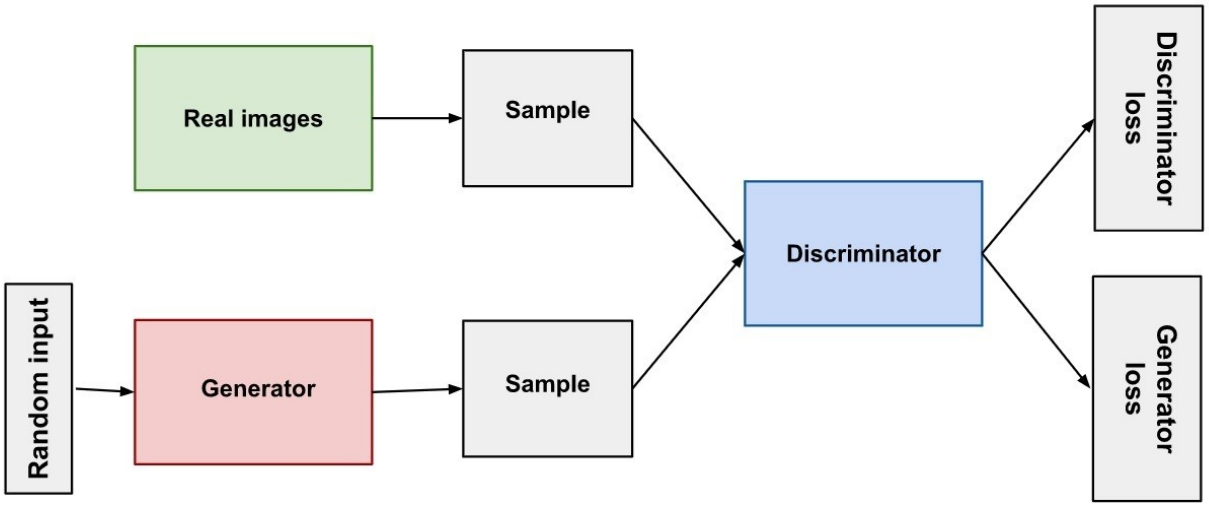
GAN Fashion-MNIST

GAN architecture & F-MNIST introduction:

A GAN's (aka Generative Adversarial Networks) is a network composed by two neural networks Generator network and the Discriminator network. our goal is to pass through a random distributed noise vector (gaussian distributed noise vector) and learn the transformation to the training distribution. this should help us generate images that match the images classes distribution. the discriminator network tries to distinguish between real and fake images while the generator network tries to fool the discriminator network hence this evolves into a classical game theory problem in a two player game.



F-MNIST introduction:

Fashion MNIST is a well know dataset similar to the MNIST (numbers 0-9) set. I choose to use this set as I want to show arithmetic operations on averages of produced images by the generator (i thought **pullover - t-shirt** would be interesting as well as (**ankle boot - shoe + sandal**).

The dataset is composed of 60k images and 10 classes. A few examples of images from these classes are shown below:

Generator & Discriminator code:

My **generator network** receives as input a noise vector size 100, it then uses a sequence of linear layers with Batch normalization and ReLU activation functions to transform it into a generated image. forward method, the input noise vector z is passed through the generator's model, and the output is reshaped to the size (batch size, 1, 28, 28) to match the dimensions of a single-channel (grayscale) 28x28 image.

Linear(z,128)->reLU()->linear(128,256)->batchnorm(256)->reLU()->linear(256,512)->batchnorm(512)->relu()->linear(512,img_shape)->tanh()

Our **discriminator network** receives as input a image of size image_shape (1,28,28) in our case, it then uses a sequence of linear layers with Batch normalization and LeakyReLU activation functions and outputs a single value representing the validity of the image.

Linear(img_shape, 512)->BatchNorm(512)->LeakyReLU(0.2)->Linear(512, 256)->BatchNorm(256)->LeakyReLU(0.2)->Linear(256, 1)->Sigmoid()

Producing Images from classes with a trained Generator:

I then save the models weight parameters to the files gans_weights.pth and gans_weights_discriminator.pth **after training**. we then can load the model generator and produce 100 images from classes as shown below (the following is a run of the main in myscript.py file):

as seen above the picture may not be as crisp as the originals but one could certainly say this is a huge improvement rather than guessing 28x28 pixels for each image. we can clearly see the images "belong" to a certain class out of the ten classes. note these images above are not from the training set and have been produced through the generator by simply feeding it a noisy vector size 100 (noisy with gaussian distribution)

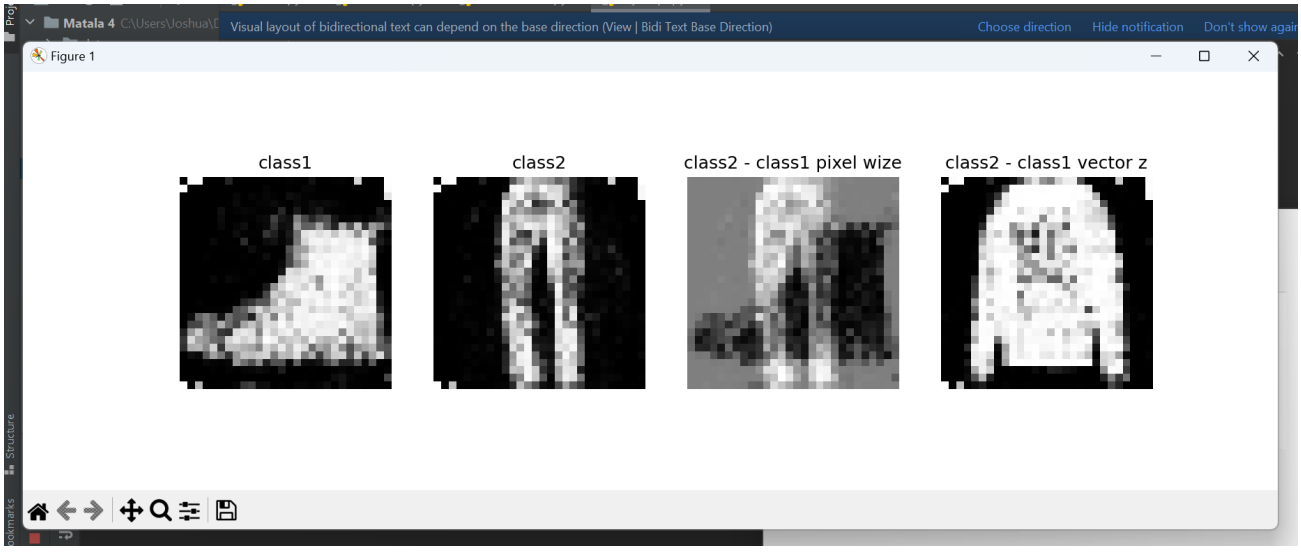
Computing arithmetical operations on means of samples of class images:

here I produced two noisy vectors and a third vector which is the combination of v3 = v1 - v2 or v3 = v1 + v2.

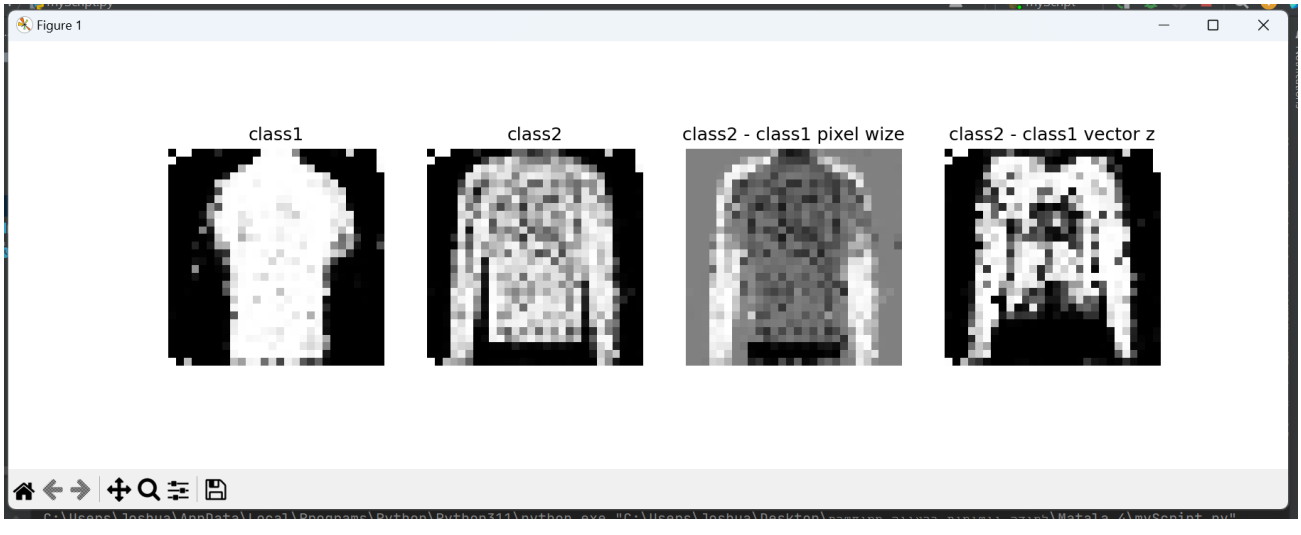
The Generator then generated three images with the three vectors and I then plotted them as well as image1 - image2 / image1+image2 which is pixel wise subtraction / addition rather than on the noisy vector before feeding it to the generator and here are some findings:

Findings:

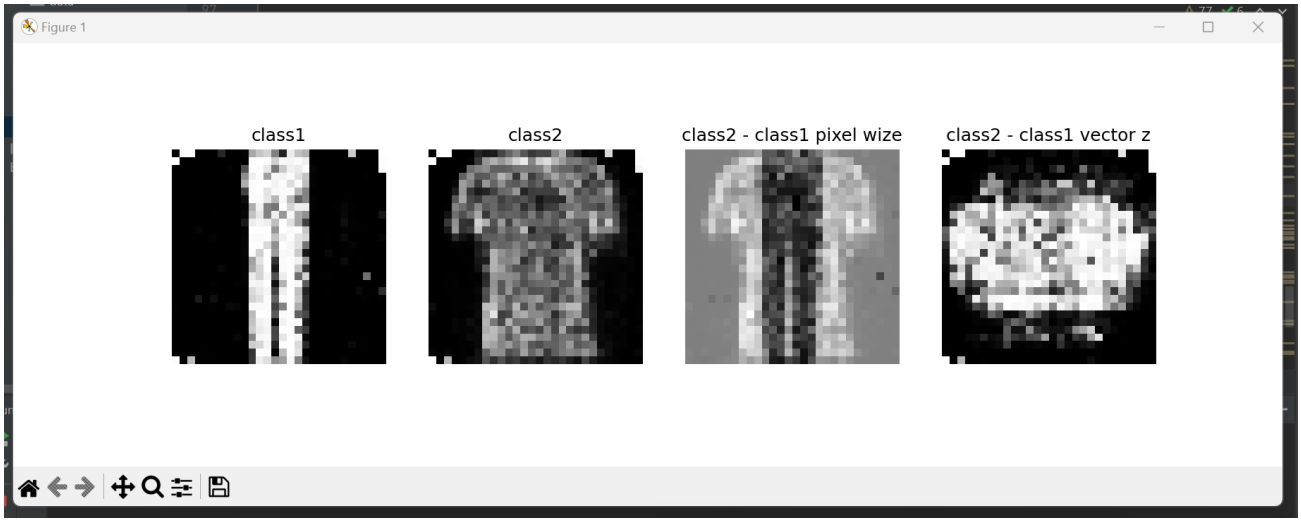
we can see the pixelwise subtraction in image 3 as opposed to the vector wise subtraction before feeding to the Generator. this would indicate that the v3 = v2 - v1 was sensitive to the change and produces a generative image of a different class (top/shirt)



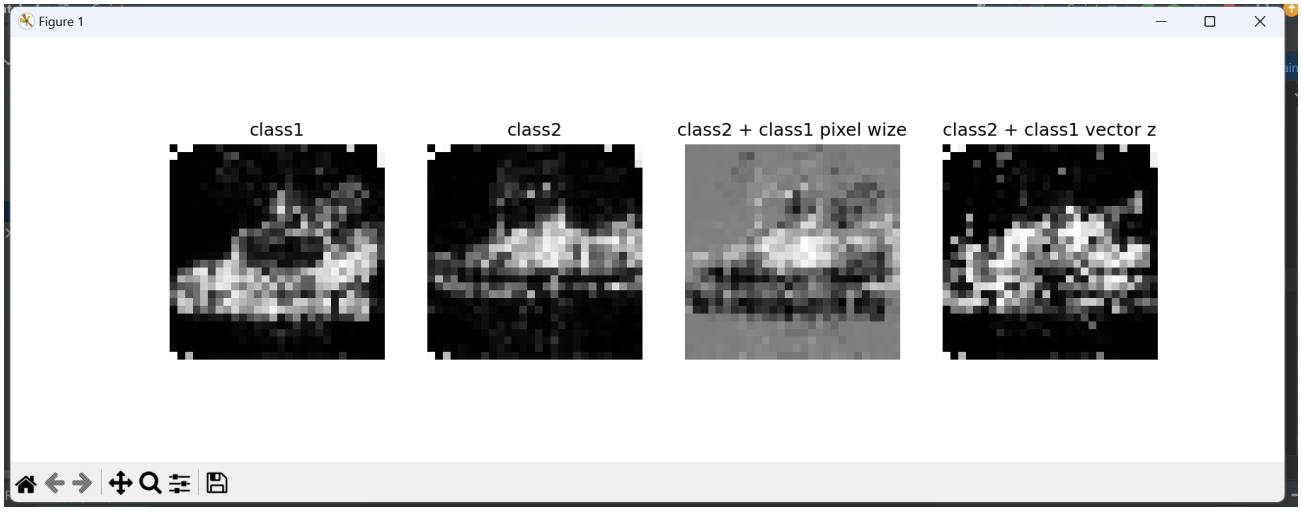
first of all image 3 , pixel wise subtraction came out really cool (looks like 2002 FIFA quality). we can see v3 = v2 - v1 kind of produce a hybrid image of a t shirt and shirt/top.



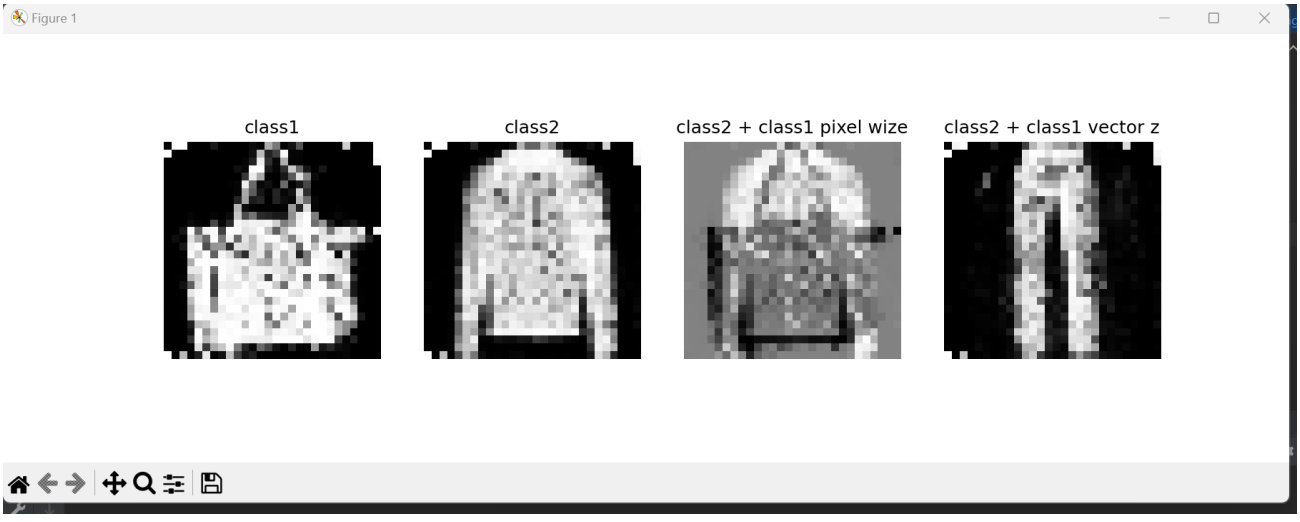
here we can see that a very noisy image was produced with v3 = v2 - v1. conclusion is classes trousers and shirt don't go well together.



once again image 3 pixel wise addition of generative sandal and sneaker images came out cool, definitely NFT worthy. how ever the v3 = v2 + v1 did not come out so well although it dose resemble a shoe (kind of).



as before it seems the hand bag generative image and top/shirt vectors v2, v1 created a misleading vector v3 = v1 + v2, that produces a clearly generative trouses image.



About the Code:

open the zip folder and extract to your local environment.

note the files Discriminator.py and Generator.py container the networks for the GAN

the file **loader.py** loads the Fminst data set and trains the network. you do not need to do this as the model has been trained and saved parameters to the pth files!

Myscript.py contains the main function containing:

plotting 10 x 10 grid with generative images from the Gan.

1 loop generating 5 subtractions and plots pixelwise and vector wise (before feeding to the model)

1 loop generating 5 additions and plots pixelwise and vector wise (before feeding to the model)