INSTITUT POLYTECHNIQUE DE PARIS

SE314 - Optimal Transport : theory, computations, statistics and ML applications

Report on OT-GAN

Prepared by: Jingyi LI, Wendong LIANG
Date: May 4, 2022

Contents

1 Introduction

Generative Adversarial Network (GAN) is an example of generative models. A generative model is able to generate new samples from a hidden probability distribution. Unlike the classical generative models such as gaussian mixture, GAN does not contain an explicit form of the hidden distribution. It uses a neural network generator to characterize a distribution from uniform distribution, and a disriminator or critic as a classifier to distinguish the true from the generated.

In the original formulation of GAN, the two main components are:

- **Generator**: The generator is a neural network represented by a map $g_\theta : \mathcal{Z} \to \mathcal{X}$ where θ represents the weights. The map takes a Gaussian or uniforme Z as input, and the goal is to make the distribution generated closed to real data X.
- **Discriminator**: In the original paper of [1], the discriminator is a map $D_w : \mathcal{X} \to [0, 1]$, and its goal is to assign 1 to real samples and 0 to generated samples.

The parameters of GAN are (θ, w). They are obtained by solving the min-max problem:

$$\min_\theta \max_w \mathbb{E}[\log(D_w(X)) + \log(1 - D_w(g_\theta(Z)))]$$

In practice we do not solve this directly because it is intractable. However, we can find a equivalent formulation of this problem: if the maximization over w is instead over all functions D, then the min-max problem reduces to

$$\min_\theta JS(P_X, P_\theta)$$

where $JS(P_X, P_\theta)$ is the Jensen-Shannon divergence:

$$JS(p, q) = KL(p, \frac{p+q}{2}) + KL(q, \frac{p+q}{2})$$

This result reduces the min-max problem of two parameters to a minimization of one parameter, and the JS distance can also be relaxed or changed to other distances. And WGAN's idea stem from that [2]. The main point of WGAN is to replace JS distance by Wasserstein-1 distance.

1.1 WGAN

If the topology induced by a distance is too fine, then it is hard for sequences to converge. For GAN this is crucial because it is hard to converge to the saddle point. Theorem 2 of [2] proved that the topology of KL is finer than Total Variation and JS, which are finer than W-1 distance. But after using W-1 distance, the quantity to minimize

$$\min_\theta W_1(P_X, P_\theta)$$

is highly intractable, so they use the Kantorovich-Rubinstein duality to compute it:

$$W_1(P_X, P_\theta) = \sup_{||D||_L \leq 1} \mathbb{E}_{x \sim P_X, Z \sim p}[D(X) - D(g_\theta(Z))]$$

where sup is over all functions with Lipschitz constant less than 1. Pluging this in the minimization problem we get:

$$\min_{\theta} \max_{w} \mathbb{E}[D(X) - D(g_\theta(Z))]$$

with $||D||_L \leq K$. In practice it is achieved by weight clipping, i.e. constrain the infinity norm of weights. But just as the authors say, "Weight clipping is a clearly terrible way to enforce Lipschitz constraint". If the threshold is too high then it would take too long for weights to converge; if too low then the gradients would disappear during training.

Another approach to WGAN is to approximate the Wasserstein distance by Sinkhorn distance, based on the penalization of entropy.

1.2 Sinkhorn-GAN

Here they did not approximate the primal form of optimal transport, but approximated the sinkhorn distance by evaluating it on mini-batches of data X, Y:

$$D_{Sinkhorn}(p, g) = \inf_{\gamma \in \Pi_\beta(p,g)} \mathbb{E}_{x,y \sim \gamma} c(x, y)$$

where $\Pi_\beta(p, g)$ is the joint distribution with entropy at least β. It can be calculated by Sinkhorn, as a matrix M. After that, we calculate the result distance as

$$\mathcal{W}_c(X, Y) = \inf_{M \in \mathcal{M}} Tr[MC^T]$$

In general, the idea of [3] is an interpolate between Wasserstein distances and Maximum Mean Discrepency (MMD). They defined the Sinkhorn loss between two measures μ, ν as

$$\bar{\mathcal{W}}_{c,\epsilon}(\mu, \nu) = 2\mathcal{W}_{c,\epsilon}(\mu, \nu) - \mathcal{W}_{c,\epsilon}(\mu, \mu) - \mathcal{W}_{c,\epsilon}(\nu, \nu)$$

which converges to Wasserstein distance or MMD when $\epsilon \to 0$ or ∞. The advantage is that this is quantity is solvable efficiently using Sinkhorn's algorithm.

1.3 OT-GAN

The disadvantage of Sinkhorn-diff is that the joint distribution matrix M is not a valid metric over probability distributions. The authors in [4] propose to instead use the Generalized Energy Distance, with the norm replaced by any distance d:

$$D_{GED}(p, g) = \sqrt{2\mathbb{E}[d(X, Y)] - \mathbb{E}[d(X, X')] - \mathbb{E}[d(Y, Y')]}$$

where x, x' are independent samples from data distribution p and y, y' are independent samples from the generator distribution g.

This distance can be directly generalized to mini-batch $\mathbf{X}, \mathbf{Y}$:

$$D_{MED}(p, g) = \sqrt{2\mathbb{E}[d(\mathbf{X}, \mathbf{Y})] - \mathbb{E}[d(\mathbf{X}, \mathbf{X}')] - \mathbb{E}[d(\mathbf{Y}, \mathbf{Y}')]}$$

Algorithm 1 Optimal Transport GAN (OT-GAN) training algorithm with step size α, using mini-batch SGD for simplicity

```
Require: $n_{gen}$, the number of iterations of the generator per critic iteration
Require: $\eta_0$, initial critic parameters. $\theta_0$, initial generator parameters
1: for $t = 1$ to $N$ do
2:     Sample $\mathbf{X}, \mathbf{X}'$ two independent mini-batches from real data, and $\mathbf{Y}, \mathbf{Y}'$ two independent
       mini-batches from the generated samples
3:     $\mathcal{L} = \mathcal{W}_c(\mathbf{X},\mathbf{Y}) + \mathcal{W}_c(\mathbf{X},\mathbf{Y}') + \mathcal{W}_c(\mathbf{X}',\mathbf{Y}) + \mathcal{W}_c(\mathbf{X}',\mathbf{Y}') - 2\,\mathcal{W}_c(\mathbf{X},\mathbf{X}') - 2\,\mathcal{W}_c(\mathbf{Y},\mathbf{Y}')$
4:     if $t \bmod n_{gen} + 1 = 0$ then
5:         $\eta \leftarrow \eta + \alpha \cdot \nabla_\eta \mathcal{L}$
6:     else
7:         $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \mathcal{L}$
8:     end if
9: end for
```

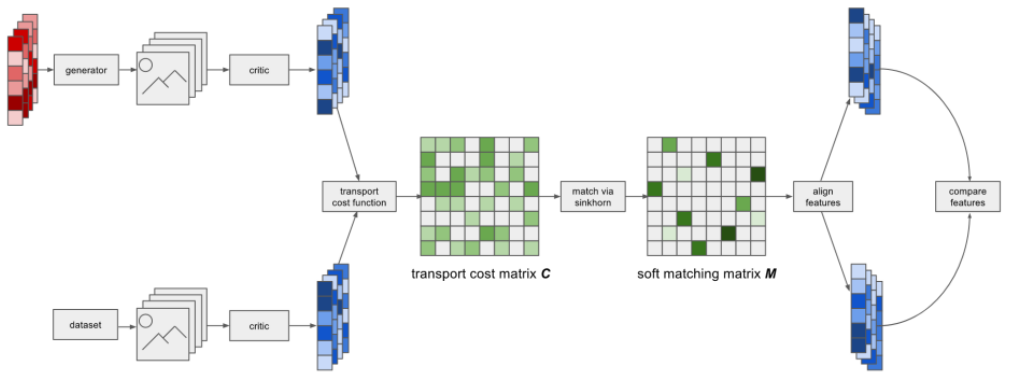

Figure 1.1: Figure 1: Illustration of OT-GAN. Mini-batches from the generator and training data are embedded into a learned feature space via the critic. A transport cost matrix is calculated between the two mini-batches of features. Soft matching aligns features across mini-batches and aligned features are compared. The figure only illustrates the distance calculation between one pair of mini-batches whereas several are computed.

Using entropy-regularized Wasserstein distance (Sinkhorn distance), we get the final version:

$$D_{MED}(p, g) = \sqrt{2\mathbb{E}[\mathcal{W}_c(\mathbf{X},\mathbf{Y})] - \mathbb{E}[\mathcal{W}_c(\mathbf{X},\mathbf{X}')] - \mathbb{E}[\mathcal{W}_c(\mathbf{Y},\mathbf{Y}')]}$$

where c is the cost matrix for optimal transport. Many choices are possible but the authors find the following the most stable:

$$c_\eta(x, y) = 1 - \frac{v_\eta(x) \cdot v_\eta(y)}{||v_\eta(x)||_2 ||v_\eta(y)||_2}$$

Unlike [3], they do not backpropagate through the Sinkhorn algorithm.

2 Experiments

2.1 OT-GAN

Table 1 shows the architecture of the generator of OT-GAN. To be adapted to the image size of MNIST, the 3rd convolution layer has a padding size 2 to make the final

operation	activation	kernel	stride	output shape
z				100
linear	GLU			16384
reshape				$1024 \times 4 \times 4$
2x NN upsample				$1024 \times 8 \times 8$
convolution	GLU	5×5	1	$512 \times 8 \times 8$
2x NN upsample				$512 \times 16 \times 16$
convolution	GLU	5×5	1	$256 \times 16 \times 16$
2x NN upsample				$256 \times 32 \times 32$
convolution	GLU	5×5	1	$128 \times 32 \times 32$
convolution	tanh	5×5	1	$3 \times 32 \times 32$

Table 1: Generator architecture for OT-GAN

operation	activation	kernel	stride	output shape
convolution	CReLU	5×5	1	$256 \times 32 \times 32$
convolution	CReLU	5×5	2	$512 \times 16 \times 16$
convolution	CReLU	5×5	2	$1024 \times 8 \times 8$
convolution	CReLU	5×5	2	$2048 \times 4 \times 4$
reshape				32768
l2 normalize				32768

Table 2: Critic architecture for OT-GAN

output $3 \times 28 \times 28$. For the critic, the change of padding is in the first convolution layer.

The input z of generator is 100 uniform random variables from -1 to 1. Then the main module of the generator is $w \times 2$ nearest-neighbor upsampling and a convolution with 5×5 kernel, using gated linear units:

$$GLU(a, b) = a \otimes \sigma(b)$$

where σ is the usual sigmoid function. We use Adam optimizer with learning rate 3×10^{-4}, $\beta_1 = 0.5$, $\beta_2 = 0.999$.

Our code is available here:
`https://colab.research.google.com/drive/1snh2OfLuE6DRdseyEuKyssaTHf9jYGh9?usp=sharing`.

2.1.1 MNIST

We implement the OT-GAN on MNIST dataset. We use batch size 256. In the beginning we wrote the dataloader as a zip of two independent loaders, and the time was 6min per epoch, with a relatively worse result. Now we rewrite the dataloader as a single loader of batch size 512, then split every batch into two batches. The time per epoch becomes 2min. That is also what the authors in OpenAI do in their code, but actually it is not strictly what the algorithm says in their paper. If we split the batch into two, then the probability that two batches have same images is zero.

Figure 2.1 shows some generated images during training epochs. The result is not stable, for example on the color of background, and the writing seems not progress too much during the training. Figure 2.2 shows the evolution of loss, from which we can see that most of the time the generator can beat the critic, since the loss is almost zero.

We also implemented conditional OT-GAN on MNIST with label 2. Figure 2.9 shows that the critics is not degenerated before 40 epochs. That is why the result is better than the previous experiment.

The parameter entropy is very important in Sinkhorn algorithm. In previous experiments we set penalty = 500. We also show in Figure 2.4 the result when the penalty = 100. The result has different feature in writings. Furthermore, if penalty =100 and $\beta_1 = 0.5$ in Adam optimizer, Figure 2.3 shows that the result of learning is completely indistinguishable.

2.1.2 CIFAR10

In the paper the authors used a batch size of 8000, which is infeasible for the memory of Colab. It would take thousands of epochs to get a meaningful image generator, and 100 epochs still show bad results. The loss figure shows that the generator dominates in the beginning, and then the critic dominates.

2.2 Sinkhorn-Autodiff

We implemented Sinkhorn-Autodiff algorithm in [3]. The input random vector is only of dimension 2, and it could already have meaningful figures from 10th epoch. It is also much faster than OT-GAN.

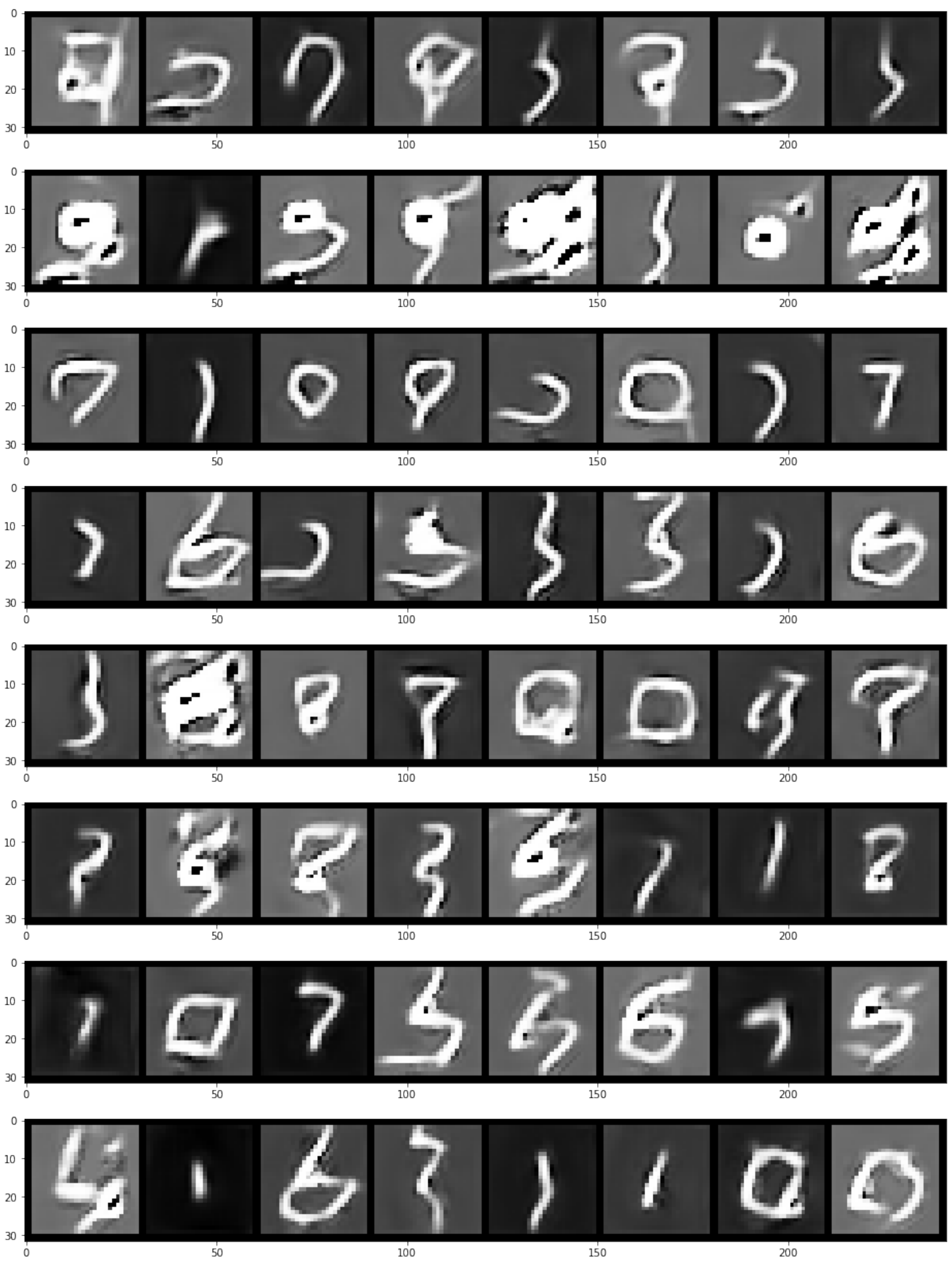

Figure 2.1: Samples generated by OT-GAN generator from 30th to 100th epoch.

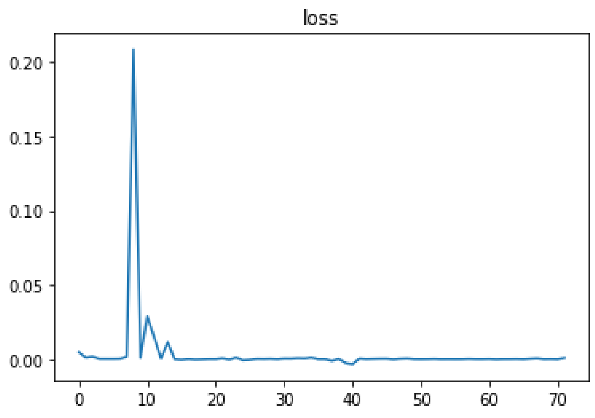

Figure 2.2: Loss of the last batch of every epoch during training MNIST

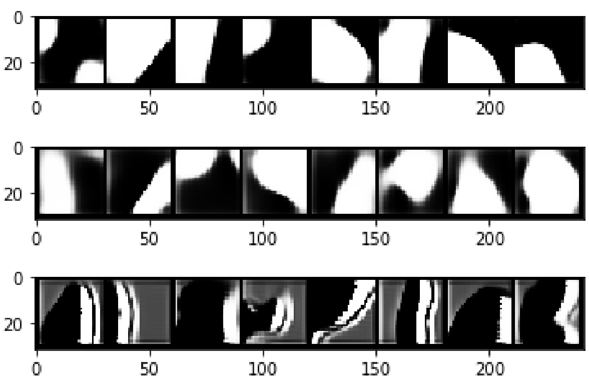

Figure 2.3: OT-GAN with reg=100, $\beta_1 = 0.5$ in Adam optimizer, at epoch 0, 10, 20.

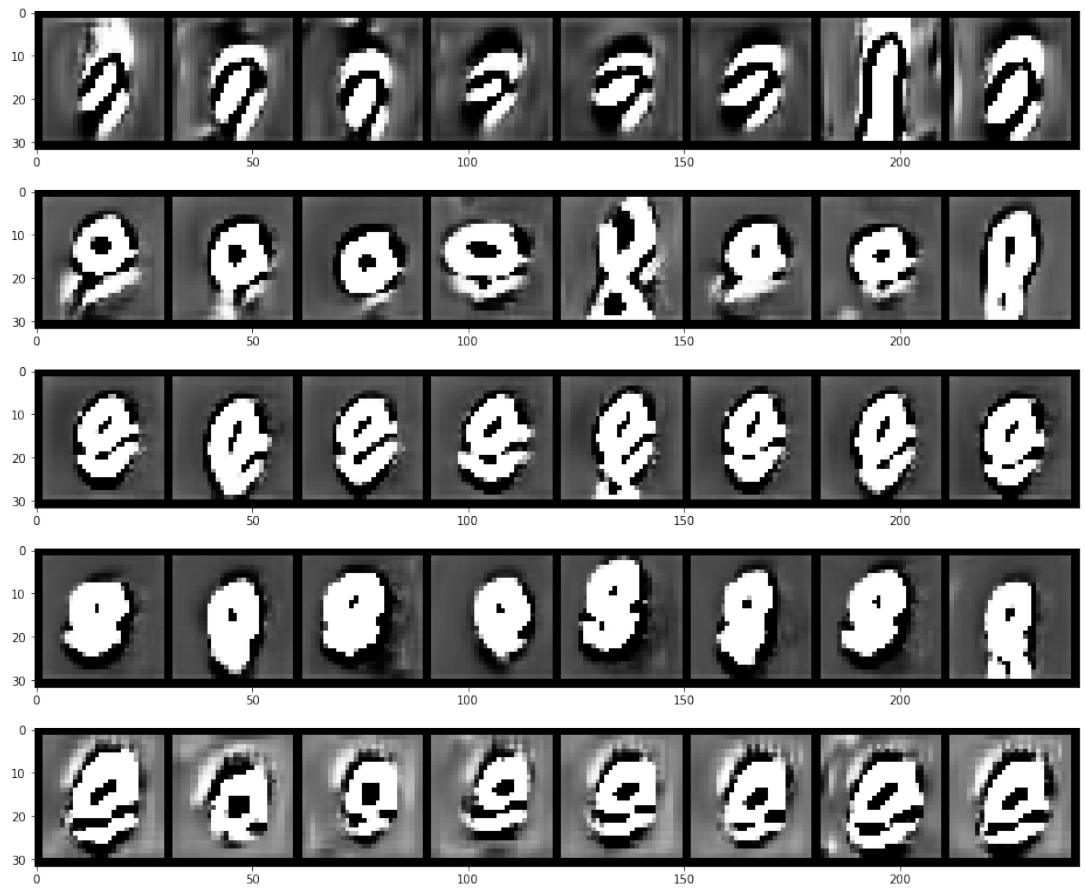

Figure 2.4: Samples generated by OT-GAN generator from 0th to 20 epochs, with entropy penalty = 100.

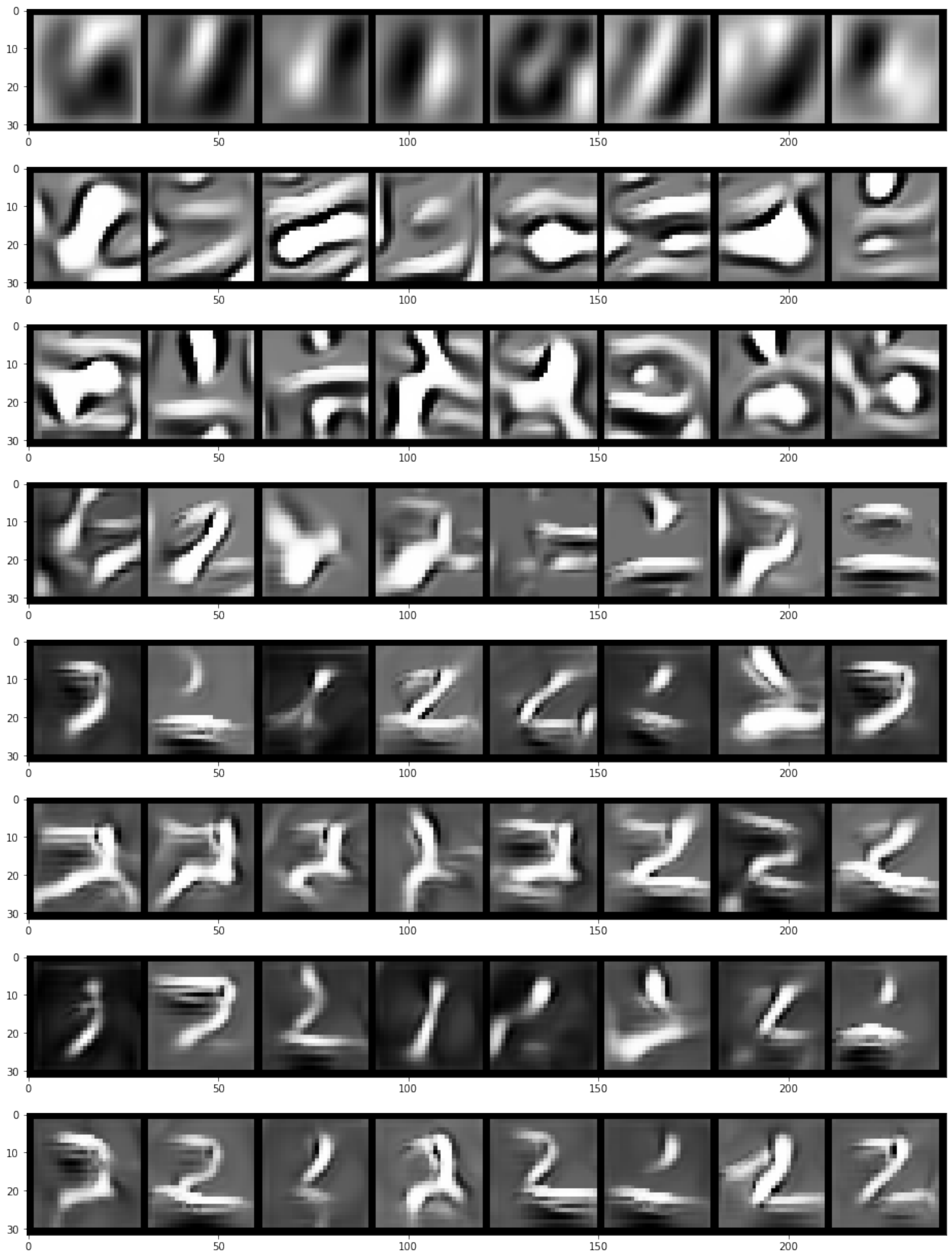
Figure 2.5: Samples generated by OT-GAN generator on MNIST with only label 2 from 0th to 70th epoch.

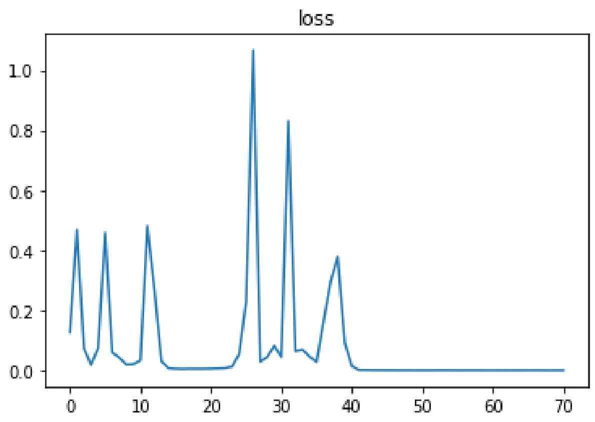

Figure 2.6: Loss of the last batch of every epoch during training MNIST conditional on label 2.

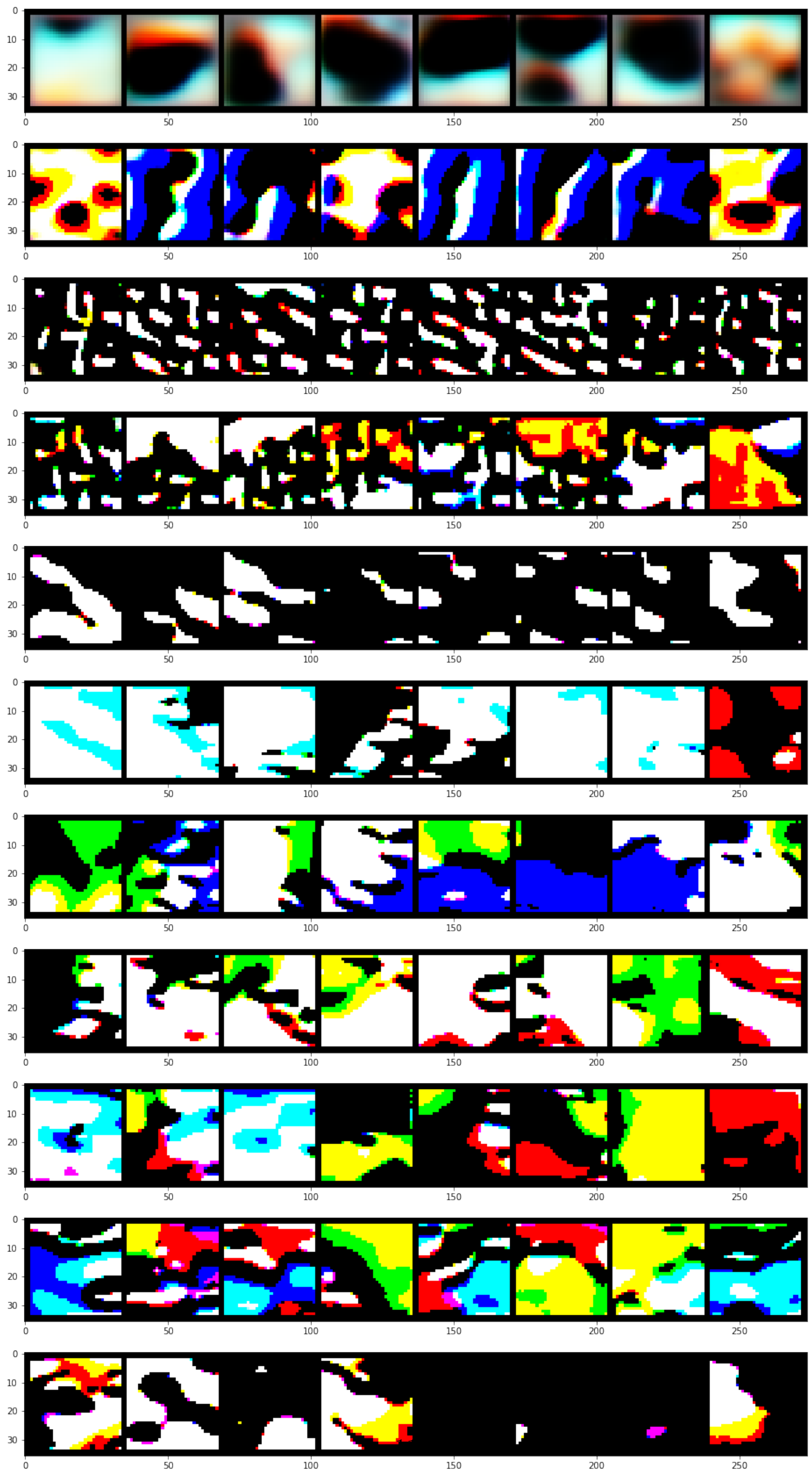

Figure 2.7: Samples generated by OT-GAN generator on CIFAR10, from 0th to 100th epoch.

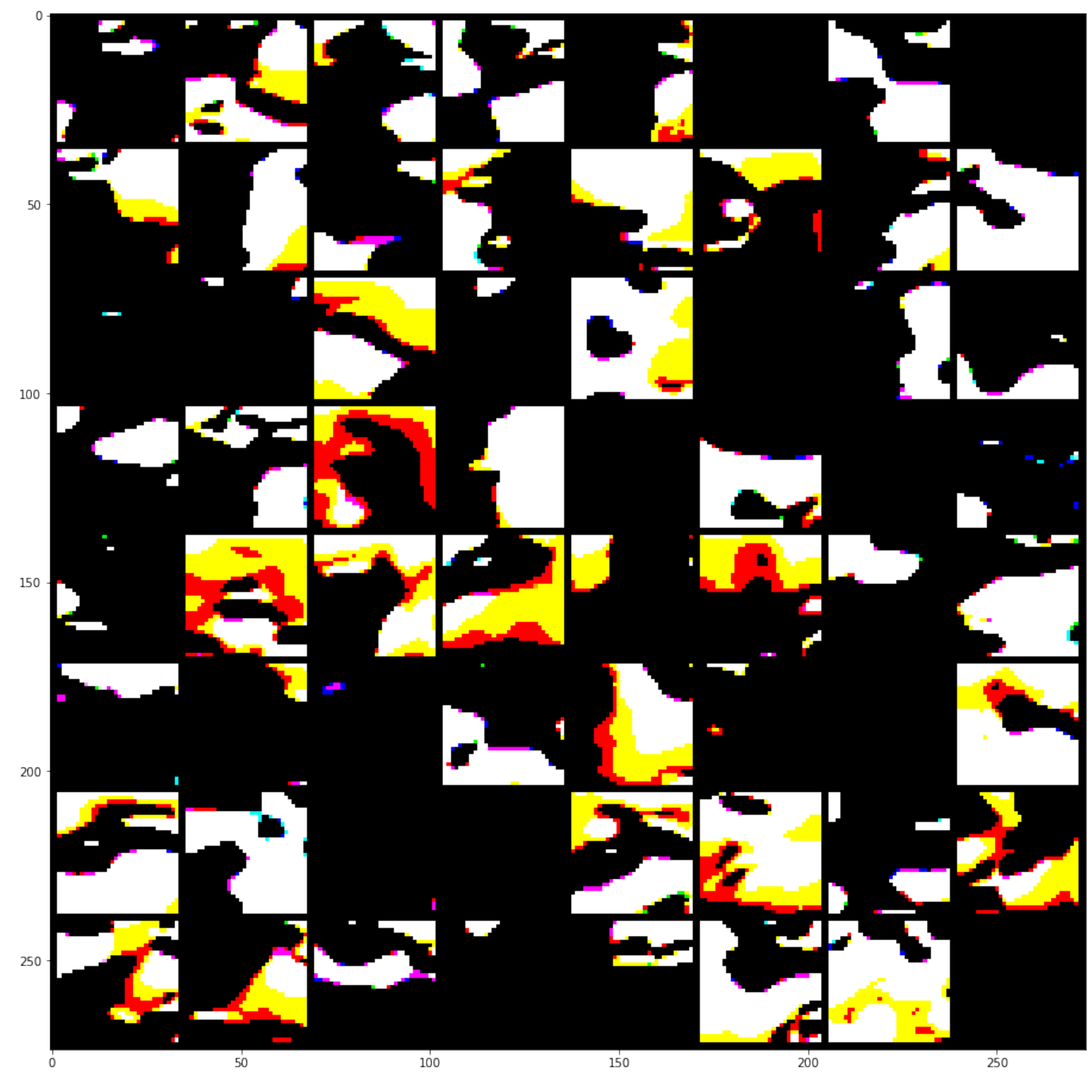
Figure 2.8: On the 100th epoch, some generated samples.

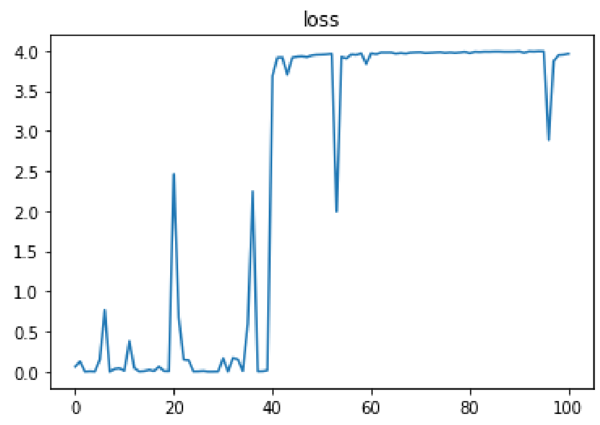

Figure 2.9: Loss of the last batch of every epoch during training CIFAR10.

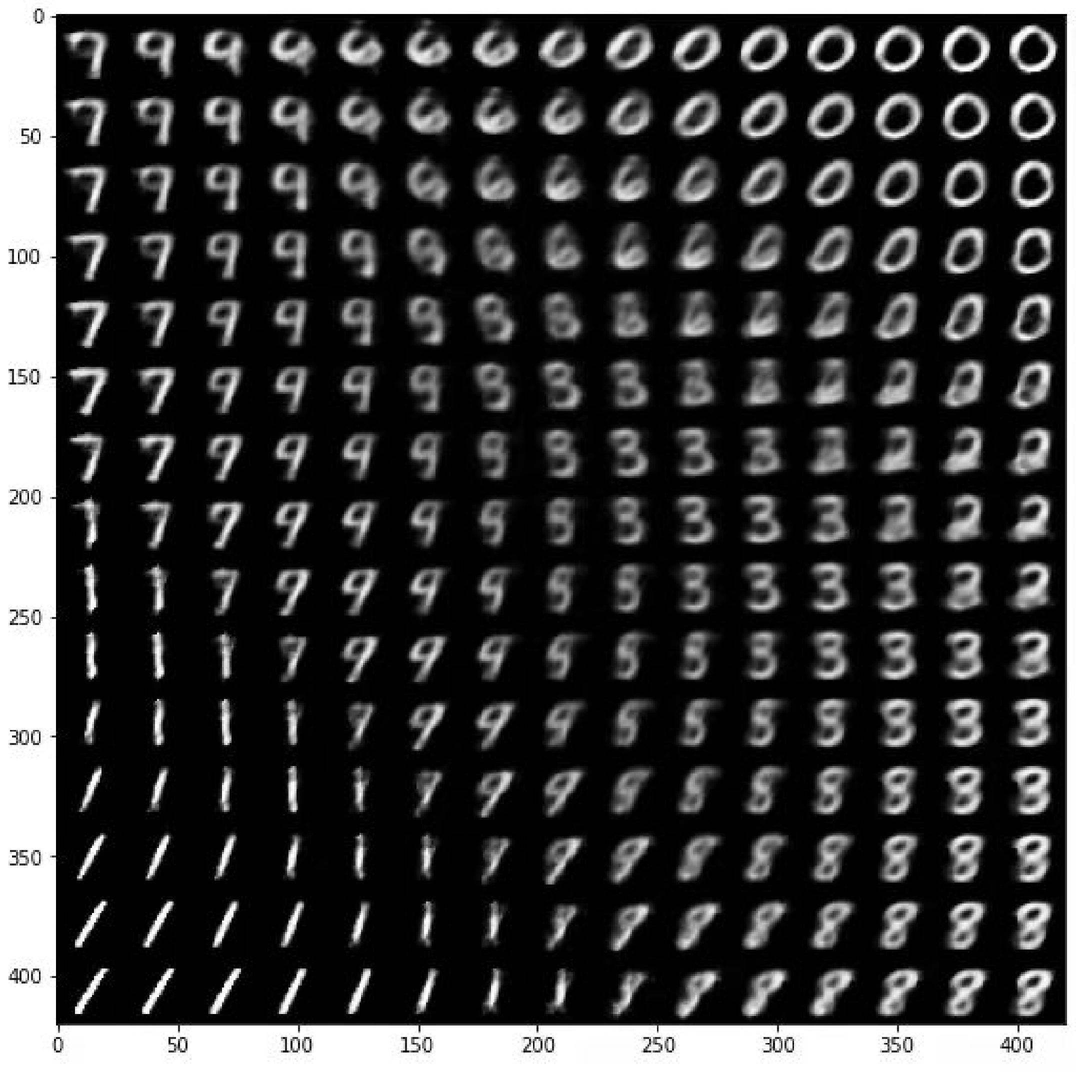

Figure 2.10: Sinkhorn-Autodiff at 10 epoch

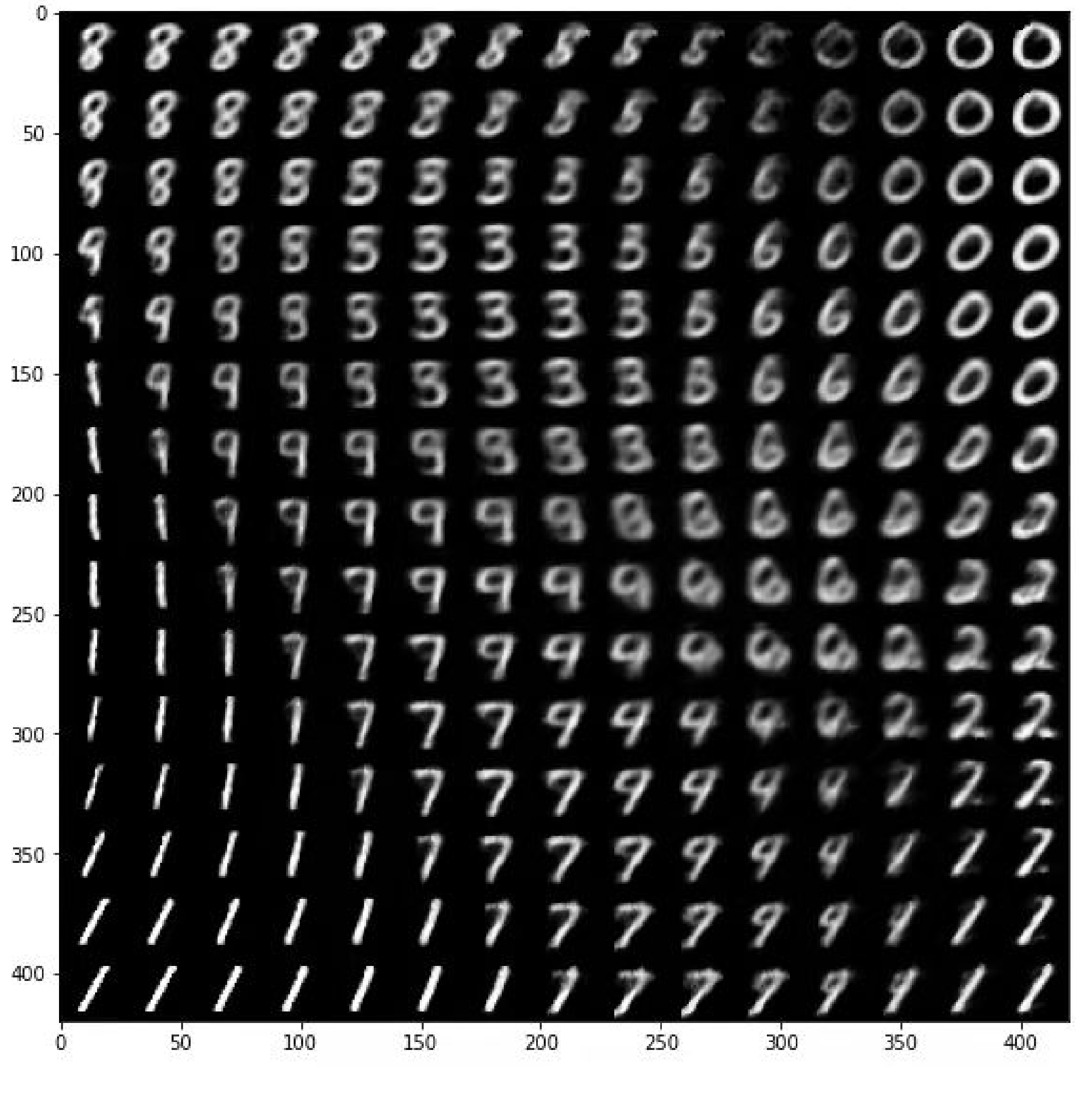

Figure 2.11: Sinkhorn-Autodiff at 20 epoch

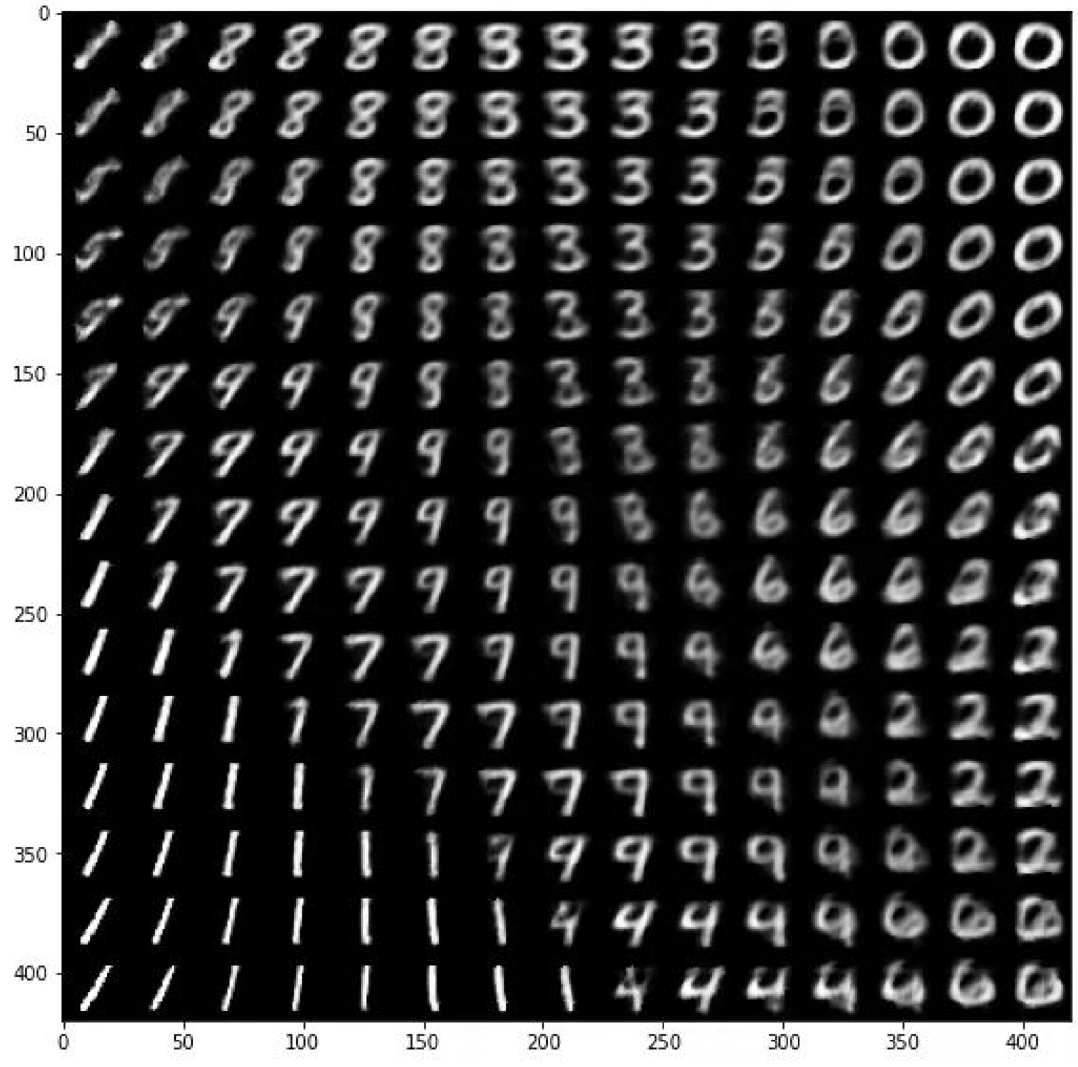

Figure 2.12: Sinkhorn-Autodiff at 40 epoch

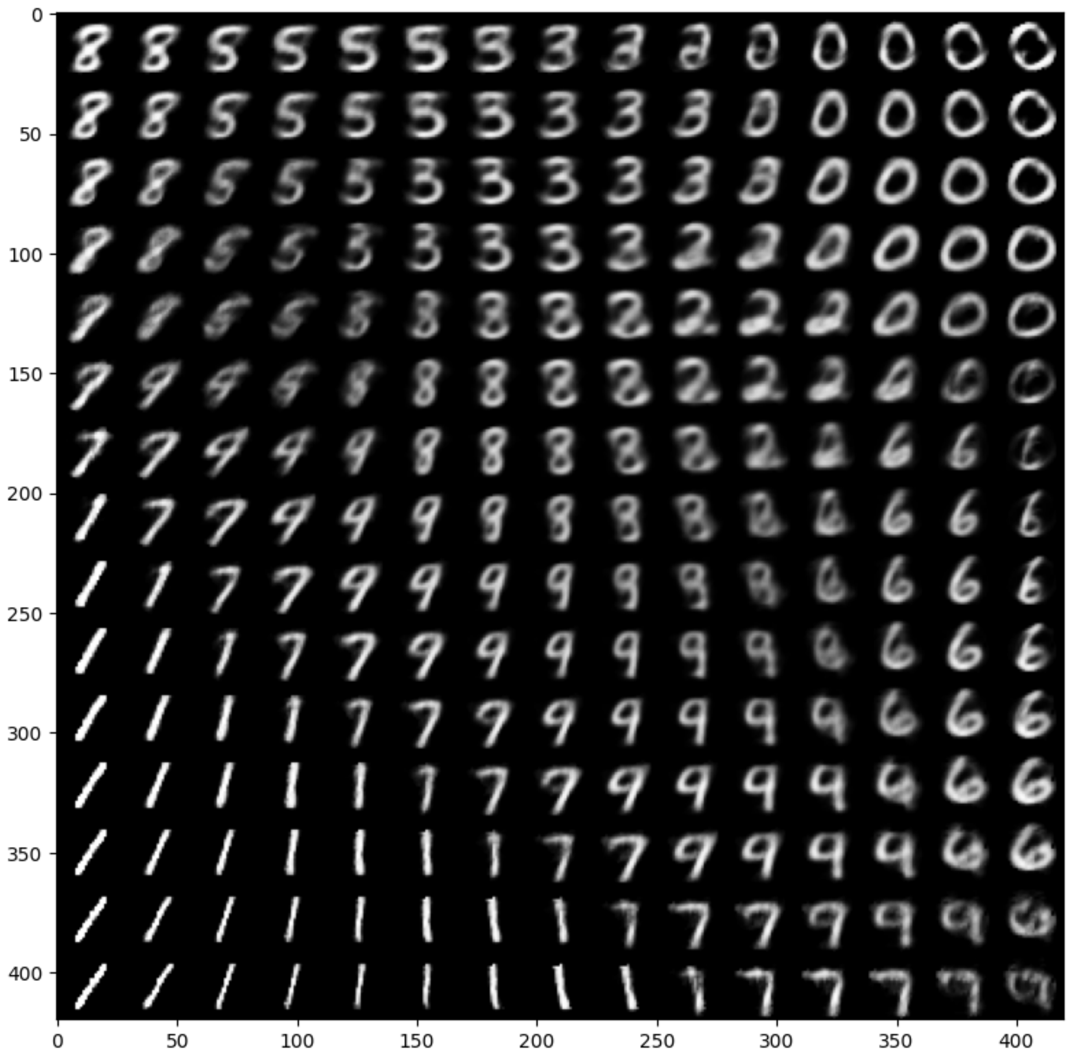

Figure 2.13: Sinkhorn-Autodiff at 70 epoch

3 Conclusion

We introduced WGAN and its variants, and compared the performance of Sinkhorn-Diff and OT-GAN. A better tunning of GAN models is expected, but we did not explore that direction too much because of the time of training is considerable. We expect that there should be a changing of learning rate according to the evolution of loss, so that the loss of adversarial learning process will be oscillating in a reasonable interval.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[2] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 214–223.

[3] A. Genevay, G. Peyré, and M. Cuturi, "Learning generative models with sinkhorn divergences," 2017. [Online]. Available: https://arxiv.org/abs/1706.00292

[4] T. Salimans, H. Zhang, A. Radford, and D. Metaxas, "Improving gans using optimal transport," *arXiv preprint arXiv:1803.05573*, 2018.