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

1

k - Nearest Neighbors + Nearest Centroid Classifiers (kNN + NCC) 4

Radial Basis Function Neural Networks (RBFNN)

2

Convolutional Neural Network + **Multi Layer Perceptron** (CNN + MLP)

5

Autoencoders

3

Support Vector Machines

(SVM)

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



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

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

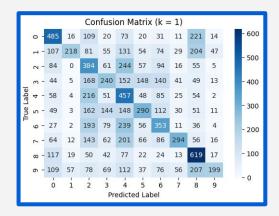
Image Dimensions:

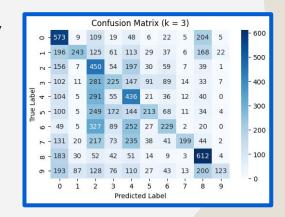
32x32 pixels 3 color channels (RGB)

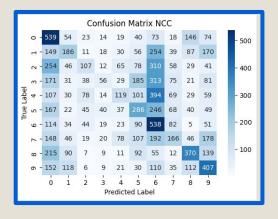
Challenges: Low Resolution

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**)

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%

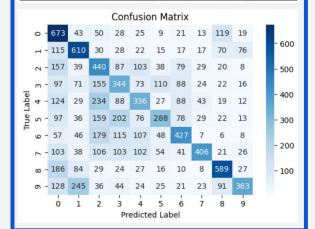


Metrics for Best Case

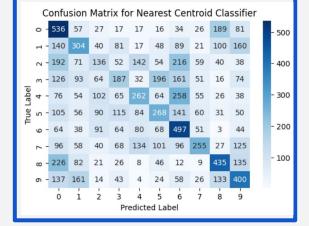
Class	Precision	Recall	F1-Score	Support
Airplane	0.49	0.58	0.53	1000
Automobile	0.55	0.56	0.55	1000
Bird	0.36	0.37	0.37	1000
Cat	0.34	0.33	0.33	1000
Deer	0.38	0.39	0.38	1000
Dog	0.41	0.35	0.38	1000
Frog	0.44	0.51	0.47	1000
Horse	0.55	0.48	0.51	1000
Ship	0.52	0.59	0.56	1000
Truck	0.55	0.43	0.49	1000
Accuracy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

	Confusion Matrix (k = 1)											
	0 -	581	40	76	27	40	26	27	18	137	28	
	п-	52	555	24	39	33	32	26	32	98	109	- 500
	- 2	98	22	369	83	147	70	107	50	36	18	
_	m -	52	40	86		69	160	127	59	41	41	- 400
True Label	4 -	74	15	165	70	392	45	122	61	37	19	200
rue	ი -	41	28	72	180	80		102	77	36	31	- 300
_	9 -	23	29	107	92	113	63	507	33	13	20	- 200
	۲ -	47	26	69	75	99	72	65	479	30	38	
	∞ -	134	68	30	23	37	20	19	24	595	50	- 100
	ი -	75	187	21	39	30	28	38	36	112	434	
	0 1 2 3 4 5 6 7 8 9 Predicted Label											

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

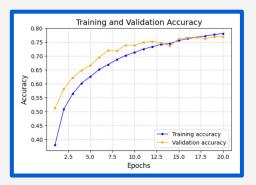


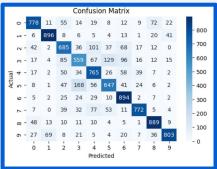
CNN

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.





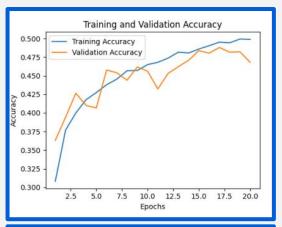
Accuracy: 76.88% - 20 epochs - Time: 643.2s

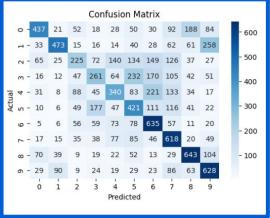
- 1. Layers:
 - 1.1. **Three Conv2D layers** (32, 64, 128 filters) with **MaxPooling** and **Dropout.**
 - 1.2. **Fully connected Dense** layer (1024 units) with Dropout.
 - 1.3. Output layer: 10 units (**Softmax**).
- Optimizer: Adam with categorical crossentropy loss.
- 3. Features: **Learning rate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





SVIM OvR Approach

Linear Kernel

Kernel	C parameter	Accuracy
Linear	1.0	0.38
	7	7
	(
Kernel	C parameter	Accuracy

Polynomial Kernel

Kernel	C parameter	Deg	Degree		Accuracy				
Polynomial	1.0	2		1.0	0.52				
Kernel	C parame	C parameter		gree	Accuracy				
Polynomia	1.0	1.0		3	0.52				
Kernel	C parame	C parameter		gree	Accuracy				
Polynomia	1.0			5	0.50				
Kernel	C parameter	Deg	gree	coef	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

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

SVM

Techniques used to for improved Accuracy:

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

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

3. PCA

Best Results for StandardScaler + HOG + PCA

True: cat (predicted: cat)

True: ship (predicted: ship)

True: airplane (predicted: airplane)

True: frog (predicted: frog)

True: ship (predicted: sutemobile)

True: frog (predicted: bird)

True: frog (predicted: horse)

True: frog (predicted: horse)

True: frog (predicted: horse)

True: sutemobile (predicted: ship)

True: automobile (predicted: frog)

Accuracy: 57% -> 64%

Time: 63min

					Co	nfusio	n Mat	trix			
	0 -	737	17	60	20	31	10	14	11	80	20
	ч-	28	770	7	22	16	5	17	8	55	72
	- 2	77	7	508	70	108	88	75	32	28	7
s	m -	46	22	94	446	74	162	82	48	9	17
abel	4 -	43	12	98	74	572	54	64	59	10	14
True Labels	ω -	23	9	81	165	70	516	51	61	11	13
Ė	9 -	23	17	47	59	60	45	707	13	20	9
	۲ -	18	7	58	64	86	60	15	656	6	30
	∞ -	107	55	25	13	13	8	11	13	706	49
	ი -	34	81	13	32	24	13	9	20	35	739
		Ó	i	2	3 P	4 redicte	5 d Labe	6	7	8	9
						Carete	G Labe				

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.

True: ship Pred: ship	True: airplane Pred: airplane	True: frog Pred: frog	True: frog Pred: frog	True: automobile Pred: automobile
-	Annual.	CA	-	-
-	Do	10 1	No.	ALL OV
True: cat Pred: dog	Yrue: ship Pred: automobile	Yrue: ship Pred: frog	True: horse Pred: truck	True: deer Pred: airplane
I. OF	1.0%	March .	S O I	· white
Ser Call		a.E.	1	8
				1000

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

					Con	fusio	n Ma	atrix				
	0 -	725	34	44	16	21	12	8	13	93	34	- 700
	н-	28	772	12	11	9	3	7	9	49	100	- 700
	- 2	76	15	520	74	101	83	75	31	17	8	- 600
L	m -	23	17	83	484	64	180	70	42	17	20	- 500
True Label	4 -	24	8	84	51	632	48	70	67	11	5	- 400
rue	ი -	16	9	71	183	58	560	25	62	6	10	
-	9 -	5	10	57	68	49	26	764	8	8	5	- 300
	۲-	29	15	41	61	69	66	8	683	6	22	- 200
	ω -	103	78	21	14	8	5	4	11	711	45	- 100
	ი -	40	125	12	20	6	9	9	24	54	701	
		ó	í	2	3	4	5	6	7	8	9	
					Pr	edicte	ed Lat	oel				

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

						Con	fusio	n Ma	atrix					
	0		742	29	42	18	25	6	11	19	69	39	-	700
	н.	-	25	752	3	18	4	6	11	9	38	134		
	7	-	66	14	583	72	94	45	66	25	20	15		600
	m ·	-	28	30	72	500	53	161	75	24	15	42	-	500
abe	4	-	25	9	86	48	614	31	72	76	14	25	-	400
True Labe	2	-	21	15	51	199	40	554	41	54	7	18		200
-	9	-	14	9	64	66	44	42	739	7	8	7		300
	7	-	19	11	44	57	72	53	11	683	14	36	-	200
	ω.	-	79	45	26	23	9	7	8	6	739	58	-	100
	6		34	104	2	33	11	16	8	25	40	727		
			Ó	i	2	3 Pr	4 edicte	5 ed Lal	6 bel	7	8	9		

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

automobile - 27 670 16 24 9 13 23 27 64 119 bird - 79 32 350 78 131 62 128 60 30 20 cat - 37 29 72 339 61 134 154 60 40 54 66 67 67 68 68 68 68 68 68 68 68 68 68 68 68 68		airplane	automobile	ā	٥		ਚ ed Label	¥	horse	-61	truck	
Begin bird - 27 678 16 24 9 13 23 27 64 119 bird - 79 32 350 78 131 62 128 60 30 20 cat - 37 29 72 319 61 174 154 60 40 54 deer - 47 15 122 57 417 50 146 93 32 21 deg - 24 22 81 152 63 413 90 86 27 22 freg - 9 20 63 50 80 37 683 20 15 23 horse - 31 18 36 47 72 68 42 610 26 50 ship - 90 77 10 19 14 12 19 18 692 54		- 90	- 0		- te	ė.	- 6	- 6	- 9	g.	×	
Beg deer 47 15 122 57 417 50 146 9 3 22 27 64 119 bird - 79 32 50 78 131 62 128 60 30 20 cat - 37 29 72 339 61 174 154 60 40 54 deer - 47 15 122 57 417 50 146 93 32 21 deep - 24 22 81 152 63 433 90 86 27 22 freq - 9 20 63 50 80 37 633 20 15 23 horse - 31 18 36 47 72 68 42 610 26 50	truck -	45	166	15	27	14	17	22	36	80	578	
automobile - 27 678 16 24 9 13 23 27 64 119 bird - 79 32 550 78 131 62 128 60 30 20 cat - 37 29 72 319 61 174 154 60 40 54 g deer - 47 15 122 57 417 50 146 93 32 21 beg deg - 24 22 81 152 63 433 90 86 27 22 frog - 9 20 63 50 80 37 653 20 15 23	ship -	90	72	10	19	14	12	19	18	692	54	
automobile - 27 678 16 24 9 13 23 27 64 119 bird - 79 32 350 78 131 62 128 60 30 20 6at - 37 29 72 319 61 174 154 60 40 54 g deer - 47 15 122 57 417 50 146 93 32 21 deg - 24 22 81 152 63 433 90 86 27 22	horse -	31	18	36	47	72	68	42	610	26	50	
automobile - 27 678 16 24 9 13 23 27 64 119 bird - 79 32 350 78 131 62 128 60 30 20 6at - 37 29 72 3199 61 174 154 60 40 54	frog -	9	20	63	50	80	37	683	20	15	23	١
automobile - 27 678 16 24 9 13 23 27 64 119 bird - 79 32 350 78 131 62 128 60 30 20 6at - 37 29 72 3199 61 174 154 60 40 54	e dog -	24	22	81	152	63		90	86	27	22	١
automobile - 27 678 16 24 9 13 23 27 64 119 bird - 79 32 340 78 131 62 128 60 30 20	deer -	47	15	122	57	417	50	146	93	32	21	۱
automobile - 27 678 16 24 9 13 23 27 64 119	cat -	37	29	72	319	61	174	154	60	40	54	۱
	bird -	79	32	380	78	131	62	128	60	30	20	
airplane - 633 34 40 22 22 11 31 25 138 44	automobile -	27	678	16	24	9	13	23	27	64	119	١
	airplane -		34	40	22	22	11	31	25	138	44	

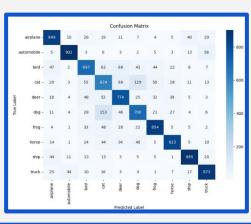
CNN + RBFNN

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

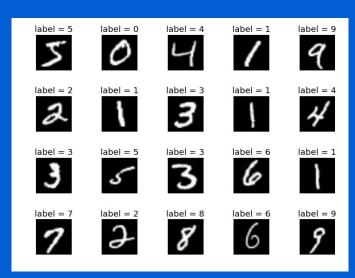
Time: 68 min





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

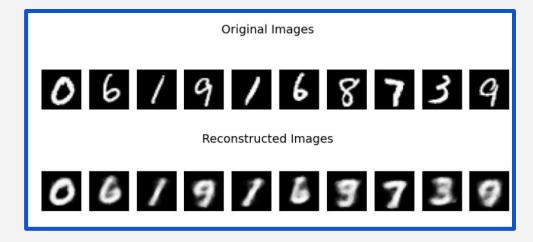
Encoder:

- 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].

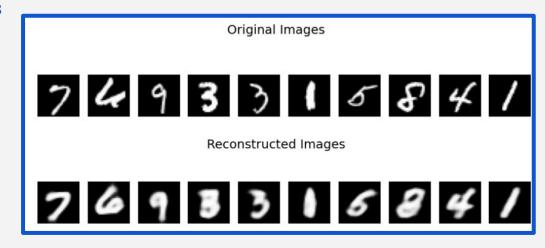
The autoencoder is trained to minimize the mean squared error (MSE) loss with additional metrics for Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) to evaluate image quality. Data augmentation is applied during training to improve generalization.



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.

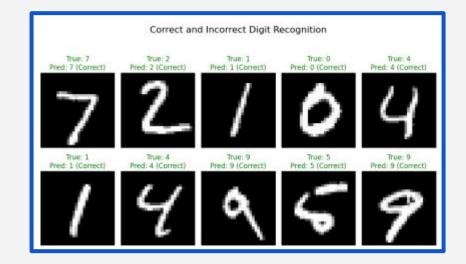


Digit Recognition

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