



## **Department of Computer Science**

**Faculty Member:** <>

**Date:** <>

**Semester:** <>

### **Computer Vision**

## **Lab 8: Introduction to Deep Learning with TensorFlow – Implementing a Simple CNN on MNIST**

### **Introduction**

In this laboratory exercise, you will get started with deep learning using TensorFlow, one of the most popular frameworks for building and training neural networks. You will begin by exploring TensorFlow's official beginner guide, understanding how a simple neural network (Multi-Layer Perceptron) is built and trained. Then, you will extend that example by implementing a Convolutional Neural Network (CNN) to solve the same MNIST handwritten digit classification problem. The goal of this lab is to help you understand how deep learning models are structured, how they learn from data, and why CNNs are more effective for image-based tasks than traditional fully connected networks.

### **Objectives**

- Get familiar with TensorFlow and Keras environments.
- Learn how to set up a virtual environment and install TensorFlow.



- Understand the basic components of a deep learning model: layers, activation functions, optimizer, and loss.
- Implement and train a simple CNN to classify handwritten digits from the MNIST dataset.
- Compare CNNs with simple dense (MLP) architectures and discuss why CNNs are preferred for visual data.

## Lab Task 1 – Setting Up TensorFlow Environment

## Lab Task 2 – Exploring TensorFlow's Beginner Example

1. Open the TensorFlow beginner tutorial:  
<https://www.tensorflow.org/tutorials/quickstart/beginner>
2. Run the example to train a simple dense (fully connected) neural network on MNIST.
3. Observe the dataset loading, model definition, compilation, training, and evaluation steps.
4. Note the accuracy achieved using the simple MLP model.

## Lab Task 3 – Building a Simple CNN for MNIST

1. Load and preprocess the MNIST dataset (normalize and reshape images).
- Load the MNIST dataset using
  - Normalize the image pixel values to the range **[0, 1]** by dividing by 255.
  - Reshape the data to include a single grayscale channel → from (28, 28) to (28, 28, 1).
  - Create a sequential CNN model with the following layers:
    - **Convolutional Layer 1:**
      - 32 filters of size (3×3)
      - Activation: ReLU
      - Input shape: (28, 28, 1)
    - **MaxPooling Layer 1:**
      - Pool size: (2×2) to reduce spatial dimensions.
    - **Convolutional Layer 2:**
      - 64 filters of size (3×3)
      - Activation: ReLU
    - **MaxPooling Layer 2:**
      - Pool size: (2×2)
    - **Flatten Layer:**
      - Converts 2D feature maps into a 1D feature vector for dense layers.
    - **Fully Connected (Dense) Layer:**
      - 64 neurons



- Activation: ReLU
- o **Output Layer:**
  - 10 neurons (representing digits 0–9)
  - Activation: Softmax (to output class probabilities).
- o **Evaluate and Visualize Results**

## Lab Task 4 – Building a Simple CNN for MNIST

1. Load the custom dataset using TensorFlow's `image_dataset_from_directory()` function.
2. Use the `train` directory for training images and the `test` directory for testing images.
3. Set appropriate image dimensions and a batch size (e.g., 32).
4. Normalize the pixel values to the range  $[0, 1]$  for faster training convergence.
5. Use the **CNN architecture** defined in **Lab Task 3**.