Assignment 4

Matheus Vinicius Ferreira Figueiredo Teixeira  
301236904

# Introduction

In this assignment, we implement a Generative Adversarial Network (GAN) using TensorFlow to generate images of pants from the Fashion MNIST dataset. GANs, introduced by Goodfellow et al. in 2014, consist of two neural networks—a generator and a discriminator—that work in opposition. The generator creates images from random noise, while the discriminator learns to distinguish between real and generated images. Our implementation leverages modern deep learning techniques including batch normalization, leaky ReLU activation functions, and transposed convolutions. We examine the model's performance across different hardware setups and analyze scaling considerations for larger training datasets.

## Build the Generator Model of the GAN

### Display (print) a summary of the model using summary(). **Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons) and number of weights in each layer. Note: The generator model should output an image the same dimension as the dataset**

A blue rectangular object with white text

Description automatically generated

## Build the Generator Model of the GAN

### Display (print) a summary of the model using summary(). **Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons) and number of weights in each layer.**

A blue rectangular box with white text

Description automatically generated

## Using the train\_dataset\_firstname from Step b.6 and the training function defined in Step f.3, train the models in batches with 10 epochs. Use Python's time module to calculate and display (print) how long each epoch takes. # Note: GAN's are trained typically on tens of thousands to hundreds of thousands samples with large number of epochs. In your report, calculate and explain how long it would take to train the same model using 70,000 training samples on 100 epochs using your current hardware.

To estimate the training time for 70,000 samples over 100 epochs, we must first analyze training performance of our current setup

* **Samples trained:** 7,000
* **Number of epochs:** 10

For this setup, we observed varying epoch times ranging from 0.31 to 3.62 seconds, giving us an arithmetic mean of μ ≈ 0.6635 seconds per epoch.

For the following proposed setup:

* **Samples trained:** 70,000
* **Number of epochs:** 100

we would have 10 times more images per epoch; therefore, assuming linear scaling each epoch would take 10 times longer to be executed:

Multiplying this by the new total number of epochs our new training time would be:

If I was running this in a machine without GPU, such as my laptop the time could be much higher, about 14.32 hours, in my local machine. However, several factors could influence this estimation: thermal throttling of hardware components during extended training periods, system memory management overhead, and potential background processes. Furthermore, our model's architecture, comprising approximately 2.3M parameters in the generator and 424K parameters in the discriminator, suggests significant computational requirements. Taking these factors into consideration, a conservative estimate could go up to 16-18 hours of total training time to account for system variations and potential performance degradation during extended operation.