Bonus Assignment

Submission Requirements:

- Total Points: 1
- Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
- Submit your GitHub link and video on BrightSpace.
- Comment your code appropriately ***IMPORTANT.***
- Make a simple video about 2 to 3 minutes which includes demonstration of your home assignment and explanation of code snippets.
- No late submission accepted.

M. Siddartha Reddy
700774070

Question 1: ***Question Answering with Transformers***

Use Hugging Face's transformers library to build a simple question answering system using pre-trained models.

Setup Instructions:

Before starting, make sure your Python environment has the transformers and torch libraries installed.

Assignment Tasks:

1. Basic Pipeline Setup

- Import the pipeline function from transformers.
- Initialize a question-answering pipeline using the default model.
- Ask a question based on the given context**.**

Expected output

- 'answer': 'Charles Babbage' (or close variant)
- A confidence 'score' key with a float value above 0.65
- Valid 'start' and 'end' indices

2. Use a Custom Pretrained Model

- Switch to a different QA model like deepset/roberta-base-squad2.

Expected output

- 'answer': 'Charles Babbage'
- 'score' greater than **0.70**
- Include 'start' and 'end' indices

3. Test on Your Own Example

- Write your own 2–3 sentence context.
- Ask two different questions from it and print the answers.

Expected output

- Include a relevant, meaningful 'answer' to each question

- Display a 'score' above **0.70** for each answer

Question2:

1. Digit-Class Controlled Image Generation with Conditional GAN

Objective:

Implement a Conditional GAN that generates MNIST digits based on a given class label (0–9). The goal is to understand how conditioning GANs on labels affects generation and how class control is added.

Task Description

1. Modify a basic GAN to accept a digit label as input.
2. Concatenate the label embedding with both:
    - the noise vector (input to Generator),
    - the image input (to the Discriminator).
3. Train the cGAN on MNIST and generate digits conditioned on specific labels (e.g., generate only 3s or 7s).
4. Visualize generated digits label by label (e.g., one row per digit class).

Expected Output

- **A row of 10 generated digits, each conditioned on labels 0 through 9.**
- **Generator should learn to control output based on class.**
- **Loss curves may still fluctuate, but quality and label accuracy improves over time.**

Short Answer

- **How does a Conditional GAN differ from a vanilla GAN?**
  → Include at least one real-world application where conditioning is important.

Ans:

Vanilla GAN: Generates random, but realistic, samples from a learned distribution. No control over specific output.

Conditional GAN (CGAN): Generates specific, realistic samples by taking additional "conditioning" information as input, allowing control over the output.

Real-world application: Text-to-Image Synthesis. CGANs generate images based on text descriptions (e.g., "a cat wearing a hat"), which requires conditioning on the input text

- **What does the discriminator learn in an image-to-image GAN?**
  → Why is pairing important in this context?

Ans:

The discriminator in an image-to-image GAN learns to tell if an input image-output image pair is real (a true transformation) or fake (a generated transformation).
Pairing is crucial because it provides:

1. Direct supervision: A clear ground truth for the generator to learn from.
2. Meaningful mapping: Ensures the generated output directly corresponds to the specific input.
3. Reconstruction accuracy: Allows pixel-level comparison (e.g., L1 loss) for sharper, structurally similar results.
4. Stability: Aids in faster and more stable training.