

WASSERSTEIN AUTO-ENCODERS

MARZIEH MEHDIZADEH, JOSEPH D VIVIANO, JOHNATHAN GUYMONT

1. PAPER SUMMARY

1.1. **Motivation.** (What is the problem)

In this paper, the authors introduce a new efficient algorithm called Wasserstein Auto-Encoders (WAE) to build a generative model of data distribution. This generative model is based on the optimal transport (OT) finding the minimum Wasserstein distance between the true data probability distribution (unknown distribution) P_X and the latent variable distribution P_G . In other words, WAE minimizes a penalized form of the Wasserstein distance between P_X and P_G which leads to a regularizer that is different from the regularizer in VAE. This regularizer helps the posterior $P_G(X|Z)$ distribution of data points $X \in \mathcal{X}$ to match the prior P_Z of latent codes $Z \in \mathcal{Z}$.

1.2. **Proposed approach.** (Analysis)

First we list the main contributions of this paper as below:

- (1) The first and the main contribution is the family of Wasserstein Auto-Encoders (WAE) defined as above. The loss function of WAE algorithm is a combination of two following important terms:

Loss = reconstruction cost + a regularizer penalizing a discrepancy between distributions

The cost function is the squared cost and the regularizer is denoted as $D_Z(P_Z, Q_Z)$, where Q_Z is the distribution of encoded data points defined by $Q_Z := \mathbb{E}_{P_X} [Q(Z|X)]$.

- (2) We use two MNIST and CelebA datasets to evaluate the WAE algorithm and we will observe that the generated samples are less blurry than VAE's generated samples.
- (3) Two different forms of regularizer $D_Z(P_Z, Q_Z)$ will be introduced: One is based on the adversarial training in latent space \mathcal{Z} (Like GANs) so we denote it the **GAN-based** \mathcal{D}_Z and the other form is based on the maximum mean discrepancy denoted as **MMD-based** \mathcal{D}_Z .
- (4) Theoretical part containing a theorem that helps finding objective function for WAE.

How WAE works:

The WAE method tries to minimize the optimal transport cost $W_c(P_X, P_G)$ such that the decoder and encoder have the following tasks:

Decoder: It tries to reconstruct the encoded training examples with a good accuracy by using the cost function c , defined by $c(x, y) = \|x - y\|_2^2$.

Encoder: It simultaneously does two tasks: It tries to match the encoded distribution of training examples Q_z to the prior P_Z measured by using one of the form of $\mathcal{D}_Z(Q_Z, P_Z)$ introduced above, and at the same time it makes sure that the latent codes $Z \in \mathcal{Z}$ are learning enough to reconstruct the encoded training examples. In ?? we give more explanation that how WAE is more efficient than VAE.

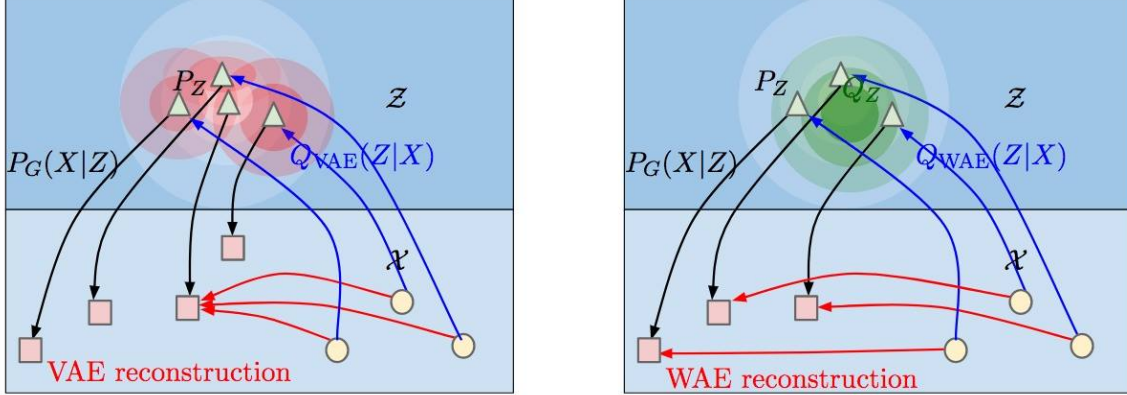


FIGURE 1. In these figures circles true data points X for data space \mathcal{X} , triangles are the latent points Z from space \mathcal{Z} , squares represent the generated samples Y and arrows show the different conditional distributions. In both cases VAE (left) and WAE (right) we have to minimize the $c(x, y)$ and $\mathcal{D}_Z(P_Z, Q_Z)$ but there is a big difference that as we observe in left, VAE tries to encourage every single $Q(Z|X = x)$ shown by the red balls to match P_Z shown by the white ball, but the problem is that the red balls have intersection which leads the blurriness problem to the reconstructed results. On the other hand, WAE forces the expectation $Q_Z = \mathbb{E}_{P_X}[Q(Z|X)]$ with the green balls to match P_Z shown by white balls. If we define $\Gamma_{VAE}(Y|X) = P_G(Y|Z) + Q_{VAE}(Z|X)$ and $\Gamma_{WAE}(Y|X) = P_G(Y|Z) + Q_{WAE}(Z|X)$ which are shown by the red arrows, then we see that in the left picture the data point X are mapped to a same Y with a high probability while in the right picture data points X are mapped to Y a lower probability.

What is Wasserstein distance? Wasserstein Distance is a measure of the distance between two probability distributions. (It is also called Earth Mover's distance, short for EM distance, because informally it can be interpreted as moving piles of dirt that follow one probability distribution at a minimum cost to follow the other distribution.) The cost is quantified by the amount of dirt moved times the moving distance. We take the minimum one among the costs of all dirt moving solutions as the EM distance. In the definition of Wasserstein distance, the \inf (infimum, also known as greatest lower bound) indicates that we are only interested in the smallest cost.) More formally we define the Wasserstein distance as follows:

$$(1) \quad W_c(P_X, P_G) := \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y) \sim \Gamma} [c(X, Y)],$$

where $c(x, y)$ is any measurable cost function and $\mathcal{P}(X \sim P_X, Y \sim P_G)$ is the set of all possible joint probability distribution between P_X and P_G .

Why Wasserstein distance is better than JS or KL divergence used in VAE?

Even when two distributions are located in lower dimensional manifolds without overlaps, Wasserstein distance can still provide a meaningful and smooth representation of the distance in-between. Let's give an examples: If P and Q are two probability distributions and we define:

$$\forall (x, y) \in P, x = 0, y \sim U(0, 1)$$

$$\forall (x, y) \in Q, x = \theta, 0 \leq \theta \leq 1, y \sim U(0, 1)$$

when $\theta \neq 0$.

$$D_{KL}(P||Q) = \sum_{x=0, y \sim U(0,1)} 1. \log(1/0) = +\infty,$$

$$D_{KL}(Q||P) = \sum_{x=\theta, y \sim U(0,1)} 1. \log(1/0) = +\infty,$$

$$D_{JS}(P, Q) = 1/2 \left(\sum_{X=0, Y \sim U(0,1)} 1. \log \frac{1}{(1/2)} + \sum_{X=\theta, Y \sim U(0,1)} 1. \log \frac{1}{(1/2)} \right) = \log 2$$

and

$$W(P, Q) = |\theta|$$

But when $\theta = 0$, two distributions are fully overlapped:

$$D_{KL}(P||Q) = D_{KL}(Q||P) = D_{JS}(P, Q) = 0,$$

and

$$W(P, Q) = 0 = |\theta|$$

D_{KL} gives us infinity when two distributions are disjoint. The value of D_{JS} has sudden jump, not differentiable at $\theta = 0$. Only Wasserstein metric provides a smooth measure, which is super helpful for a stable learning process using gradient descents.

2. EXPERIMENTS

2.1. Motivation for the experiments. To evaluate any improvement in learning the latent data distribution using the optimal transport-based Wasserstein distance over the standard variational auto encoder approach (making use of the KL-divergence), the authors trained 3 models: a 'stock' VAE, a WAE using the GAN-based Wasserstein penalty, and a WAE using the MDD-based Wasserstein penalty. The motivation was to empirically test whether the Wasserstein distance improves the reconstruction quality observed. The reconstruction analysis and latent space traversal analyses were standard. It was also good that they included a measure of image sharpness and the inception distance scores to quantify the claims made by this paper.

2.2. Reproducing the main results. The reproduction of this paper can be found in the following repo:

<https://github.com/josephdviviano/wae>

These experiments were run on both the celeba and mnist datasets. Note that, due to time restrictions, we only trained our models on 10% of the celeba dataset, and used a small test set of 1000 images.

Both the encoder-decoder architecture as well as the discriminator architecture (when applicable) used the ADAM optimizer. As in the original paper, the learning rate was halved at epoch 30, and then reduced further by a factor of 5 at epoch 50. We trained all models for 80 epochs. We used a convolutional architecture mirroring that used the in the DC-GAN paper. The discriminator was a 4-layer fully connected network. Below, we report the hyperparameters used in each of our 6 analysis. All hyperparameters that are different from those in the original paper (out of necessity to stabilize training) are marked with a *:

NB: lambda refers to the scaling factor on the WAE penalty, scalebullshit refers to the scaling factor on the reconstruction loss and sigma refers to the variance of the latent layer used for sampled noise.

For the two WAE CELEBA runs, we used mean squared error reconstruction loss. For the VAE, we used binary cross entropy loss. The latent dimension was 64 for all analysis:

CELEBA VAE: Batch size = 64, learning rate = 0.0001, lambda = N/A, scalebullshit = 0.05, sigma = 2.

CELEBA WAE-GAN: Batch size = 64, learning rate = 0.0003, lambda = 1, scalebullshit = 0.05, sigma = 2.

CELEBA WAE-MMD: Batch size = 64, learning rate = 0.00001*, lambda = 10*, scalebullshit = 1, sigma = 2.

Notes: We had to turn down lambda and the learning rate, and turn up scalebullshit, in order to stabilize training (i.e., more heavily weight the reconstruction part of the loss). Training of this model was quite unstable and required many attempts.

Surprisingly, we generally had more trouble stabilizing the training for the MNIST data. Again, the two WAE CELEBA runs used mean squared error reconstruction loss, while VAE run used binary cross entropy. The latent dimension was 8 for all analysis:

MNIST VAE: Batch size = 100, learning rate = 0.00001*, lambda = N/A, scalebullshit = 0.05, sigma = 2.

Notes: why can't we get this to work properly?

MNIST WAE-GAN: Batch size = 100, learning rate = 0.001, lambda = 10, scalebullshit = 1, sigma = 2.

Notes: This was relatively easy to train (little instability issues).

MNIST WAE-MDD: Batch size = 100, learning rate = 0.00001*, scalebullshit = 1, sigma = 2.

Notes: Extremely finicky, even worse than the CELEBA dataset. Gradients easily explode, requiring very small learning rates.

Below we show the reconstructed data for all analysis at epoch 80:



FIGURE 2. Test CelebA reconstructions at epoch 80 using the VAE model.

Finally, the authors report the sharpness and inception scores for the CELEBA dataset, which we reproduce from our own results here:

ONLY THING LEFT TO ADD

Due to time constraints, we did not replicate the latent-space traversal analysis. We do not consider it a main result because it does little to compare WAE with VAE, and we ran short on time due to having to debug the unstable training of the MNIST data and the MMD-based analysis.

3. DISCUSSION

We are not critical of the experiments conducted considering the nature of the result. Namely, they present a new kind of generative model, and then compare the performance of this model using standard generative-model measures on stock datasets.



FIGURE 3. Test CelebA reconstructions at epoch 80 using the WAE GAN model.



FIGURE 4. Test CelebA reconstructions at epoch 80 using the WAE MMD model.

We are critical that not many quantitative results are shown. Training curves would have been welcome. There was also very little discussion of training stability, which we found to be an extremely challenging aspect of this replication.

We found multiple details of the implementation in the official source code not reported in the paper. All references below refer to the official code:

<https://github.com/tolstikhin/wae>

- For the reconstruction term of the loss (mean squared error), the authors **scale it by 5%** (line 335 of wae.py). This is obviously crucial, as it weights the reconstruction term of the WAE loss ($\text{RECON} + \text{LAMBDA} * \text{PENALTY}$) quite low compared to how they present the results in the paper. In our experiments, we tried using this scaling factor, lovingly called **SCALEBULLSHIT**.

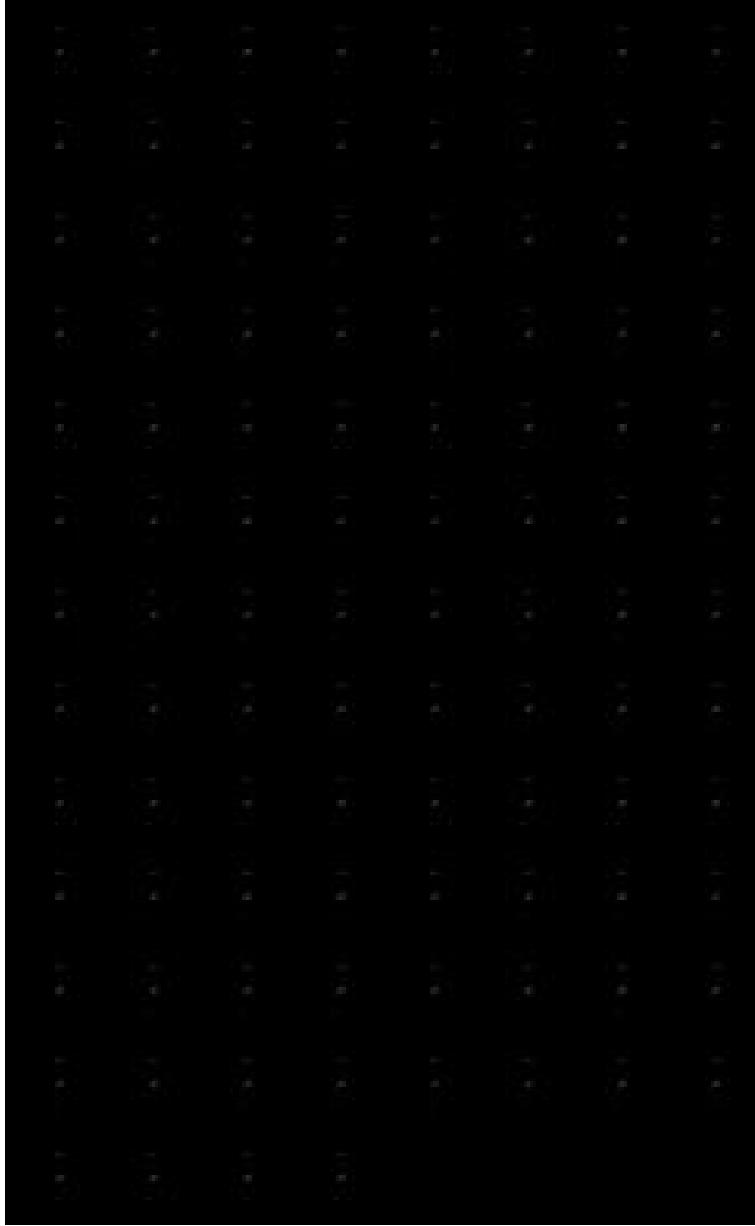


FIGURE 5. Test MNIST reconstructions at epoch 80 using the VAE model.

- For the MMD loss, the MMD value is accumulated times over different scales (line 294 of `wae.py`). We copied this implementation detail, but are unsure of it's relevance. Nonetheless, this implementation detail should be covered in the paper, or at the very least, be justified by a comment in the code.
- We found training to be rather unstable for the MMD method in particular. Training tended to collapse if the mean or standard deviation of the encoded data grew too large, which happened frequently for the MMD method. The authors combat this by clamping the sigma value during reparameterization between -50 and 50, which they

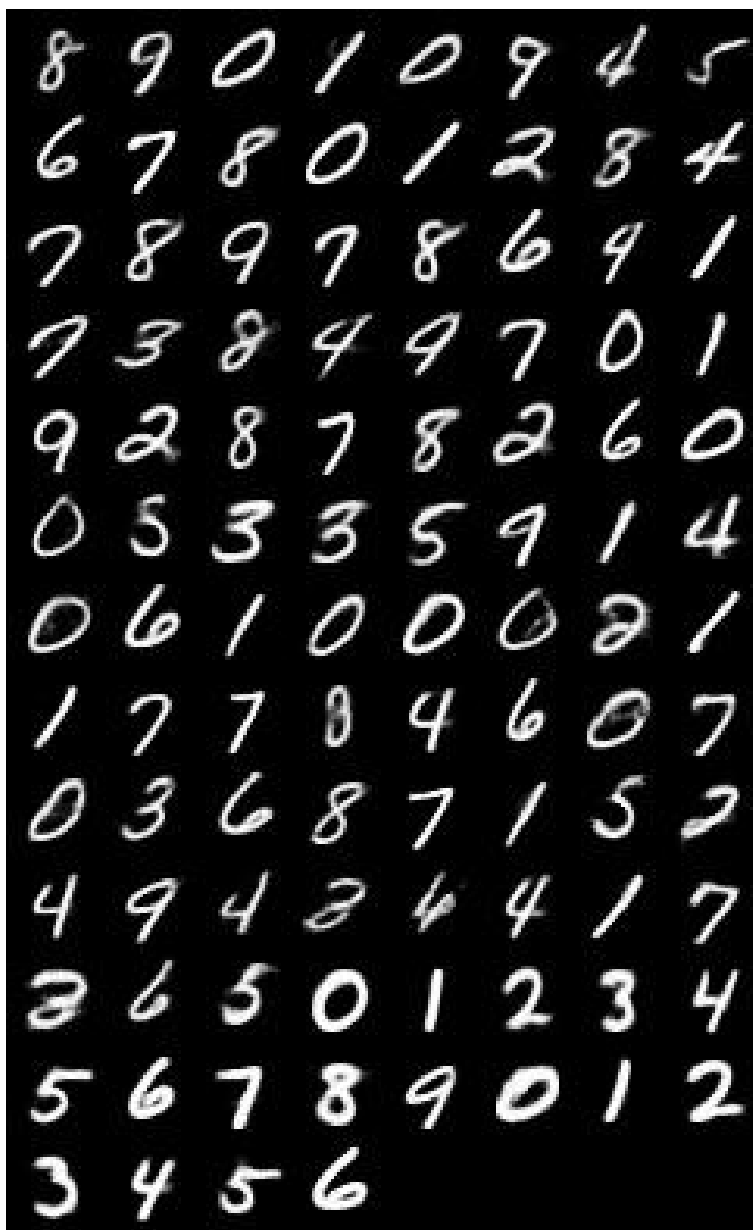


FIGURE 6. Test MNIST reconstructions at epoch 80 using the WAE GAN model.

FIGURE 7. Test MNIST reconstructions at epoch 80 using the WAE MMD model.

do not comment on anywhere in the paper or code. We found this to be a crucial implementation detail, although it did not combat the issue entirely.

- In general, the balance between the reconstruction term and the Wasserstien penalty is extremely sensitive. The authors would have done well to have given a theoretical

intuition as to why this might be true as well as practical recommendations to mitigate this issue when trying to implement the WAE yourself.

Our major concern is the large difference in performance observed between using **WAE-GAN**, which we found to be both stable and produce very good results, and **WAE-MMD**, which we found to be extremely finicky and only sometimes produce reasonable results after fastidious hyper-parameter fine-tuning. The relationship between the reconstruction term and the WAE penalty is really evident when using the MMD method, but the authors don't give any explanation as to why either method would perform differently. Based on our experiments, we find it very hard to believe that the authors did not encounter training instabilities when using the MMD method, and would have liked to see the authors devote some attention to explaining the real-world differences between these two approaches to estimating the Wasserstein distance.

DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE UNIVERSITÉ DE MONTRÉAL PAVILLON ANDRÉ-AISENSTADT, DIRO.