# CS 203: Software Tools & Techniques for AI

## IIT Gandhinagar

## Sem-II - 2024-25

---

## LAB 07

## Lead TA: Himanshu Beniwal

**Total marks: 100**
**Submission deadline: Monday, 15/03/2025 11:59 PM**
**Submission guidelines:**

1. Code should be added to a GitHub repository, and the **repository details should be shared in the pdf.**
2. **Submit the PDF showing screenshots** of all steps involved in the following code.

**Note:** Submitting this assignment solution confirms that you will follow the IITGN's honor code. We shall strictly penalize the submissions containing plagiarized text/code.

---

**Objective:**
This assignment aims to learn about text classification tasks for checkpoint creation.

---

**1. Dataset Preparation (10%)**

- Load the training dataset and test data (Dataset 1). [UPDATED DATASET]
- Also, the IMDB dataset (Dataset 2) can be used for continual learning.
- Use 20% of the training dataset as the validation set.

**2.** Construct a **Multi-Layer Perceptron (MLP) model**. **(20%)**

- The parameter should be with:
  - hidden_sizes=[512, 256, 128, 64]
  - Output should have two labels.
  - With the following architecture:

```
Model Summary:
==================================================
Input Size: 10000
Layer 1: Linear(10000, 512)
  Parameters: 5,120,512
  Activation: ReLU + Dropout(0.3)
Layer 2: Linear(512, 256)
  Parameters: 131,328
  Activation: ReLU + Dropout(0.3)
Layer 3: Linear(256, 128)
  Parameters: 32,896
  Activation: ReLU + Dropout(0.3)
Layer 4: Linear(128, 64)
  Parameters: 8,256
  Activation: ReLU + Dropout(0.3)
Layer 5: Linear(64, 2)
```

- Count the number of trainable parameters in the model using the automated function.

**3. Implement case 1: Bag-of-words (20%)**

- Implement the bag-of-words (max_features=10000).
- Hint: from sklearn.feature_extraction.text import CountVectorizer

**4. Implement case 2: Construct a function to use embeddings on the same model. (20%)**

- Use the model: meta-llama/Llama-3.1-8B or use bert-base-uncased if facing issues with the GPU constraints.
  TIPS:
  You can use the distilled version, gather embeddings for 200 samples, and even reduce the precision to deal with computing issues!
- Hints:

```
self.tokenizer = AutoTokenizer.from_pretrained(model_name)
self.model = AutoModel.from_pretrained(model_name).to(device)
self.embedding_size = self.model.config.hidden_size
self.model_loaded = True
```

USE BOTH CASES IN PARALLEL TO EACH OTHER (ONE WITH BOW AND ANOTHER WITH EMBEDDINGS). NOT ON TOP OF EACH OTHER.

**5. Train the model with 10 epochs and create the best-performing model (checkpoint.pt) on the Dataset 1. (10%)**

- Get the validation accuracy.

**6. Use the checkpoint from before and train on the IMDB dataset (Dataset 2). (10%)**

- Use the following parameters:
  criterion = nn.CrossEntropyLoss()
  optimizer = optim.Adam(model.parameters(), lr=0.0001)  # Smaller learning rate

**8. Compute the validation loss and accuracy on the validation set of the IMDB dataset. (10%)**

**9. Submission Requirements**

- **Python code** for training, testing, and evaluation.
- **Screenshots of the following** displaying:
  - Model architecture.
  - Hyperparameters.
  - Logged metrics.
  - Final evaluation results.
  - Confusion matrix visualization.
  - Training and validation loss curves.

Evaluation Criteria

- ☐ Implement resume training from checkpoint
- ☐ Add model parameter logging
- ☐ Implement checkpoint compression
- ☐ Add TensorBoard integration