	0.80	10000
Weighted Avg	0.80	0.80	0.80	10000

MNIST



Dataset Split:

Training set: 60,000 images.

Test set: 10,000 images.

Dataset Overview
Comprises 70,000 grayscale images of handwritten digits (0-9). Each image is 28x28 pixels with a single channel (grayscale)

Normalization: Pixel values range from 0 to 255 and should be normalized to [0, 1] for better model performance.

Class Distribution:
Balanced distribution across the 10 digit classes.

AUTOENCODER

IMPLEMENTATION APPROACH:

➔ **Autoencoder**
An initial autoencoder was implemented to perform image reconstruction.

➔ **Autoencoder with Convolutional Layers**
To enhance reconstruction accuracy, a second autoencoder with convolutional layers was developed.

➔ **CNN for Digit Recognition on Reconstructed Image**
The reconstructed images were then utilized for digit recognition using a CNN.

First Model

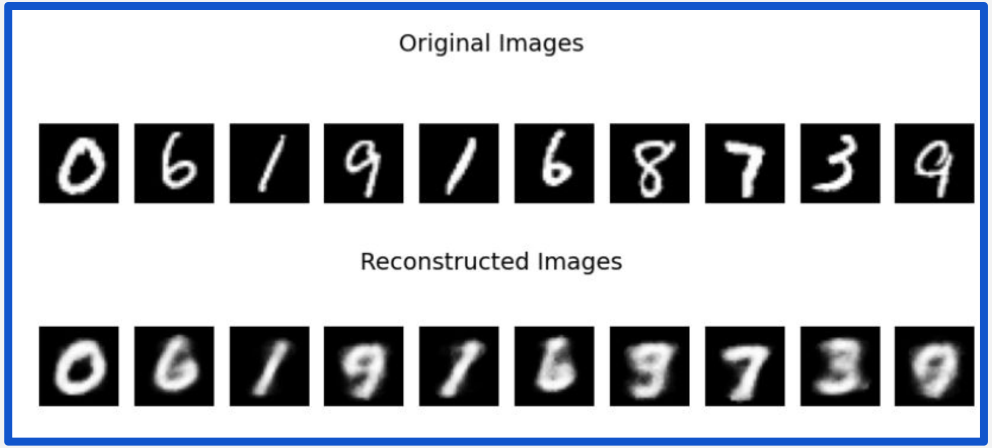
Encoder:

1. The flattened vector passes through three fully connected layers with decreasing dimensions (512, 256, and 128 units), each using the ReLU activation function.
2. The final layer is a dense layer with a latent representation of size 256, also activated by ReLU.

Decoder:

1. The decoder takes the latent representation as input and reconstructs the image by passing it through three fully connected layers (128, 256, and 512 units) with ReLU activation.
2. The final layer outputs a reshaped image (28, 28, 1) using a sigmoid activation, ensuring pixel values are in the range [0, 1].

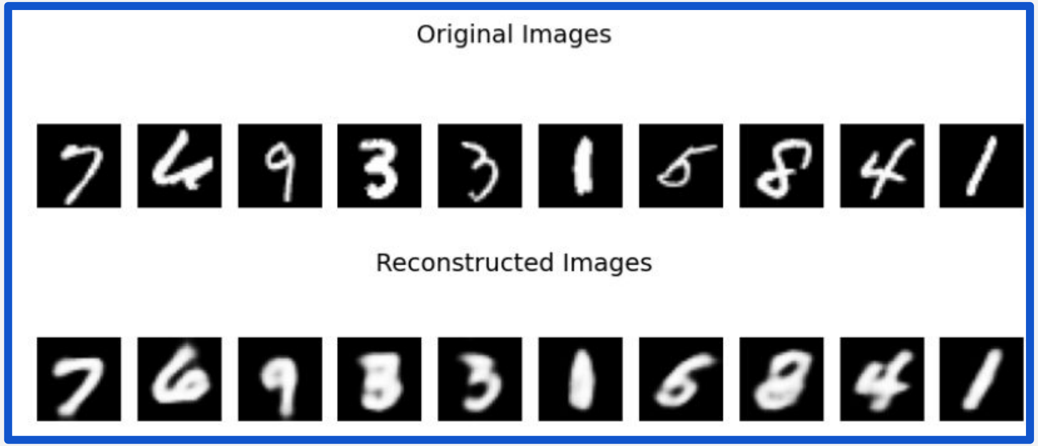
The autoencoder is trained to minimize the mean squared error (MSE) loss with additional metrics for Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) to evaluate image quality. Data augmentation is applied during training to improve generalization.



Second Model

- **Encoder**: Convolutional layers (32, 64, and 128 filters) with max-pooling reduce the image dimensions. The output is flattened and passed through a dense layer to form a latent space of size 512.
- **Decoder**: The latent vector is expanded and reshaped, followed by transposed convolution layers (128 and 64 filters) and upsampling to reconstruct the original image (28x28x1).

The custom loss function combines **MSE**, **SSIM**, and **L1 loss** to optimize both pixel-wise and perceptual quality. Data augmentation is used during training.

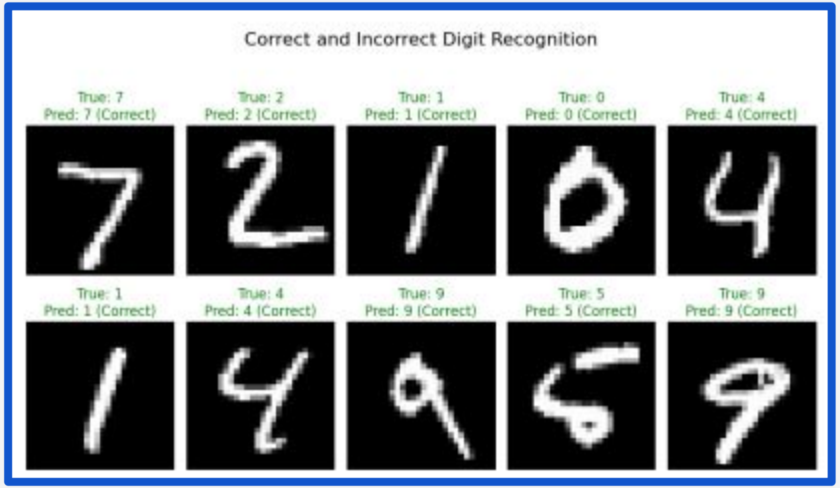


Digit Recognition

The **CNN for Digit Recognition** consists of:

- **Two convolutional layers** with ReLU activation (32 and 64 filters) followed by max-pooling.
- **Flattening** layer to convert the feature map into a 1D vector.
- **Dense layers** (128 units) and a final **softmax layer** with 10 units for classification.

The performance rates for digit recognition on the original and reconstructed datasets are **99.40%** and **82.62%**, respectively. The training times are **20 minutes** for the first autoencoder model and **50 minutes** for the second autoencoder model and **20 minutes** for the digit recognition part



Thank You!

Koukouletsou Aikaterini 10218