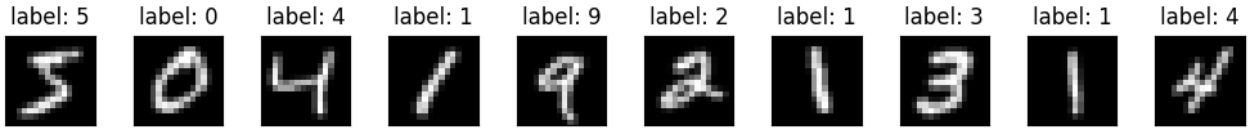
Report

SMILES - GANs - HW1

Part 1: conditional GANs

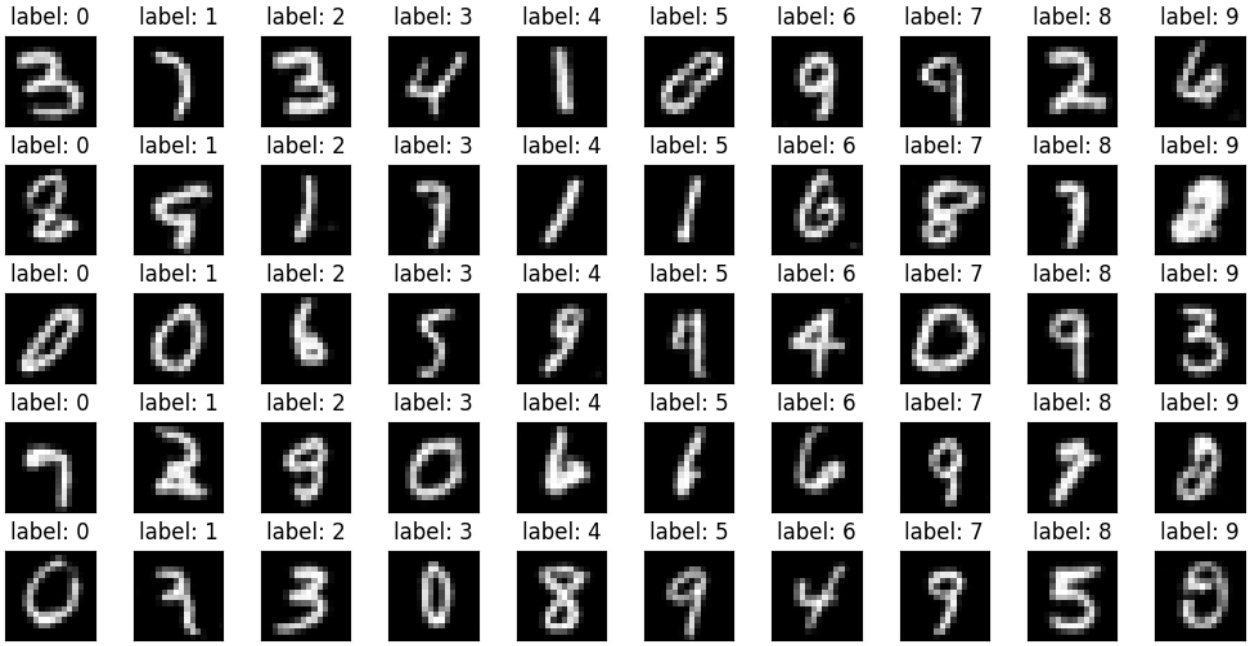
The goal is to learn generator to draw numbers in MNIST style with respect to the given one-hot encoded class (number) mark. Both generator and discriminator have a simple convolutional architecture.

The following picture shows GT examples:



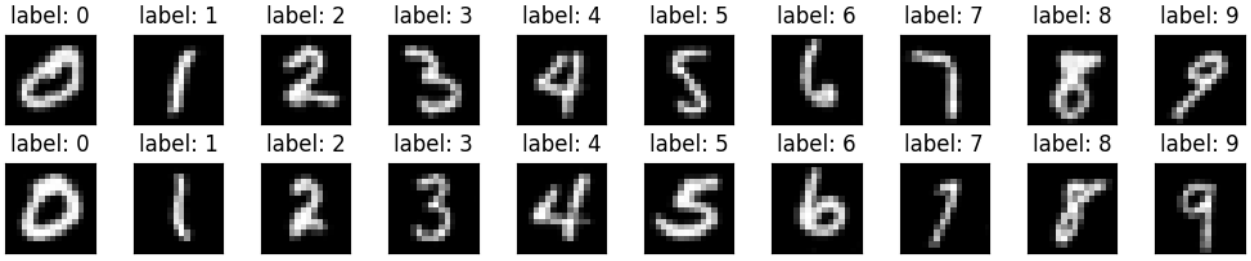
Vanilla GANs

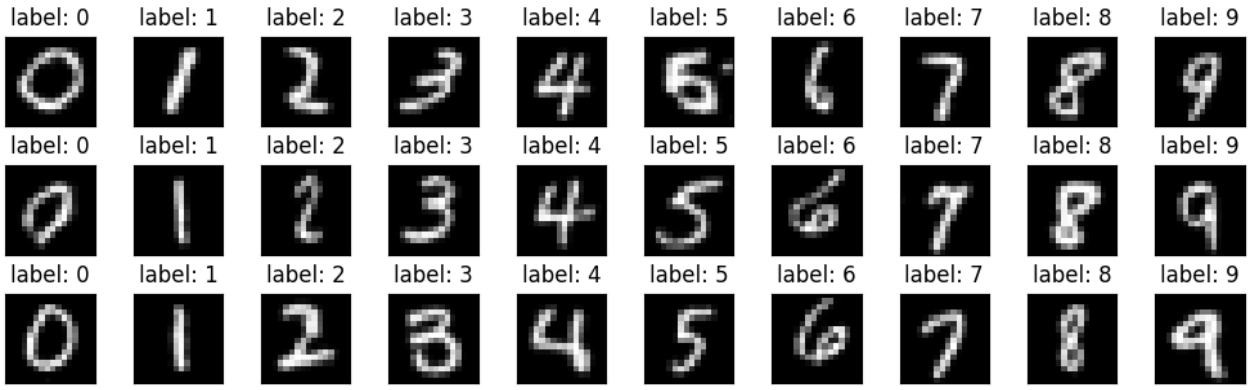
Let's implement conditional vanilla generator and classic vanilla discriminator and train them for 100 epochs with batch size of 256, 1 discriminator step and LR = 0.0002. The following pictures show results for different conditions:



The generator behaves like ordinary vanilla GAN generator without any respect to conditions. The quality of the generated pictures sometimes is poor: maybe the longer train procedure can fix this issue. The behaviour is also under discussion: my colleague @olgamatykina, who used "fair" embeddings in one of her experiments, noticed that generator learns to draw the same numbers with same condition, but they may not correspond to the GT mark.

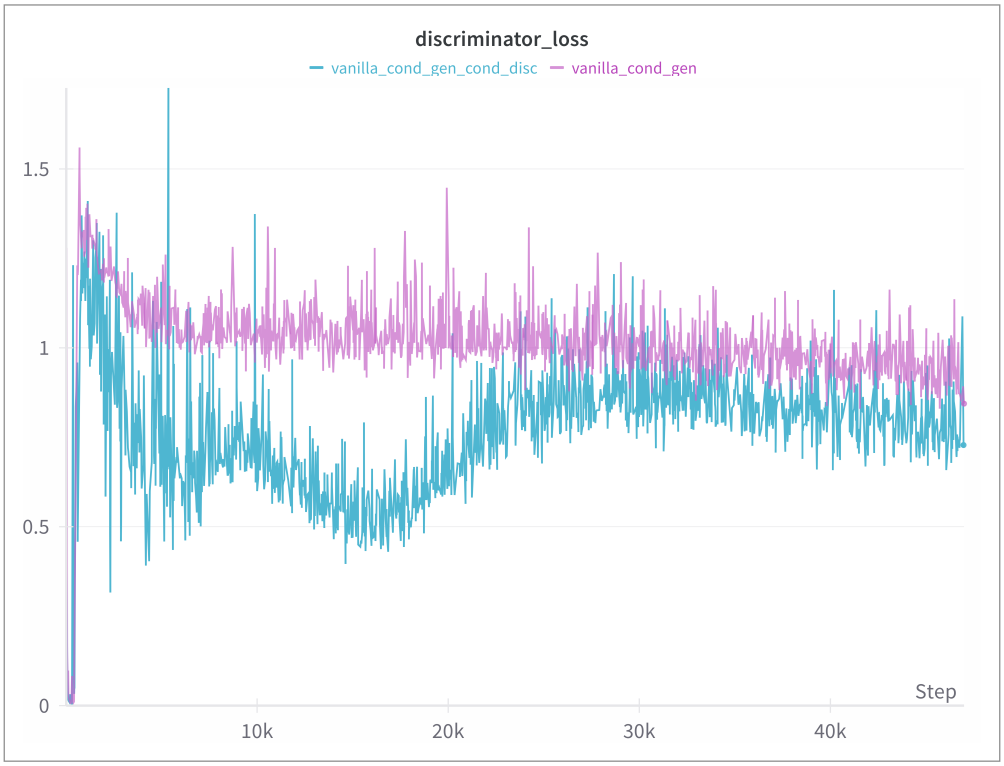
Let's implement conditional discriminator and compare results in the same setup. After 100 epochs we get the following:





The quality of the generated numbers has increased and now they match their conditions.

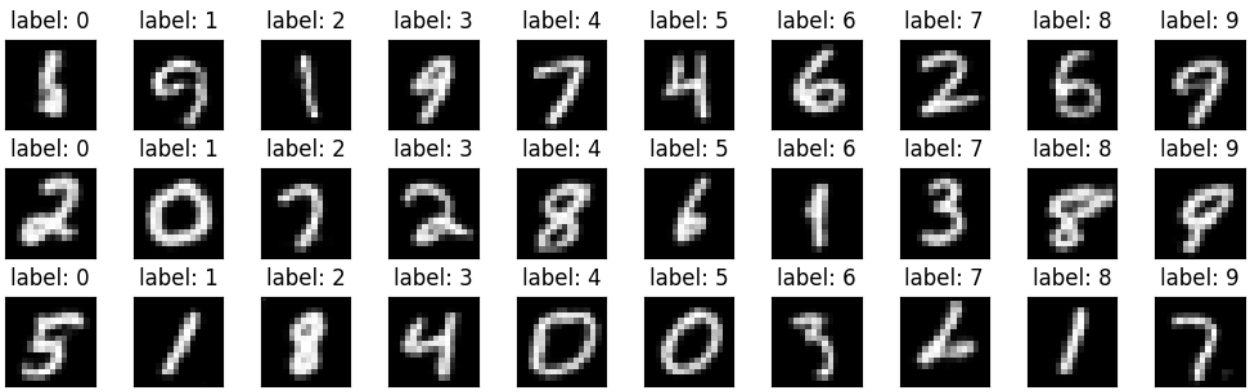
In the discriminator loss graph we can see the specific behaviour that was noticed by @emptyhooks in “Kypc GANs”.

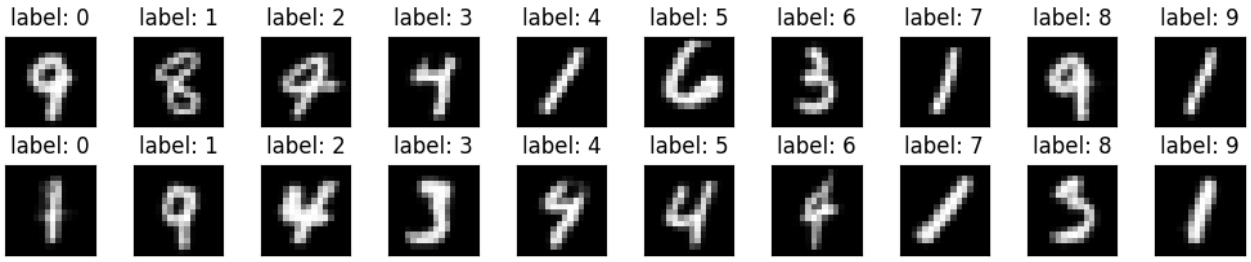


NS GANs

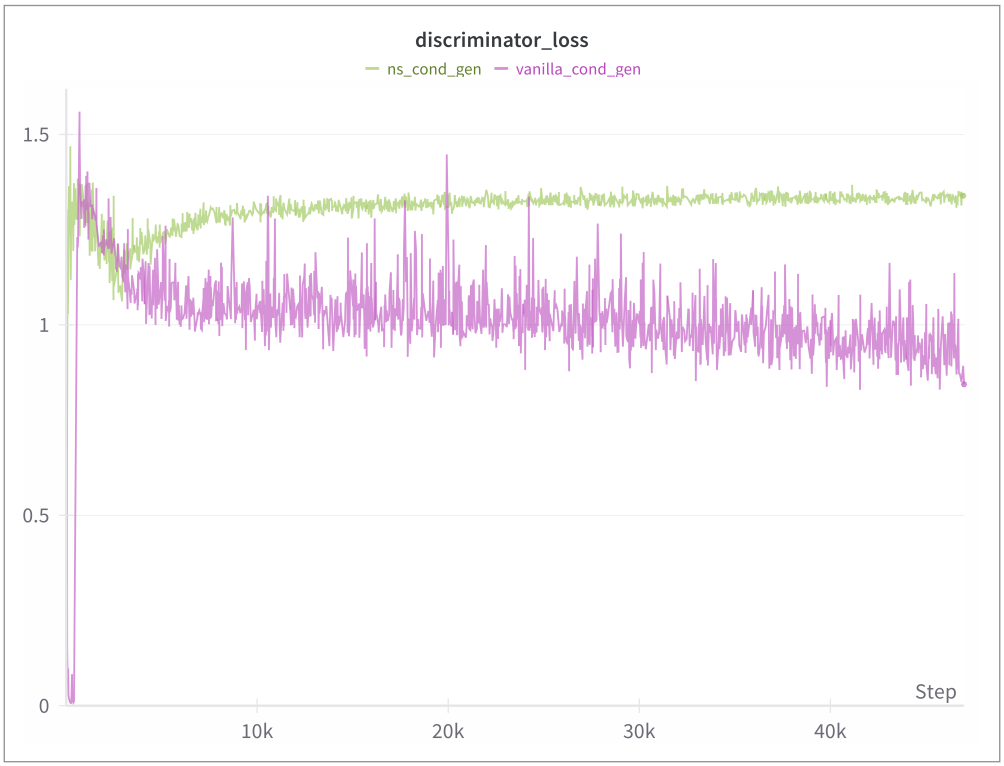
Now let’s implement non-saturating algorithm and compare results.

In case of conditional generator and ordinary discriminator we get the following after 100 epochs:

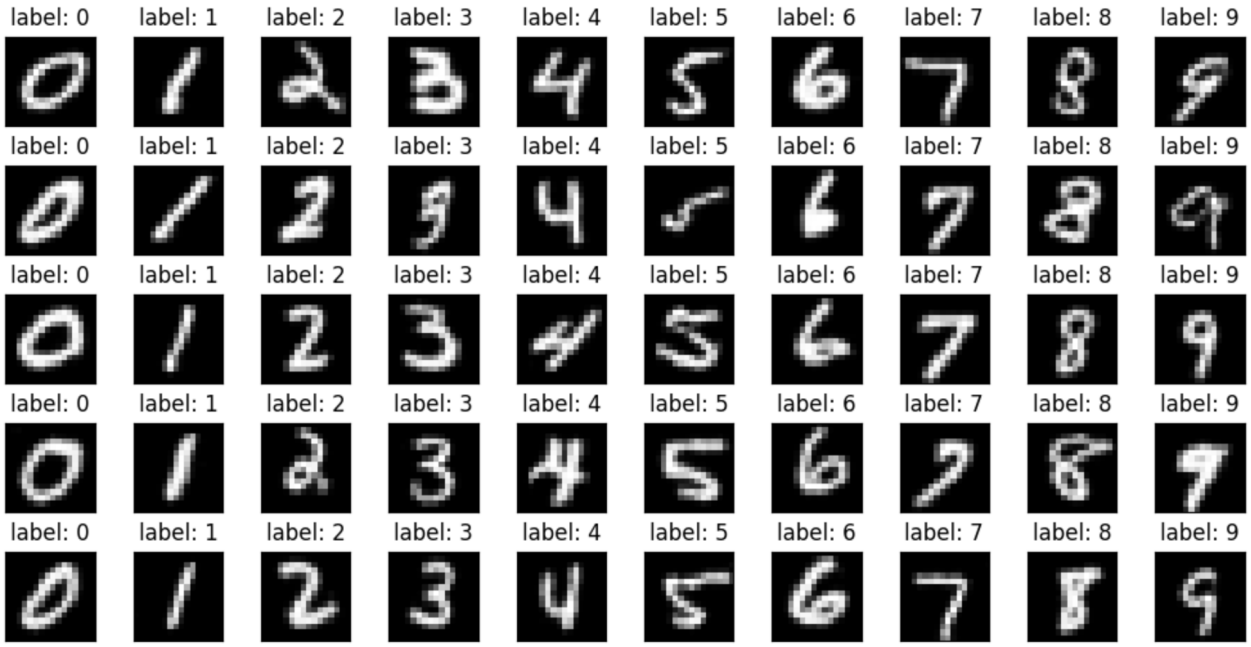




The behaviour is the same with vanilla case, but the quality has increased. Generator loss looks more stable:

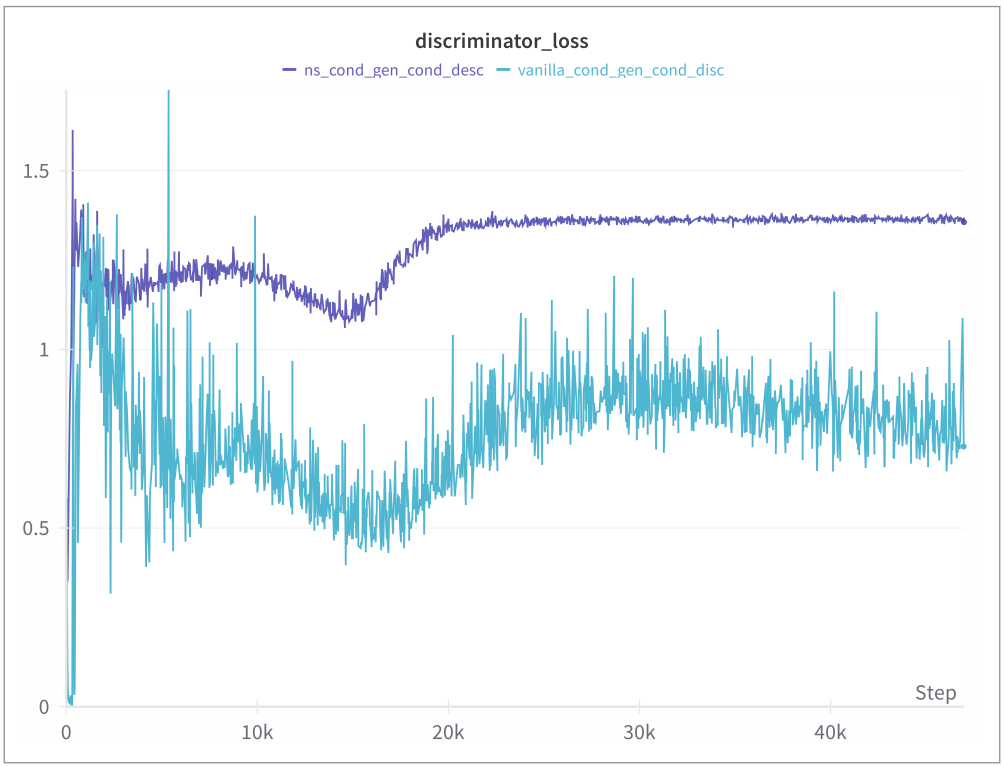


In the “full-conditional” setup we get the following results (still 100 epochs):



The quality also looks a bit better.

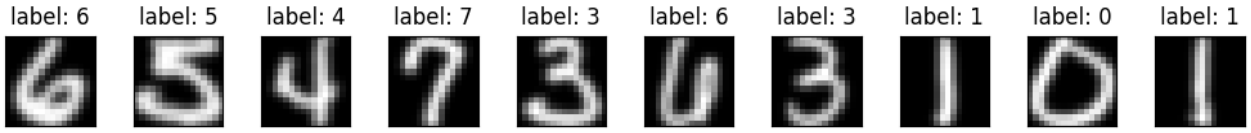
Discriminator losses comparison shows more stable graph in NS case, which also has some kind of asymptote in the second half of the training process.



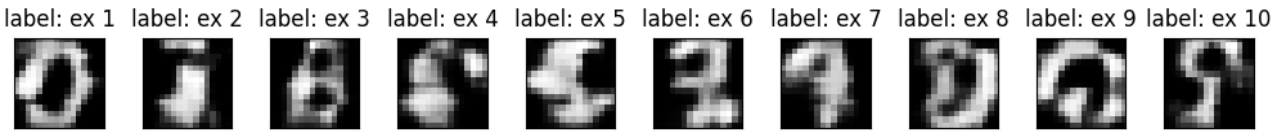
Part 2: Wasserstein GAN

The goal is to learn Wasserstein generator with spectral normalisation to draw numbers in USPS. Both generator and discriminator have a simple convolutional architecture.

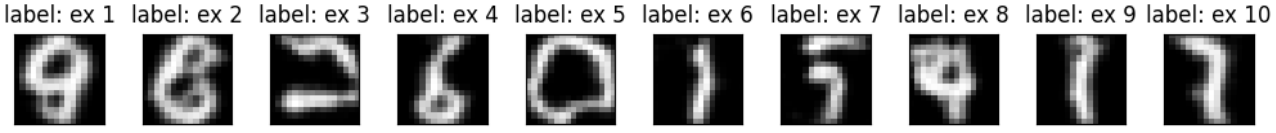
The following picture shows GT examples:



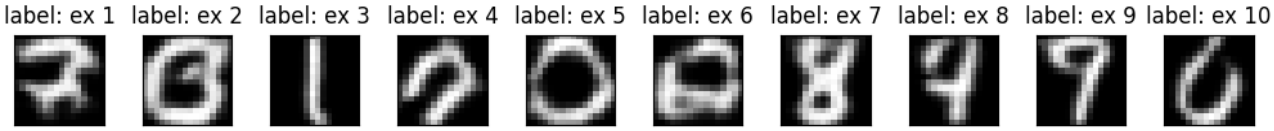
In this very sad part of homework no any acceptable results were reached: for different spectral normalisation steps, discriminator steps, LR and even number of epochs the quality is very poor. For example, 100 epochs with batch size of 256, 1 critic step, 2 normalisation steps and LR = 0.0002 we get the following:



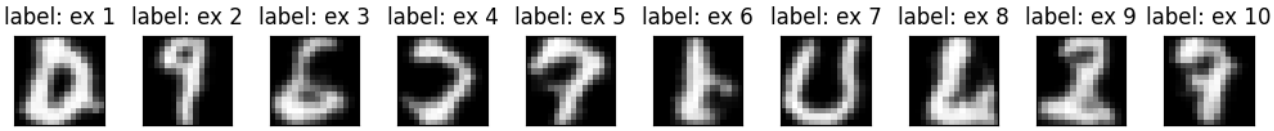
100 epochs with batch size of 256, 1 critic step, 2 normalisation steps and LR = 0.002:



100 epochs with batch size of 256, 1 critic step, 1 normalisation step and LR = 0.002:



1000 epochs with batch size of 256, 1 critic step, 1 normalisation step and LR = 0.002:



… but the author didn’t find a mistake in the training algorithm and gave up.