Course Project

Neural Networks for Classification

Neural networks are powerful tools for classification tasks. In this course project, students will explore, experiment with different neural network libraries and frameworks for classification tasks. The focus will be on implementing across various libraries and frameworks as following:

1. **Scikit-learn (MLPClassifier)**
2. **TensorFlow/Keras**
3. **PyTorch**
4. **Additional libraries or frameworks (Bonus score)**

Students are required to conduct a detailed comparison of the above libraries and frameworks based on the following criteria:

1. **Training Time**
2. **GPU Memory Usage**
3. **Pros and Cons** (derived from your research)
4. **Ease of use** (based on your personal experience throughout the task)

By completing this project, students will gain experience with multiple libraries and frameworks for designing and training neural networks, as well as insights into the strengths and limitations of these tools for future classification tasks.

1 Dataset

For this project, students will use the *CIFAR-10*, a widely dataset for image classification tasks. The objective is to classify images into their corresponding classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck).

The dataset contains 60,000 32×32 RGB images (50,000 for training and 10,000 for testing).

Visit: `http://www.cs.toronto.edu/~kriz/cifar.html`

2 Specification

Building a classifier model involves several stages. Regardless of the libraries or frameworks used, the basic procedure is consistent and follows these steps:

2.1 Data Preparation

The first step in building any classifier model is preparing the data:

- **Data collection**: Gather the data from a dataset.
- **Data preprocessing**: This includes removing missing data, encoding categorical features, normalizing the data, and splitting the data into training, validation, and test sets.
- **Feature engineering**: Features may need to be engineered to improve performance.

Notes: In this project, students are required to (1) normalize and (2) flatten the image data; then (3) apply one-hot encoding to the output labels.

2.2 Model Design

Designing the model is defining the architecture of the classifier. For example, in case of MLP:

- **Number of layers**: Decide on how many hidden layers the MLP will have.
- **Number of neurons per layer**: Decide on the number of neurons in each layer.
- **Activation functions**: Sigmoid, ReLU and Softmax (for multi-class classification).
- **Dropout or batch normalization**: If needed, incorporate dropout or batch normalization to prevent overfitting and to speed up training phase.

To build a MLP classifier, the following libraries and frameworks can be considered:

- **Scikit-learn (MLPClassifier)**.
- **TensorFlow/Keras**.
- **PyTorch**.
- **Other libraries suggested by students (Bonus score)**.

Notes: In this project, students need to use the same architecture when experimenting and evaluating across the libraries and frameworks to ensure consistency and fair comparison.

2.3 Selecting Loss Function and Optimizer

The next step is selecting the appropriate loss function and optimizer:

- **Loss function**: For classification tasks, the common loss functions are cross-entropy loss (for multi-class problems) and binary cross-entropy loss (for binary classification).
- **Optimizer**: such as SGD, Adam, etc. Adjust the learning rate to control the step size.

2.4 Model Training

Once the model architecture, loss function, and optimizer are defined, proceed with training model:

- **Epochs**: Specify the number of epochs (iterations over the entire training dataset).
- **Batch size**: Choose the batch size, which defines the number of samples per gradient update.
- **Training process**: For each batch, perform forward propagation, compute the loss, and update the model weights using backpropagation.
- **Early stopping**: Monitor the validation set performance during training process and stop if it doesn't improve after a set number of epochs.

Notes: In this project, students need to record the training time and the GPU consumption of libraries and frameworks during the training process for later comparison.

2.5 Model Evaluation

After training, it is important to evaluate the performance of the model on unseen data (test set):

- **Evaluation metrics**: Accuracy, precision, recall, F1-score, etc., depending on the problem.
- **Confusion matrix**: used for analyzing how well the model performs on each class.

2.6 Further Usage

If the model meets the required quality standards, it can be prepared for practical applications by (1) saving and loading model, and (2) deploy the model.

Notes: In this project, students are NOT required to deploy the model as a real-life product; however, you are encouraged to explore it on your own to enhance your skills.

3 Submission

Your report and source code must be contributed in the form of a compressed file (.zip, .rar, .7z) and named according to the format **StudentID.zip(.rar/.7z)**. If your submission is really large, it can be uploaded to the Google Drive and then submit a text file that contains the shared link (the "last updated" field must be before the deadline).

3.1 Source code (Jupyter Notebooks)

The source code, results will be reported in Jupyter Notebooks with the following requirements:

- Student information (Student ID, full name, etc.).
- Detailed explanation of each step. Illustrative images, diagrams and equations are required.
- Each processing step must be fully commented, and results should be printed for observation.
- The notebooks need to be well-formatted.
- Before submitting, re-run the notebook (`Kernel` → `Restart & Run All`).

3.2 Report

Students are required to prepare a report analyzing and comparing the libraries and frameworks. The report must adhere to the following structure:

- Student information (Student ID, full name, etc.).
- Self-evaluation of the assignment requirements.
- Analysis and comparison of the libraries and frameworks based on the following criteria:
  - **Training Time**: Chart, analysis and your comments on training time.
  - **GPU Memory Usage**: Chart, analysis and your comments on GPU consumption.
  - **Pros and Cons**: The advantages and disadvantages of each library and framework derived from your research.
  - **Ease of Use**: Reflect your experience while working with them throughout the task.
- The report needs to be well-formatted and exported to PDF. Ensure no figures or tables are improperly formatted (e.g., cut off by page breaks), as this will result in point deductions.
- References (if any), formatted appropriately in the bibliography section.

4 Assessment

No.	Details	Score
1	Scikit-learn (`MLPClassifier`).	30%
2	TensorFlow/Keras.	30%
3	PyTorch.	30%
4	Report	10%
5	Bonus: Use additional libraries or frameworks for classification.	20%
	Total	**120%**

5 Notices

Please pay attention to the following notices:

- This is an **SOLO** or **DUAL** assignment.

  For **DUAL** submissions, rename the file to **StudentID1_StudentID2.zip**.
- Duration: about 2 weeks.
- Any plagiarism, any tricks, or any lie will have a 0 point for the course grade.

The end.