Generative AI, Assignment # 2

Department of Computer Science
National University of Computer and Emerging Sciences,
Islamabad, Pakistan
Instructor: Dr. Akhtar Jamil

Due Date: November 03, 2025

Instructions

- Each student must submit the following three files packaged into a single ZIP file and named as **ROLLNO_NAME.ZIP**:
  - A Jupyter Notebook (`.ipynb`) or Python script (`.py`) with the complete implementation.
  - A detailed PDF report containing all details of your implementation written in LaTeX using Overleaf, following the Springer's LNCS paper format:
    Springer LNCS Template on Overleaf.
  - A plain text file (`.txt`) containing all the GPT prompts used for each question.
- Ensure that the code is well-structured with proper comments for each function. Include all necessary dependencies to ensure the code runs without errors.
- There is a **grace time of 2 hours** after the submission deadline expires. You must verify that your submissions are correct. Any submission received after this slack time will be considered late, and NO marks will be awarded.

1 CycleGAN Implementation for Person Face Sketches

1.1 Objectives

Implement the CycleGAN model based on the original paper, using the **Person Face Sketches** dataset available at:
`https://www.kaggle.com/datasets/almightyj/person-face-sketches`

The objective is to perform image-to-image translation: converting a face image into a sketch and a sketch back into a real face image.

At test time, the model should be able to:

- Take a sketch and generate a corresponding real face image.
- Take a real face image and generate a corresponding sketch.

The model must be trained end-to-end. It is highly recommended to use Google Colab for training and experimentation. Ensure that model weights are saved after every epoch to prevent loss of training progress. In case training needs to be restarted, resume from the last saved weights rather than beginning from scratch.

If you experience memory issues and cannot load the entire dataset, you may:

- Split the dataset into smaller batches.
- Reduce the number of training samples as a last resort.

Train your model for multiple epochs. Carefully monitor training progress and tune hyperparameters such as learning rate and batch size to optimize performance.

1.2 User Interface

You are also required to create a simple user interface to demonstrate the trained model. The interface must be deployed using Flask or any similar API framework. The UI should be user-friendly and capable of handling real-time image conversion tasks. The interface must allow users to:

- Upload a picture (or use live camera input).
- Automatically detect whether the input is a sketch or a real face and convert it accordingly.

Dataset: Person Face Sketches
`https://www.kaggle.com/datasets/almightyj/person-face-sketches`

2 Machine Translation using Transformers for English-to-Urdu

2.1 Objectives

Your task is to develop a machine translation model that translates text from English to Urdu using Transformer architectures. The objectives include:

- Implementing a Transformer-based model for English-to-Urdu translation.
- Training the model on a suitable parallel corpus.
- Evaluating the model's performance using appropriate metrics.

2.2 Dataset

For this task, you may utilize the following resources:

- **Parallel Corpus for English-Urdu Language:** Available at `https://www.kaggle.com/datasets/zainuddin123/parallel-corpus-for-english-urdu-language`, this dataset contains over 24,000 sentence pairs in both English and Urdu, making it suitable for training translation models.
- **UMC005: English-Urdu Parallel Corpus:** Detailed information can be found at `https://ufal.mff.cuni.cz/umc/005-en-ur/`. This corpus includes texts from various sources such as the Quran, Bible, Penn Treebank, and the Emille corpus, providing a diverse set of parallel sentences.

If you choose to create your own dataset, consider using Large Language Models (LLMs) to generate synthetic parallel sentences. Ensure that the generated data is accurate and aligns well with the nuances of both languages.

2.3 Implementation Guidelines

- **Model Architecture:** Implement the Transformer model as introduced by Vaswani et al. ("Attention is All You Need"). You may use frameworks such as TensorFlow or PyTorch, and leverage libraries like Hugging Face's Transformers for efficient implementation.
- **Training:** Train your model on the selected dataset. Pay attention to hyperparameters such as learning rate, batch size, and the number of epochs. Implement techniques like learning rate scheduling and gradient clipping to enhance training stability.
- **Evaluation:** Evaluate your model using metrics like BLEU (Bilingual Evaluation Understudy) to assess the quality of translations. Provide both quantitative results and qualitative analysis by presenting example translations.

2.4 Additional Tasks

To further improve your model's performance, consider:

- **Pre-trained Models:** Fine-tuning pre-trained multilingual models such as mBART (`https://huggingface.co/facebook/mbart-large-50`) can provide a strong baseline and accelerate training.
- **Data Augmentation:** Employ data augmentation techniques to increase the diversity of your training data, which can help in improving the model's generalization capabilities.
- **Subword Tokenization:** Utilize subword tokenization methods like Byte Pair Encoding (BPE) to handle out-of-vocabulary words effectively.

3 Image Generation with Diffusion Transformers

Note: This is a research-based question. You are required to extensively read and understand the paper shared with you. The title of the paper is: **"Representation Entanglement for Generation: Training Diffusion Transformers Is Much Easier Than You Think"**.

3.1 Objectives

You are required to implement and evaluate the **Diffusion Transformer (DiT/SiT)** model as described in the attached paper *Diffusion Transformers*. The aim is to study how transformer backbones can replace the conventional U-Net in diffusion models for image generation and to reproduce selected results from the paper.

3.2 Tasks

1. **Model Implementation:**
   - Implement a diffusion model where the denoising network is based on a Vision Transformer (ViT) backbone instead of a U-Net.
   - Use the SiT (Scalable Image Transformer) architecture variant described in the paper.
   - Incorporate the REG (Representation Enhancement Guidance) or REPA techniques to improve convergence and sample quality.
2. **Dataset:**
   - Train the model on a subset of CIFAR-10. To make training feasible, restrict the dataset to two classes (e.g., "cat" and "dog").
3. **Training Setup:**
   - Follow the hyperparameter setup provided in the paper (e.g., AdamW optimizer, batch size of 256, 250 denoising steps, v-prediction objective, and Euler–Maruyama sampler).
   - Explore the effect of scaling model size (e.g., small vs. medium transformer depth or hidden dimension).
4. **Evaluation:**
   - Compare your results against a baseline diffusion model that uses a U-Net backbone.
   - Evaluate with both qualitative samples (visual inspection of generated images) and quantitative metrics such as FID, Inception Score, or representational alignment.

- Report training stability and convergence behavior for both models.

5. **Analysis:**
    - Discuss the benefits and limitations of transformers as backbones in diffusion models compared to U-Nets.
    - Analyze the role of REG/REPA in improving sample diversity and training efficiency.
    - Highlight any differences between your reproduction and the results reported in the paper.