Hi all
Welcome to our fifth, one-before-last, exercise!
This exercise will be done alone, but as in the previous one, you are welcome to share code inside your group and to look for references online, as long as you mention your sources.
In this exercise you will implement a version of Generative Latent Optimization - specifically, the first stage of the algorithm described at “Demystifying Inter-Class Disentanglement” by Aviv Gabbay and Yedid Hoshen. You are encouraged to read the paper. I will walk you along the method here, in the exercise description.

Exercise requirements:

- Implement Generative Latent Optimization (GLO) with content-class embedding and noise regularization. Train it with a simple model on MNIST dataset (cpu is enough) and answer a few questions about it. This method was described by Yedid in the end of the second lecture.

More information:
Datasets:
We will use MNIST only. We treat the labels (digits) as “class” and the image ids as content. Every image has its own “content” vector. Please use only the first 2000 images of the dataset, so this whole thing will converge quickly.

Model:
In the moodle there is a pytorch model to be used as a generator. You are not required to implement the architecture mentioned in the paper.

Computation:
I trained it on my i5-fifth gen cpu in a few minutes.

Generative Latent Optimization

Model diagram:

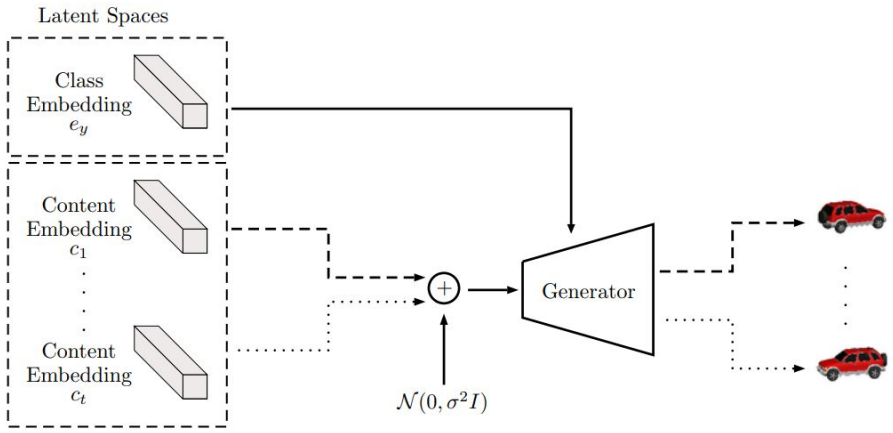

Figure 1: A sketch of the first stage: all class and content embeddings and the generator are jointly optimized. All images of the same class share a single class embedding. The content embeddings are regularized by a gaussian noise. By the end of this stage, the latent space of the training set is disentangled. Note that the second stage is not shown.

The code you have downloaded from the moodle contains a simple generator model to be used with MNIST. The results are not nice as in the paper, but the model is small enough to be trained on a cpu.

1. Implement generative latent optimization with noise regularization as described in the diagram above. Implement the training loop and train the model on MNIST, log the loss and some reconstructed image examples using tensorboard. (yes yes, tensorboard has this feature!). Present your results.
    a. I trained it with Adam optimizer, default learning rate, 50 epochs, loss=L1+L2, noise stddev=0.3.
    b. the reconstructed images won't be perfect and the disentanglement effect won't be as nice as in the paper.
    c. Don't use perceptual loss, it is too heavy to run on the CPU.
2. What will happen if you will set the noise too high? Show it.
3. What will happen if you will set the noise too low? Explain how classifying the class by the content vector can quantify this effect.
    a. you don't need to actually implement that.

A note about pre-made models -
I am giving you the Network architecture because I want it to be possible to run it on the CPU and It took some time for me to find a lean, working configuration. Also, I think there is no much added value in dwelling into transposed convolutions configurations.