## **Practical Assessment 5: Deep Learning**

### **Objective:**

Build, train, evaluate, and deploy a deep learning model for classification or regression using TensorFlow/Keras or PyTorch. (choose any one framework)

## **Task 1: Dataset Selection and Preprocessing**

**Instructions:**

* Choose a dataset suitable for deep learning (image or structured data).
  + **Recommended for classification**: MNIST, Fashion-MNIST, CIFAR-10, IMDB Sentiment.
  + **Recommended for regression**: Boston Housing, California Housing, or custom CSV.
* Load and preprocess the data:
  + Normalize numerical inputs (MinMaxScaler / divide by 255 for images).
  + Encode categorical labels if needed (to\_categorical for classification).
  + Train/test split using train\_test\_split().

**Deliverables:**

* Data loading and preprocessing code.
* Summary of dataset shape and class distribution.

## **Task 2: Build a Deep Learning Model**

**Instructions:**

* Use **Keras Sequential API** or **PyTorch nn.Module**.
* For **classification**:
  + Use multiple Dense layers with ReLU, final layer with softmax.
* For **regression**:
  + Output a single neuron with linear activation.
* Include:
  + Dropout or BatchNorm
  + Proper activation functions

**Deliverables:**

* Model architecture code.
* Model summary (model.summary() in Keras).

## **Task 3: Model Compilation and Training**

**Instructions:**

* Compile the model with:
  + **Loss**: categorical\_crossentropy / binary\_crossentropy / mse
  + **Optimizer**: Adam or SGD
  + **Metrics**: accuracy / MSE / MAE
* Train the model:
  + Use model.fit() with validation split.
  + Plot training & validation loss and accuracy.

**Deliverables:**

* Training code and loss/accuracy curves.
* Optional: Save plots as .png.

## **Task 4: Model Evaluation and Metrics**

**Instructions:**

* Evaluate on test set.
* For classification:
  + Show confusion matrix, classification report (precision, recall, F1).
  + ROC-AUC (if binary).
* For regression:
  + Show MSE, RMSE, MAE, and R².

**Deliverables:**

* Evaluation metrics and visualizations.
* Confusion matrix (for classification).

## **Task 5: Hyperparameter Tuning**

**Instructions:**

* Tune key hyperparameters:
  + Learning rate
  + Number of layers/neurons
  + Dropout rate
  + Batch size
* Use either:
  + Manual tuning
  + Keras Tuner (Optional but recommended)

**Deliverables:**

* Experiment results with different configurations.
* Final tuned model performance.

## **Task 6: Model Saving and Loading**

**Instructions:**

* Save the trained model:
  + Keras: model.save("model.h5")
  + PyTorch: torch.save()
* Load the model and verify predictions still work.

**Deliverables:**

* Code to save and load model.
* Example prediction using the loaded model.

## **Task 7: Simple API with Flask or FastAPI**

**Instructions:**

* Create a lightweight API to serve predictions.
* Input data: JSON payload with test sample.
* Output: Model prediction.
* Test the API with sample inputs using curl or Postman.

**Deliverables:**

* app.py (Flask or FastAPI script)
* Screenshot of API running and returning a prediction

## **Task 8: GitHub Repository and Documentation**

**Instructions:**

* Upload all project files to GitHub.
* Include a README.md with:
  + Project title and overview
  + Dataset used
  + Steps performed
  + Instructions to run notebook and API
  + Screenshots/visualizations

**Deliverables:**

* GitHub repository link
* Clear and informative README