Convergence of SGD for Training Neural Networks with Sliced Wasserstein Losses - Report

Franck Laborde
franck.laborde@ens-paris-saclay.fr
Killian Steunou
killian.steunou@ens-paris-saclay.fr
March 20, 2025

Abstract

In this report, we verify the theoretical results of [1] on the convergence of neural networks trained with the sliced Wasserstein distance by generating 2D data points and FashionMNIST images using the proposed algorithm. We also study and experiment the proposed alternative optimizer to SGD called Noise Projected SGD (NPSGD). Our code is available at https://github.com/francklaborde/Generative-modeling-project.

1 Introduction

Generative neural networks require careful selection of loss functions to effectively measure how closely generated data matches a target distribution. Wasserstein-based metrics are widely used due to their strong theoretical foundation, but computing the Wasserstein distance directly is often impractical in high-dimensional spaces, an issue known as the curse of dimensionality.
The Sliced Wasserstein (SW) distance offers an efficient alternative, defined as:

$$\mathrm{SW}_2^2(\mathrm{z},\mathrm{y}) = \int_{\theta\in\mathbb{R}^{d-1}} \mathrm{W}_2^2\left(P_\theta\#\mathrm{z}, P_\theta\#\mathrm{y}\right)\mathrm{d}\mathfrak{D}(\theta),$$

By computing along random one-dimensional projections, the SW distance significantly reduces computational complexity. The paper [1] investigates the theoretical aspects of training generative neural networks using Stochastic Gradient Descent (SGD) with the SW distance. Its primary contribution is to establish rigorous theoretical guarantees regarding the convergence behavior of SGD within this framework. In a first part we explain these theoretical results from a practical point of view, and then perform various practical experiments to verify the convergence on simple 2D data and larger-scale image generation.

2 Main Findings

2.1 Sliced Wasserstein Distance

As mentioned before, the Wasserstein distance suffers from the curse of dimensionality, in the sense that the sample complexity for n samples in dimension d is of the order $\mathcal{O}(n^{1/d})$. On top of that, the cost of computing this distance has forced researchers to find a cheaper alternative. One of those alternatives is the Sliced Wasserstein Distance (SWD), which consists of computing the 1D Wasserstein distance between projections of input measures, and averaging over the projections.

Mathematically, given a measure x in $\mathbb{R}^d$, a projection onto a one-dimensional subspace is performed via the push-forward operation: $P_\theta : x \mapsto \theta^T x$ where θ is a unit vector sampled from the unit sphere.

Once projected onto a line, the computation of the Wasserstein distance between measures becomes significantly simpler.

In the practical case, the measures are discrete in $\mathbb{R}^d$. Let two discrete measures $\gamma_X = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $\gamma_Y = \frac{1}{n}\sum_{i=1}^n \delta_{y_i}$ where $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$. The push-forward operator is thus given by : $P_\theta \# X = \frac{1}{n}\sum_i \delta_{\theta^T x_i}$ and $P_\theta \# Y = \frac{1}{n}\sum_i \delta_{\theta^T y_i}$. The 2-Wasserstein distance between their projections can be computed by sorting their supports: let χ a permutation sorting $(\theta^T x_1, , \theta^T x_2 \dots, \theta^T x_n)$, and τ a permutation sorting $(\theta^T y_1, , \theta^T y_2 \dots, \theta^T y_n)$, one has the simple expression :

$$W_2^2(P_\theta \# X, P_\theta \# Y) = \frac{1}{n}\sum_{i=1}^n (\theta^T x_{\chi(i)} - \theta^T y_{\tau(i)})^2$$

The SW distance is the expectation of this quantity with respect to $\theta \sim \sigma$, i.e. uniform on the sphere:

$$SW_2^2(\gamma_X, \gamma_Y) = \mathbb{E}_{\theta\sim\sigma}[W_2^2(P_\theta \# X, P_\theta \# Y)]$$

We define a neural network as

$$T : \left\{ \begin{array}{ccc} \mathbb{R}^{d_u} \times \mathbb{R}^{d_x} & \longrightarrow & \mathbb{R}^{d_y} \\ (u, x) & \longmapsto & T_u(x) := T(u, x) \end{array} \right. \tag{1}$$

Sliced-Wasserstein generative models are then trained using Algorithm 1.

Algorithm 1 Training a NN on the SW loss with Stochastic Gradient Descent.

Input: Learning rate $\alpha > 0$.
Input: probability distributions $x \in \mathcal{P}_2(\mathbb{R}^{d_x})$ and $y \in \mathcal{P}_2(\mathbb{R}^{d_y})$.
  Initialization: Draw $u^{(0)} \in \mathbb{R}^{d_u}$
  for $t \in [\![0, T_{\max} - 1]\!]$ **do**
    Draw $\theta^{(t+1)} \sim,\ X^{(t+1)} \sim x^{\otimes n},\ Y^{(t+1)} \sim y^{\otimes n}$.
    SGD update: $u^{(t+1)} = u^{(t)} - \alpha \left[\frac{\partial}{\partial u} \mathrm{W}_2^2(P_{\theta^{(t+1)}} \# T_u \# \gamma_{X^{(t+1)}}, P_{\theta^{(t+1)}} \# \gamma_{Y^{(t+1)}}) \right]_{u=u^{(t)}}$
  end for

We define

$$f := \left\{ \begin{array}{ccc} \mathbb{R}^{d_u} \times \mathbb{R}^{n\times d_x} \times \mathbb{R}^{n\times d_y} \times \mathbb{S}^{d_y-1} & \longrightarrow & \mathbb{R}^{d_y} \\ (u, X, Y, \theta) & \longmapsto & \mathrm{W}_2^2(P_\theta \# T_u \# \gamma_X, P_\theta \# \gamma_Y) \end{array} \right. \tag{2}$$

With this definition for f, we have the final loss formulation

$$F(u) = \mathbb{E}_{(X,Y,\theta)\sim z}\left[f(u, X, Y, \theta)\right] = \mathbb{E}_{(X,Y)\sim x^{\otimes n}\otimes y^{\otimes n}}\left[\mathrm{SW}_2^2(T_u \# \gamma_X, \gamma_Y)\right], \tag{3}$$

They show that under Assumptions 1, 2 and 3, the losses f and F are locally Lipschitz. This result is essential in proving the convergence results. We reformulate all assumptions informally and under a practical point of view for clarity. For mathematical formal details, please refer to [1].

Assumption 1 *The neural network must be piecewise smooth with respect to its parameters, where the pieces depend on the input data. This is an assumption about the activation functions of the neural network, that must be $\mathcal{C}^2$-smooth (sigmoid, hyperbolic tangent, softplus) or locally Lipschitz semi-algebraic (ReLU, LeakyReLU).*

Assumption 2 *The neural network must be Lipschitz continuous with respect to its parameters and input data.*

Assumption 3 *The input and target data distributions must be supported on bounded sets.*

The following assumptions are used to prove the convergence of F.

Assumption 4 *The neural network must be bounded with respect to its parameters. This is the case for a neural network with bounded activation functions. Multiplying the output of the neural network by an indicator function on the network's parameters (of the form $\mathbb{1}_{B(0,R)}(u)$) makes it satisfy this assumption too.*

Assumption 5 *The neural network's second derivatives with respect to its parameters are bounded.*

2.2 Convergence of Interpolated SGD Trajectories Under Practical Assumptions

The authors demonstrate that under assumptions 1, 2, 3, 4 and 5, the trajectories of constant-step SGD schemes on the SW loss converge to the set of sub-gradient flow solutions as the step size (learning rate) decreases. They show that the interpolated SGD trajectories $u_\alpha(s)$ approach the set of solutions of the differential inclusion:

$$\dot{u}(s) \in -\partial_C F(u(s)),$$

where $\partial_C F$ denotes the Clarke sub-differential of the loss function F. This implies that with small learning rates, SGD trajectories approximate sub-gradient flows, which themselves converge to critical points of F.

2.3 Stronger Convergence with Noised Projected SGD Under Stricter Assumptions

A stronger convergence result is established for a variant of SGD that includes additive noise and projection onto a bounded domain. Given a radius $r > 0$, the SGD scheme is restricted to $u \in \overline{B}(0, r) := B_r$, by performing projected SGD. They also add a noise at each step. The noised projected SGD update is given by:

$$u^{(t+1)} = \pi_r \left(u^{(t)} - \alpha \varphi \left(u^{(t)}, X^{(t+1)}, Y^{(t+1)}, \theta^{(t+1)} \right) + \alpha a \varepsilon^{(t+1)} \right),$$

where $\pi_r : \mathbb{R}^u \to B_r$ denotes the orthogonal projection onto the ball $B_r := \overline{B}(0, r)$, and $a\varepsilon^{(t+1)}$ is the additive noise, standard Gaussian. The authors prove that the long-run limits of these trajectories converge to a set of generalized critical points of the loss function, defined as:

$$\mathcal{C}_r = \left\{ u \in \mathbb{R}^{d_u} \mid 0 \in -\partial_C F(u) - \mathcal{N}_r(u) \right\},$$

where $\mathcal{N}_r(u)$ is the normal cone of the ball $\bar{B}(0, r)$ at u.

This result requires two additional assumptions (Assumption 6 and 7), that can be stated as follows.

Assumption 6 *The input and target data distributions must be discrete.*

This is the core limitation of this theoretical work. The target distribution is always discrete in practice, but the source distribution is commonly continuous (e.g. Gaussian). Nonetheless, it is stated in [1] that "one may argue that the discrete x can have an arbitrary fixed amount of points, and leverage strong sample complexity results to ascertain that the discretisation is not costly if the number of samples is large enough".

Assumption 7 *The neural network must be path differentiable with respect to its parameters for any input data.*

Assumption 7 holds as soon as the neural network has linear units and typical activations, and is needed to show that F is itself path differentiable with respect to the network's parameters, along with Assumption 2 and Assumption 6.

Overall, these assumptions have few practical limitations and should be verified for most standard neural network architectures.

Target Distribution	F
$\mathcal{N}(5, 2)$	0.01252
two moons	0.00218
swiss roll	0.21308

Table 1: Final F loss reached for different 2D distributions, using SGD.

3 Experiments

Following [2], we have decided to perform experiments at different levels. First, we used the SWD loss with a generative model to learn simple 2D datasets. Then, we tried to use this loss to generate images from FashionMNIST. Finally, we implemented the noised projected SGD to compare under suitable neural networks and input data that satisfy Assumption 1, Assumption 2, Assumption 3, Assumption 4, Assumption 5, Assumption 6, and Assumption 7.

3.1 2D dataset

3.1.1 Stochastic Gradient Descent

The first experiments aim at approximating a 2D distribution like a Gaussian, the *two moons* or the *swiss roll* one (see Figure 1). We us a simple MLP neural network with residual connections that respects Assumption 1, Assumption 2, Assumption 4 and Assumption 5, using ReLU activation and Algorithm 1 with SGD. The network has two hidden layers with 256 neurons each. As input of the network, we used a 2D standard Gaussian distribution. We trained each generative model for 5000 epochs, using 256 samples for each SGD update. We use a base step size of 0.001, decreasing it progressively across epochs. We used 5000 different projection maps θ to approximate the total loss F. The qualitative results are shown in Figure 1.

The results are qualitatively very good, we can clearly see the generated points being distributed very closely to the target distribution, especially for the Gaussian target distribution. The worst case seems to be the two moons. The final losses for each distributions are shown in Table 1.
Interestingly, they are quite different. This loss might not reflect what we expect visually, and since it is computed in a 1-dimensional subspace, it is hard to interpret.

3.1.2 Noise Projected Stochastic Gradient Descent

Using the variant of SGD proposed in [1], with a discrete input distribution[1] to satisfy Assumption 6 (Assumption 7 is already satisfied by our neural network). We use a projection radius $r = 10$, and a noise scale $a = 1$, with 5000 projection maps to compute F. We obtain the results shown in Figure 2 using 2025 samples.
Visually, the models generate points in the same range as the target distribution, approximating its shape, but not following it very closely.
This "failure" can have multiple sources. First of all, [1] does not provide any practical guideline on any parameter. All we know is the step size must decrease to zero by the end of training, and the number of iterations must go to infinity (and assumptions must be verified). In practice we set 5000 epochs, and decrease progressively the step size to 1×10^{-8}, but it is not said how this decrease should be implemented (linearly or not, etc), and how many epochs are required to consider them "infinity". Also, [1] states that "the idea behind projecting is to do so on a very large ball", but what does "very large" mean? The author does not give any scale, and the results seem to vary depending on the chosen radius r. Maybe $r = 10$ is not "very large", maybe it is.
Another aspect is that the discrete distribution chosen may have not enough discrete points, but again, [1] does not provide any idea of the scale of this.

The loss for each model is still not really interpretable, but they are all much higher with NPSGD than with SGD.

The observed discrepancies between visual quality and the measured SW loss highlight a potential misalignment between the loss function and the intended generative task. Since the Sliced Wasserstein

[1]We use 2025 fixed points in a "noisy grid" to generate the distribution. We build a regular grid of points in a 2D square, then add a small uniform noise to each point to make the distribution less trivial.

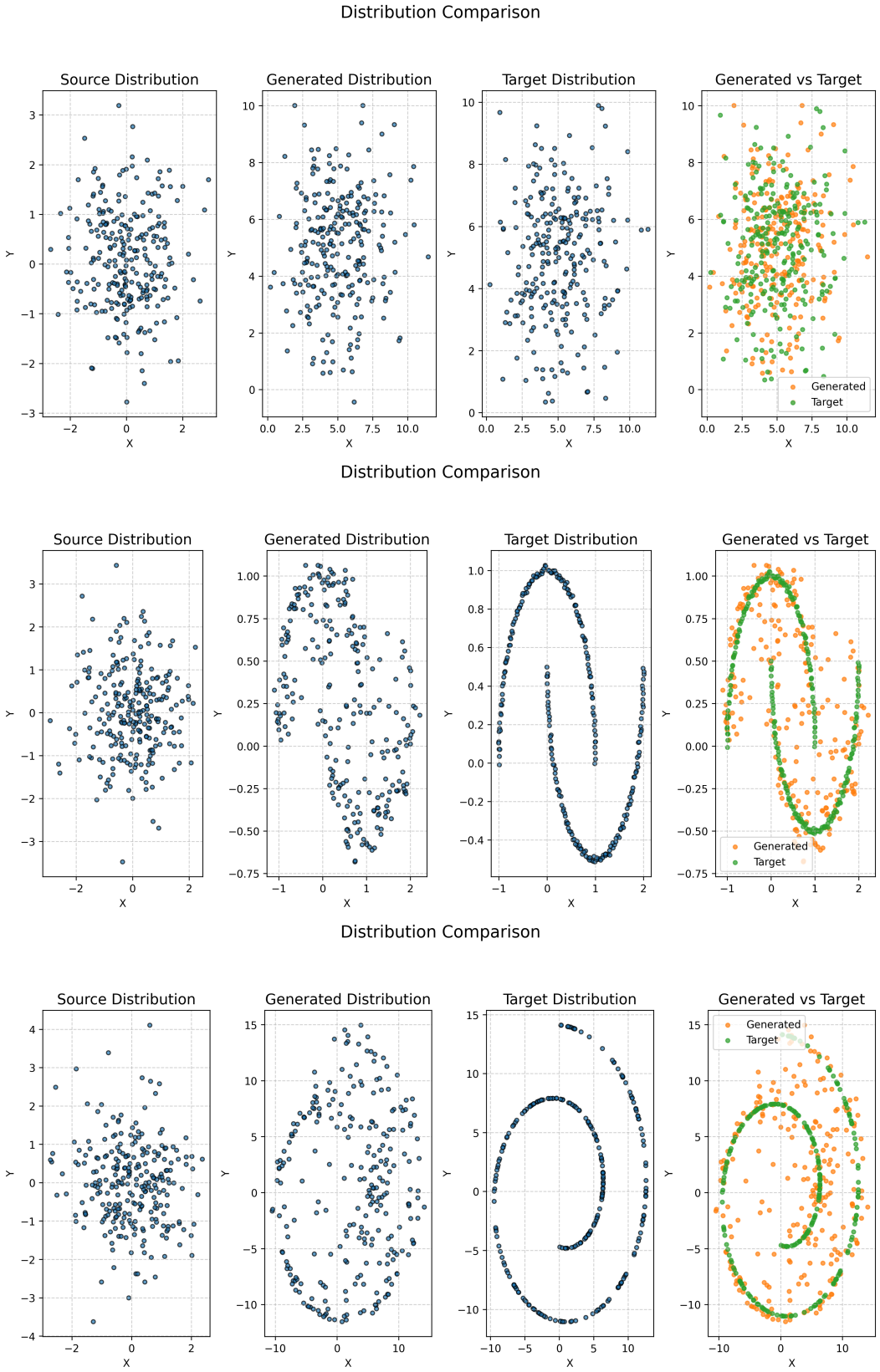


Figure 1: Generated points using Algorithm 1 with SGD to approximate a $\mathcal{N}(5, 2)$ distribution (top), a *two moons* distribution (middle) and a *swiss roll* distribution (bottom).

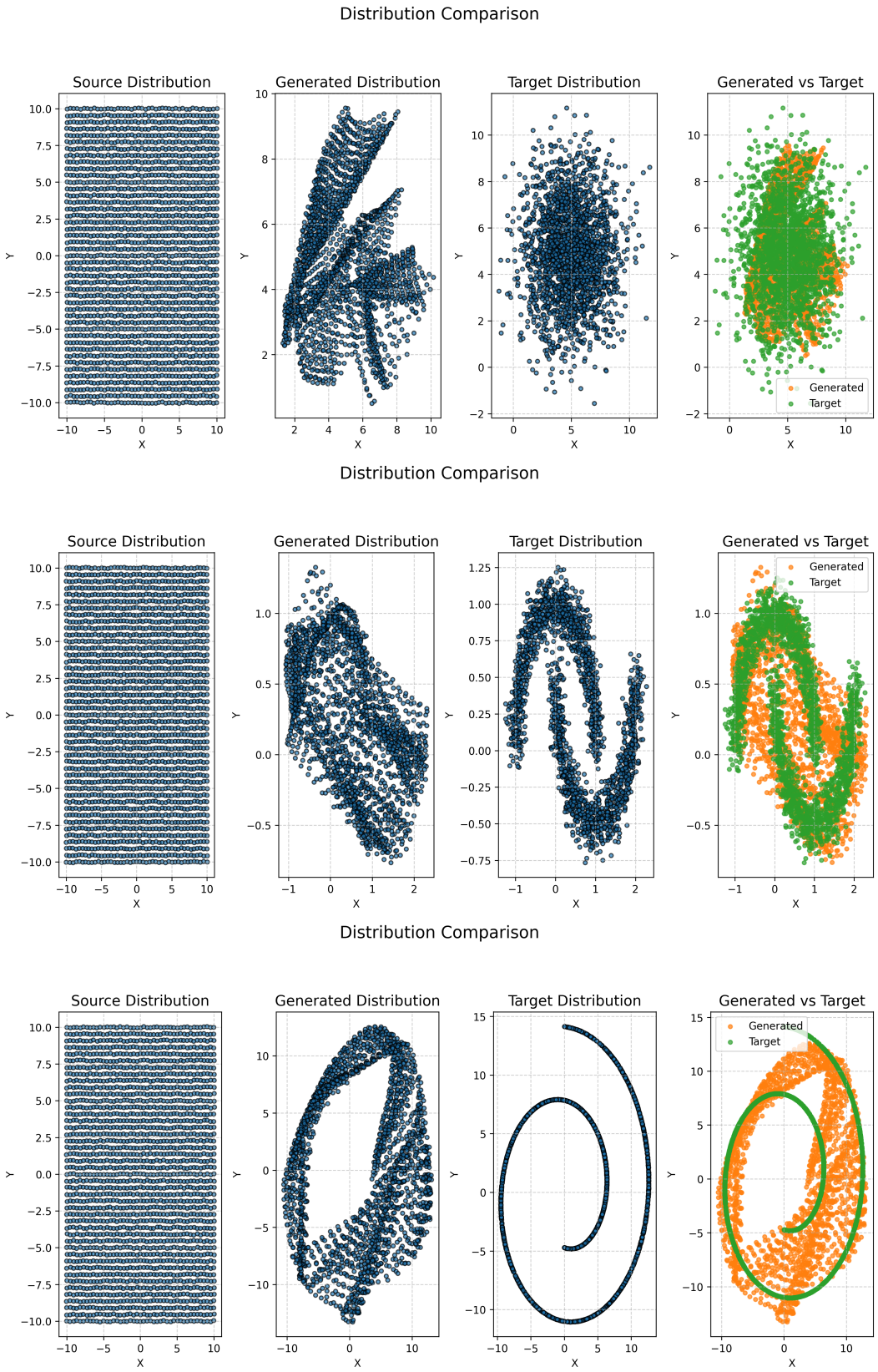


Figure 2: Generated points using Algorithm 1 with NPSGD to approximate a $\mathcal{N}(5, 2)$ distribution (top), a *two moons* distribution (middle) and a *swiss roll* distribution (bottom).

Target Distribution	F
$\mathcal{N}(5, 2)$	0.23250
two moons	0.01142
swiss roll	0.78500

Table 2: Final F loss reached for different 2D distributions, using NPSGD.

(a) Training with SGD (b) Training with NPSGD

Figure 3: Generation of images from FashionMNIST with the SWD loss. We train it for 5000 epoch with a projection over 20 000 samples and a learning rate decreasing from 0.01 to 1e-8.

Distance operates by averaging Wasserstein distances across random one-dimensional projections, the model may sometimes exploit patterns or structures that minimize these projections without accurately capturing the true multidimensional structure of the target distributions. Consequently, a low SW loss does not necessarily imply a good qualitative approximation, as the model might learn distributions that appear correct only when viewed through specific projections.

3.2 FashionMNIST

For the second experiment, we generate images from FashionMNIST [3]. As in the previous part, we use a standard normal distribution as input of the generator, but in 32 dimension inspired by [2]. We used both SGD and NPSGD to compare them. For NPSGD, we discretize the input, to respect Assumption 6. For the parameters, we use a radius $r = 10$ and a noise scale $a = 0.001$.

The generator used maps points from the input space (32-dimensional Gaussian) to the image space (a 784-D vector, reshaped for visualization as a 28×28 image) through convolutional transpose layers. We trained it on 5000 epochs with a learning rate that goes from 0.001 and decreases as the training goes on. Finally, we projected the input on 20000 vectors to compute F. Tbhe images generated with this training for both SGD and NPSGD are shown in the Figure 3.

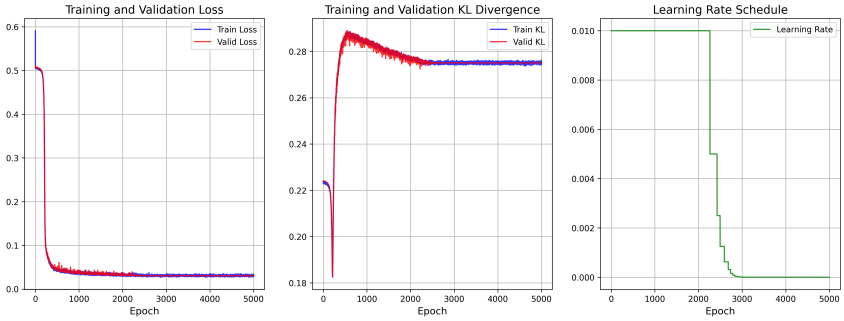
Even though the images generated are blurry, the images generated can be recognized. With more epochs and more vectors to project onto, the quality is not improved that much (see Figure 5 and Figure 6 in the Appendix A). This may be due to the generator we use that is probably too shallow and is not able to generate more accurate images. However, we adapted the network to the computational resources that we could use. The difference between the network trained with SGD and the one trained with NPSGD is hard to discern as the images generated are similar.

We also study the evolution of the training loss, meaning the sliced Wasserstein distance, between both distributions depending on the optimizer used, shown in Figure 4. The final losses are summarized in Table 3. As one can see, the losses are similar whether it be for their values or their behavior, as expected because generated images are really similar.

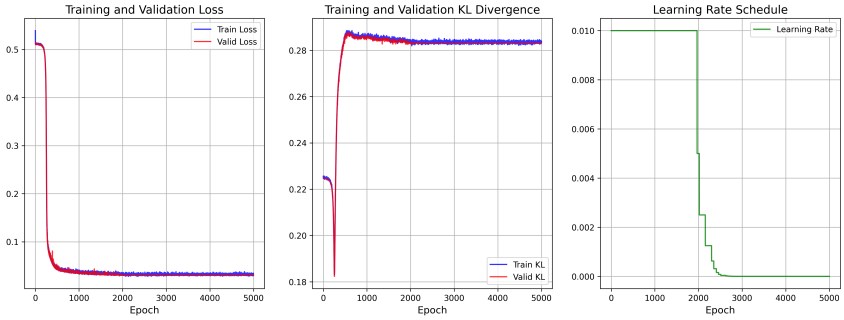
Finally, we try another architecture for the generator using residual connections, as we use this kind of network in the 2D datasets, to study the difference with the "basic" generator. Unfortunately, this

Optimizer	F
SGD	0.02946
NPSGD	0.03135

Table 3: Final loss for both optimizers used on generator for FashionMNIST.



(a) Optimizer SGD



(b) Optimizer NPSGD

Figure 4: Evolution of the loss, the KL divergence and the learning rate for the training and validation data on FashionMNIST.

network did not perform as expected so we will not discuss it in this report (the results from this network are shown in the Appendix A in Figure 7).

4 Conclusion, Limitations and Perspectives

The paper successfully bridges the gap between theoretical and practical observations of SGD convergence in the context of SW losses. It provides convergence guarantees under realistic assumptions and suggests future work to generalize these results to non-discrete input measures and other variants of the SW distance.

Our practical implementation confirms the convergence of SGD for a generative model using the Sliced Wasserstein Distance, especially for generating 2D distributions. However, convergence of the Noise Projected SGD variant appears to require more iterations to be empirically verified, and additional experiments on the learning rate (step size) schedule, the projection radius r, or the noise scale a would be required to verify the theory. Such experiments are time-consuming: in practice the computation of the SW loss F requires many projections to get good results, which is the main computational bottleneck of the method.

Concerning image generation, we found little visual difference between images produced with SGD or NPSGD, but the results are encouraging. However, it is important to acknowledge a potential misalignment between the SW loss function and the generative task, as the loss function relies on averaging Wasserstein distances over one-dimensional projections. This can lead to the model exploiting patterns or structures that minimize the loss in projection but fail to accurately represent the true multidimensional structure of the target distribution.

The core theoretical limitation of this work is the necessity of Assumption 6, requiring discrete

input measures to ensure path differentiability. Removing this assumption and generalizing results to continuous or countably supported input measures remains challenging due to technical difficulties involving the stability of path differentiability through integration. Future work should specifically address this challenge, potentially by establishing new theoretical tools or leveraging existing results in functional analysis and differentiability theory to remove this assumption and broaden the applicability of the convergence results.

References

[1] Eloi Tanguy. Convergence of sgd for training neural networks with sliced wasserstein losses, 2024.

[2] Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

A More graphs for FashionMNIST

Figure 5: Images generated after the same training than the images displayed on Figure 3 with SGD but with 10000 epochs instead of 5000.

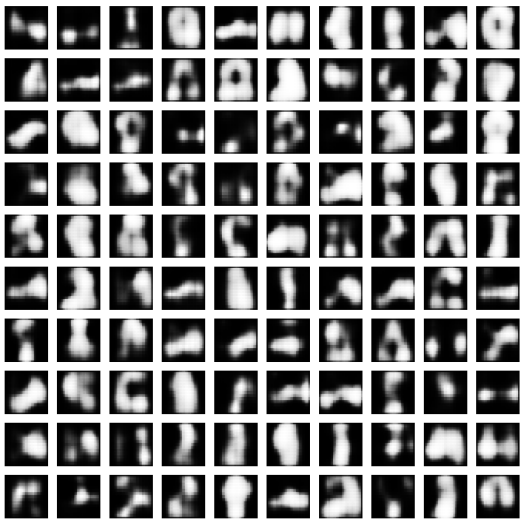


Figure 6: Images generated after the same training than the images displayed on Figure 3 with NPSGD but with a projection over 40000 vectors instead of 20000.

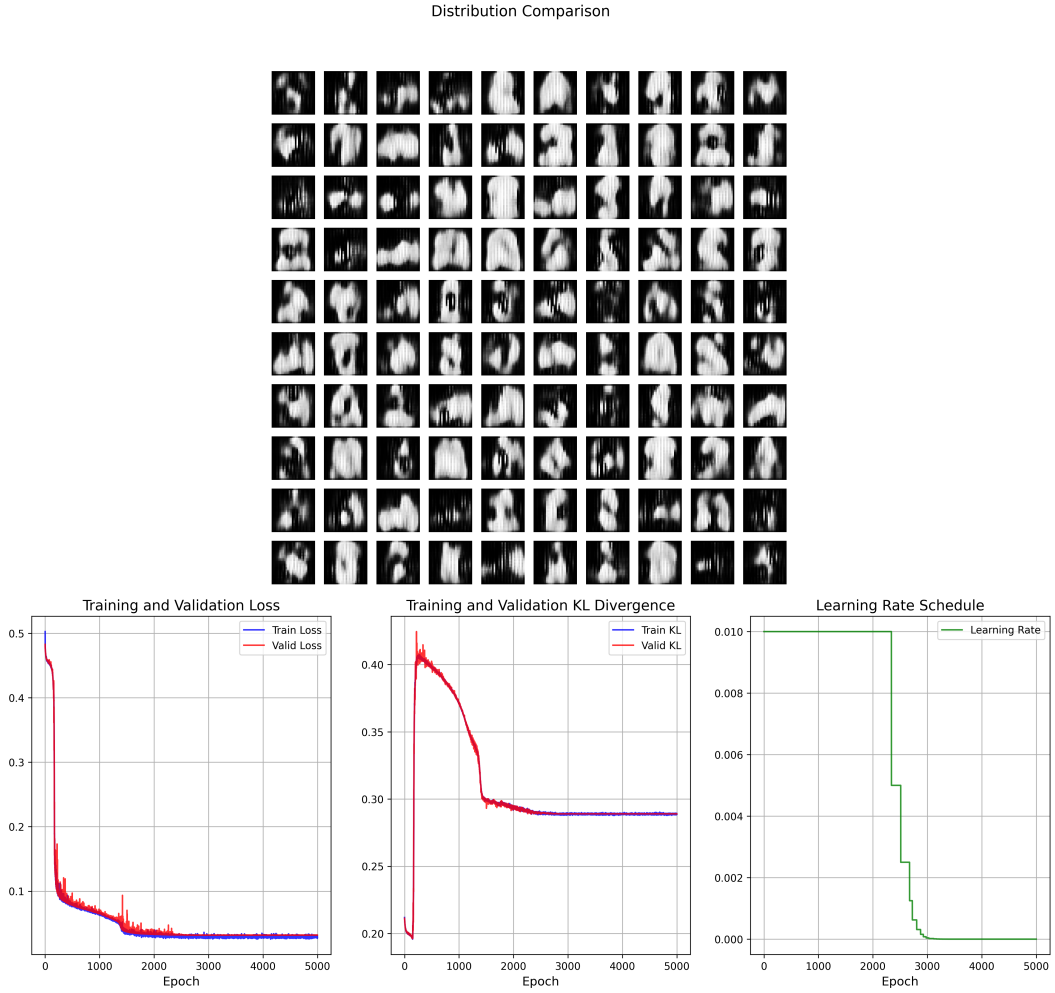


Figure 7: Image generated by the network with residual connections (top) and its training loss and metric evolution (bottom).