Theoretical Tasks

Task 2.1 Ethical Assessment

Theoretical Background

Models such as GPT-4, Stable Diffusion, and Copilot are impressive in their ability to generate realistic images but have limitations. These models require an extensive amount of training data that can be difficult to obtain and may not fully represent the complexity of the world we live in. Therefore, they may generate images that are visually convincing but contain inaccuracies or inconsistencies with reality.

While these models are undeniably useful for certain purposes, it is essential to remember that they have limitations. The models should not be solely relied upon as a complete representation of reality. It is necessary to consider their constraints and limitations while using them for practical applications. Therefore, performing a thorough analysis and validation before implementing these models in real-world scenarios is crucial. By doing so, we can ensure that these models are helping us to achieve our goals without compromising their reliability and accuracy.

Here, you will use any of the generative image LLMs (GPT-4, Copilot, Gemini, etc.) to generate pictures of humans in real-life scenarios. The main goal is that you asses the images generated (at least 5) and carefully check for inconsistencies.

Present your findings (writing) with examples (optional). You should be able to find one inconsistency with reality and present that, explaining why you think it happened.

Hint: Check their body composition!

Practical Tasks 2

This lab focuses on helping you better understand the different methods used for generating images. You can use the codes in the links provided, but you are expected to understand them and may be asked to explain what certain parts of the code do.

NOTE: This is an extension of an old lab so be aware of the TensorFlow code; we suggest you use it as a guide, not develop your code in TensorFlow.

Task 2.1

Implement a Vanilla GAN as described by Goodfellow in his paper: `https://arxiv.org/abs/1406.2661` to generate adversarial MNIST images. Here is a link to a Tensorflow implementation where they explain the code and its steps `https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/`. The paper links to a github which has a Pytorch version you can use as a guide for building your vanilla GAN. Run the code with the original loss described by Goodfellow and observe the results.

Task 2.2

Run the same code with a different loss function: Logistic loss as described in Brandon Amos blog `http://bamos.github.io/2016/08/09/deep-completion/` and compare the results with above task 1.1. Run the code for 20k iterations and then for 100k iterations and observe the results in both cases. How/why is the output different for both cases? Try to find a suitable reason for both.

Task 2.3

The vanilla GAN only generates images of random numbers between 0-9 but has no labels associated with the generated images. This means that if you want it to generate MNIST-looking images of 3s, then it will not understand and generate a random number instead. You will now add/change code in your GAN from task 1 to create a conditional GAN (CGAN) and then plot a few generated images of any number you choose between 0-9 to see if your GAN knows the classes.
HINT: The Discriminator and Generator now have two inputs instead of one.

Task 2.4

First, train a CNN MNIST model and create adversarial images to classify 4s as 9s. **HINT:** Take help from Jasoni Carter Github : `https://github.com/jasonicarter/MNIST-adversarial-images` Try to reason how these adversarial images are produced. Next, a random noise image was used to classify it as nine and observe the results. Compare the results from the first and the second steps.

Task 2.5

Now, you will implement a stable diffusion model to generate images based on MNIST. Stable diffusion models are complex and costly, so take inspiration from this article `https://levelup.gitconnected.com/building-stable-diffusion-from-scratch-using-python-f3ebc8c42da3`, which shows you how to create a simple, stable diffusion model. Train the diffusion model to generate images of any desired number, and compare the results with the images from your CGAN (task 2.3). What is the main difference between GAN and stable diffusion?