Coding test

General Instructions

1. There are 2 tasks
2. Write code in python with either the pytorch or tensorflow libraries in a **single** jupyter notebook to solve the 2 tasks below
3. Run the code, make sure it shows all the results and submit the pdf of the execution. Use plots to show the training and testing losses.
   Do not print out the text list with the loss minimization steps!
4. I will look at how you formulate the task and how you implement the ideas
5. You are not expected to perform demanding training. Use COLAB if you do not have access to GPUs.

Content of the Jupyter Notebook

1. code to define and train the neural networks
2. code to demonstrate the performance of the trained model on the test set
3. visualization of the performance of the trained model on the test set
4. visualization of the **quantitative** performance during training and on the test set
5. text to explain how the tuning and the final performance evaluation are done
6. write comments in the notebook that explain your reasoning and what each function does.

Task # 1: Denoising Autoencoders

Do not print out the text list with the loss minimization steps!

1. **Model Training:** Train a denoising autoencoder using a dataset of 100 grayscale **natural** images, each of size 64×64 pixels.
   The dataset must have natural images (e.g., animals, houses etc);

2. **Noise Model:** Corrupt the input images with Gaussian noise. Choose a noise level such that **both** the image content and the added noise are clearly visible in the noisy inputs.

3. **Performance Evaluation:**

   - Compute the reconstruction error (find out suitable metrics for this task) for both the training and test sets.
   - Present the errors in a table (optional: show also in a plot the loss – Y axis – against training time – X axis – on both the training and test sets).
   - Include visual examples of reconstructed images from both the training and test sets.
   - Ensure that test images were not used during training and are corrupted using the same Gaussian noise level.

4. **Analysis:** Describe and discuss your observations. What do the reconstruction results tell you about the model's performance and generalization?

Task # 2: Encoding Image Ordering

Do not print out the text list with the loss minimization steps!

Description We consider a task involving two neural models: a **sender** S and a **receiver** R. The goal is to train S to communicate a binary message indicating the order of a pair of images.

- The sender S receives a pair of images, stacked vertically (concatenated in the channel axis) in a specific order (e.g., image A on top and image B on bottom), and is tasked with outputting a binary message $m \in \{0, 1\}$ that encodes this order.
- The receiver R receives the same two images as S, but in stacked a randomly shuffled order. It also receives the binary message m from S.
- Using the message and the input image pair, R must decide whether the order of the images it received matches the original order that S observed.

Implementation

1. **Feasibility and Strategy:** Discuss the feasibility of the task. Consider what strategies the sender model S and receiver model R might learn to successfully encode and decode the image ordering using a single-bit message.
2. **Data Preparation:** Use a subset of 100 randomly selected images from the CIFAR-10 dataset. Do not use the labels or restrict the selection to specific categories; simply sample the images uniformly at random.
3. **Model Architecture and Training:** Design simple yet effective deep neural network architectures for both S (the sender) and R (the receiver). Keep the architectures as minimal as possible to maintain clarity and efficiency.
4. **Training Procedure:** Train both models jointly or in sequence on the prepared dataset. Track the training progress by plotting the training loss over the course of iterations.
5. **Evaluation:** After training, compute the final classification error rate of the receiver model R on the training set. Present the results in a concise table format.
6. **Performance Analysis:** Discuss the overall performance of the models. Consider how well the task was solved, potential bottlenecks or failure cases, and whether the models appear to have learned an effective communication protocol.