**Project Problem Statement:**

You are required to design and implement Neural Network models for both a **regression** and a **classification** task using different deep learning frameworks and approaches. You will compare performance across these tasks using:

1. **PyTorch ANN (from scratch)**
   * Implement a **fully-connected Artificial Neural Network** (ANN) in PyTorch **from scratch**, i.e., directly using PyTorch tensors and writing custom training loops (rather than using high-level trainer APIs).
   * Address **both** a regression problem **and** a classification problem with this PyTorch ANN.
   * Example readily available datasets:
     + **Regression**: California Housing dataset (via scikit-learn) or a simple regression dataset from the UCI Machine Learning Repository.
     + **Classification**: Available in torchvision.datasets: MNIST, FashionMNIST, CIFAR-10 or CIFAR-100 (special challenge).
2. **Keras CNN (Classification Only)**
   * Create a **Convolutional Neural Network** (CNN) model in Keras (TensorFlow backend).
   * Focus **only** on a classification task, using image-based data.
   * Example readily available dataset:
     + **Classification**: The built-in MNIST, CIFAR-10 or CIFAR-100 datasets in Keras (tf.keras.datasets.mnist, tf.keras.datasets.cifar10, tf.keras.datasets.cifar100).
3. **Comparative Analysis**
   * **Architecture & Hyperparameters**: Document your network architectures (layers, activations, etc.) and key hyperparameters (learning rate, batch size, number of epochs).
   * **Training & Validation**: Train each model on the chosen datasets, reporting both training and validation performance.
   * **Evaluation Metrics**:
     + For **classification**: Accuracy, confusion matrix, and possibly precision, recall, and F1-score.
     + For **regression**: Mean Squared Error (MSE), Mean Absolute Error (MAE), and/or R².
   * **Learning Curves**: Plot training vs. validation loss (and accuracy for classification) over epochs.
   * **Computation Time & Resource Usage**: Optionally track approximate training time or GPU usage if relevant.
4. **Presentation of Results**  
   Your final submission should include:
   * A concise **presentation slides** containing:
     1. **Dataset Description**: Overview of each dataset’s size, features, classes (if applicable), and train/test split.
     2. **Model Details**: Summaries of the PyTorch ANN architecture(s) (for regression and classification) and Keras CNN architecture (for classification only).
     3. **Training Configurations**: Information on learning rate, batch size, number of epochs, etc.
     4. **Performance Metrics**: Numerical results for each task and model.
     5. **Graphs**:
        1. Learning curves (training/validation loss and accuracy).
        2. Confusion matrix for classification tasks.
     6. **Comparative Table**: Provide a clear side-by-side comparison of results. An example table might look like:

| **Model** | **Dataset / Task** | **Key Hyperparams** | **Final Metric** | **Training Time** |
| --- | --- | --- | --- | --- |
| PyTorch ANN (Reg) | California Housing | LR=0.01, Epoch=50 | MSE = 22.5; MAE = 3.4 | ~5 min |
| PyTorch ANN (Class) | CIFAR-10 | LR=0.001, Epoch=30 | Accuracy = 82.4% | ~10 min |
| Keras CNN (Class) | CIFAR-10 | LR=0.0001, Epoch=20 | Accuracy = 88.7% | ~8 min |

* + 1. **Discussion**: Briefly discuss the strengths and weaknesses of each approach, and explain notable findings or patterns in the results.

1. **Oral Examination (Viva)**
   * **Code Explanation**: You should be able to explain your custom PyTorch training loops, how you handle forward passes, backpropagation, and weight updates. You should also be able to clarify how you built and compiled your Keras CNN, including layer configurations and compilation parameters.
   * **Theoretical Understanding**: Show a firm grasp of the difference between fully-connected ANNs and CNNs, and how convolutional layers learn to detect features.