Generative AI, Assignment # 2

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

Due Date: April 14, 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 Image Generation using GAN and VAE on CIFAR-10

1.1 Objectives

You are required to use the CIFAR-10 dataset to train and compare two generative models: a custom GAN network and a Variational Autoencoder (VAE). For

the GAN, the generator will follow the architecture discussed in class. However, instead of using a traditional discriminator that distinguishes between fake and real images, you need to develop a custom neural network that evaluates the similarity between two images. In this setup, the discriminator takes a pair of images—one real and one generated—and outputs a similarity score. The objective remains the same: the discriminator aims to maximize the dissimilarity between real and generated images, while the generator attempts to minimize this score to produce more realistic outputs.

To ensure diversity in the generated images, you must also implement one or more techniques during GAN training such as *Mini-Batch Discrimination*, *Feature Matching*, or any other suitable strategy. This is crucial to avoid mode collapse and promote variation across the generated samples.

In parallel, a VAE will be implemented for the same image generation task. Ensure both models are trained using only the "cat" and "dog" classes from the CIFAR-10 dataset.

Finally, compare the results of GAN and VAE in terms of:

- Visual quality of generated images
- Convergence behavior during training
- Any quantitative metrics you find suitable (e.g., reconstruction loss for VAE, similarity score for GAN)

Tips: For the GAN's discriminator network, design it to take two input images and produce a single similarity score. Consider using architectures such as the Siamese Network to achieve this. Additionally, explore diversity-promoting techniques during training such as Mini-Batch Discrimination to improve the quality and variability of generated images.

Dataset: The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. It is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive with no overlap between them. For more information, visit:
`https://www.cs.toronto.edu/~kriz/cifar.html`

2 CycleGAN Implementation for Person Face Sketches

2.1 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.

2.2 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`

3 Machine Translation using Transformers for English-to-Urdu

3.1 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.

3.2 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.

3.3 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.

3.4 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.

4 Vision Transformer vs CNN on CIFAR-10

4.1 Objectives

In this task, you are required to implement a Vision Transformer (ViT) for image classification on the CIFAR-10 dataset. You will also implement a standard Convolutional Neural Network (CNN) as a baseline for comparison. The objective is to evaluate the performance of ViT in comparison to traditional CNNs for small-scale image classification tasks.

4.2 Implementation Guidelines

- **CNN Model:** Design and train a CNN-based model suitable for CIFAR-10 classification. Use best practices such as batch normalization, dropout, and appropriate activation functions. You may use existing CNN architectures like ResNet or VGG as a reference, or build a custom CNN from scratch.
- **Vision Transformer (ViT):** Implement a Vision Transformer model following the original ViT paper. Key components to include:
  - Image patching (e.g., 4x4 or 8x8 patches from the 32x32 image)
  - Linear embedding of patches
  - Positional encoding
  - Multi-head self-attention
  - Transformer encoder layers
  - MLP head for classification
- **Evaluation Metrics:** Evaluate both models on:
  - Accuracy
  - Precision, Recall, and F1-Score (optional)
  - Training and validation loss curves
  - Confusion matrix (optional but recommended)
- **Using Pre-Trained ViT:** In this task, you may use a lightweight pre-trained ViT model from Hugging Face Transformers or Timm library and fine-tune it on CIFAR-10. Compare its performance with the other two techniques you implemented.
- **Training Setup:** Use techniques such as data augmentation (random crop, flip, etc.), learning rate scheduling, and early stopping. Experiment with different hyperparameters (batch size, learning rate, number of layers/heads) and provide reasoning for your choices.

4.3 Comparison and Analysis

After training all models:

- Compare their classification performance in terms of accuracy and generalization.
- Comment on the training time and complexity of both models.
- Analyze the model size and number of parameters.
- Discuss the effectiveness of attention-based modeling in ViT versus spatial modeling in CNNs.

Provide visualizations such as training/validation accuracy and loss curves, and example predictions from both models.

4.4 Dataset

CIFAR-10 Dataset: The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 testing images. Classes include airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

Dataset URL: `https://www.cs.toronto.edu/~kriz/cifar.html`

You may optionally restrict training and comparison to a subset of classes (e.g., cats and dogs) to speed up experimentation.