Learning Gradients of Convex Functions with Monotone Gradient Networks

MVA Master, ENS-Paris-Saclay

Thomas GRAVIER, Emilio PICARD
thomas.gravier@gadz.fr
emilio.picard@free.fr

Abstract

Gradients of convex functions play a fundamental role in optimization and optimal transport. Recently, deep learning techniques, particularly monotone gradient networks (MGNs), have been proposed to estimate these gradients (Chaudhari et al. [2025], Chaudhari et al. [2023]). In this work, we review the original C-MGN and M-MGN architectures, providing theoretical insights and validating them through experiments on both 2D toy datasets and large-scale benchmarks.

To further extend these methods, we investigate their application in generative modeling by approximating transport maps between probability distributions (Chaudhari et al. [2023]). Our large-scale experiments compare different architectures based on their Wasserstein scores and effectiveness in image generation.

Keywords: Convex functions, Monotone gradient, Neural networks, Optimal transport, Generative modeling.

1 Introduction

While most neural networks are unconstrained, certain applications require functions with specific properties, necessitating the design of specialized architectures. In particular, neural networks representing gradients of functions are valuable in physics, generative modeling, and optimization. Convex functions and their gradients are also fundamental in those fields. Their well-defined properties ensure efficient optimization, theoretical guarantees, and interpretability. However, designing explicit convex objectives is often laborious, requiring prior domain knowledge. Deep learning offers an alternative by providing flexible models that sidestep explicit convex formulations. Yet, neural networks often compromise interpretability, computational efficiency, and theoretical stability.

Recent research bridges convex optimization and deep learning by learning *gradients of convex functions* rather than the functions themselves (Korotin et al. [2021]). This is particularly useful when gradient fields contain richer information, as seen in optimization, inverse problems, and optimal transport. *Monotone Gradient Networks (MGN)* approximate these gradients in a data-driven manner while enforcing monotonicity, offering an efficient alternative to traditional convex optimization.

Beyond optimization, structured gradient fields are crucial in *generative modeling*. Optimal transport principles improve generative model training, notably in Wasserstein GANs (WGANs), which replace adversarial losses with transport-based objectives for stability and reduced mode collapse. Monotone gradient networks extend this idea by directly learning transport maps between distributions. Leveraging convexity and monotonicity, we introduce structured optimal transport constraints into generative models, proposing a more stable and theoretically grounded approach.

While existing MGN architectures focus on convex gradients, we extend them using *optimal transport mapping*, naturally linking them to *distribution transport*. These models, widely studied in generative modeling, learn probability distributions via distribution transformation. By integrating MGN principles, we propose a *hybrid framework* that combines convex optimization with deep generative modeling, aligning with topics explored in the MVA generative modeling course.

Our study revisits the mathematical foundations of convex gradients, evaluates MGN architectures on 2D toy examples, and explores their applications in generative modeling. We integrate MGN into domain adaptation and generative modelling for structured probability transformations and validate our approach on large-scale datasets like MNIST. Comparing classical architectures with our extended models, we assess the impact of structured convex gradients on training stability, sample quality, and efficiency.

This work contributes to the field of *structured generative models*, where optimization and deep learning combine to build robust frameworks. By extending MGN beyond convex optimization and applying them to generative modeling, we provide a new perspective on learning probability distributions, highlighting how structured gradient constraints improve deep generative models.

2 TEST

$$\sigma : \text{differentiable activation} \tag{1}$$
$$J : \text{Jacobian} \tag{2}$$
$$s_k : \text{scalar function} \tag{3}$$
$$V : \text{weighted matrix} \tag{4}$$

$f(x) = x_1^4 + \frac{x_2}{2} + \frac{x_1 x_2}{2} + \frac{3x_2^2}{2} - \frac{x_2^3}{3}$

$\nabla f(x) = [4x_1^3 + \frac{x_2}{2}, \frac{1}{2} + \frac{x_1}{2} + 3x_2 - x_2^2]$

$h(x) = \log(\exp(5x_1) + \exp(2x_2))$

$J_h(x) = [\frac{5\exp(5x_1)}{\exp(5x_1)+\exp(2x_2)}, \frac{2\exp(2x_2)}{\exp(5x_1)+\exp(2x_2)}]$

Models	Wasserstein Metric
Cascade-MGN Gaussian	0.12
Modular-MGN Gaussian	0.09
Cascade-MGN Banana	0.19
Modular-MGN Banana	0.17

3 Problem Statement and Optimal Transport

Optimal transport, introduced by Monge, seeks to map a probability distribution P_X onto another P_Y while minimizing transport cost:

$$\inf_{T:T_{\#}P_X=P_Y} \mathbb{E}_{x\sim P_X}\left[c(x,T(x))\right], \tag{5}$$

where $T : \mathbb{R}^d \to \mathbb{R}^d$ transports P_X onto P_Y (denoted $T_{\#}P_X = P_Y$), and $c(x, T(x))$ represents transport cost, typically:

$$c(x,y) = \|x-y\|^2. \tag{6}$$

Monge's problem is restrictive as it requires T to be a deterministic mapping. Kantorovich later relaxed this by considering couplings. However, under suitable conditions, ***Brenier's theorem*** (Brenier [1991]) states that if P_X and P_Y are absolutely continuous and the cost is quadratic, the optimal transport map T^* is the gradient of a convex function ϕ:

$$T^*(x) = \nabla\phi(x). \tag{7}$$

This result motivates learning *convex potential functions* to approximate transport maps, which led to the development of Input Convex Neural Networks (ICNNs) and Input Convex Gradient Networks (ICGNs).

ICNNs approximate $\phi(x)$ directly using a neural network constrained to be convex. The transport map is obtained via differentiation, but this approach suffers from expressivity limitations, slower convergence due to constrained weights, and increased computational cost. To address this, ICGNs bypass explicit convexity by learning the gradient $\nabla\phi(x)$ directly, ensuring monotonicity through architectural constraints. However, enforcing these constraints efficiently remains challenging, particularly in high-dimensional settings.

Both approaches stem from Brenier's theorem but present trade-offs in efficiency, stability, and scalability. Finding architectures that balance these constraints while maintaining theoretical guarantees is crucial for advancing optimal transport-based learning methods.

4 Monotone Gradient Network Approaches

To learn a monotone gradient function $g(x)$, we enforce:

$$g(x) = \nabla\phi(x), \tag{8}$$

for some convex, twice-differentiable function $\phi(x)$. A function $f(x)$ is convex if and only if its gradient $g(x)$ is monotone, meaning that:

$$\langle g(x) - g(y), x - y\rangle \geq 0, \quad \forall x, y \in \mathbb{R}^n. \tag{9}$$

Since directly enforcing this condition is impractical, we use an equivalent formulation: a twice-differentiable function $f(x)$ is convex if and only if its Hessian is positive semidefinite (PSD):

$$H_f(x) = J_g(x) \succeq 0, \quad \forall x \in \mathbb{R}^n. \tag{10}$$

This implies that the Jacobian $J_g(x)$ must be symmetric and PSD everywhere. Designing a neural network that satisfies this constraint is non-trivial. Our two proposed architectures, Cascaded Monotone Gradient Network (C-MGN) and Modular Monotone Gradient Network (M-MGN), are constructed to guarantee this property.

4.1 Cascaded Monotone Gradient Network (C-MGN)

The **C-MGN** is designed to ensure the PSD property of its Jacobian through a structured, layered architecture. It is defined as follows:

$$z_0 = Wx + b_0, \tag{11}$$
$$z_\ell = Wx + \sigma_\ell(z_{\ell-1}) + b_\ell, \tag{12}$$
$$\text{C-MGN}(x) = W^\top \sigma_L(z_{L-1}) + V^\top V x + b_L. \tag{13}$$

To derive its **Jacobian**, we differentiate each term step by step.

1. First layer transformation:
$$\frac{\partial z_0}{\partial x} = W.$$
2. Recursive layer update:
$$\frac{\partial z_\ell}{\partial x} = W + J_{\sigma_\ell}(z_{\ell-1})\frac{\partial z_{\ell-1}}{\partial x}.$$
Expanding this recursively for L layers, we obtain:
$$\frac{\partial z_L}{\partial x} = \sum_{i=1}^{L}\left(\prod_{\ell=i}^{L} J_{\sigma_\ell}(z_{\ell-1})\right) W.$$
3. Final output transformation:
$$J_{\text{C-MGN}}(x) = W^\top J_{\sigma_L}(z_{L-1})\frac{\partial z_{L-1}}{\partial x} + V^\top V.$$
By substituting the recursive form of $\frac{\partial z_{L-1}}{\partial x}$, we get:
$$J_{\text{C-MGN}}(x) = W^\top \left(\sum_{i=1}^{L}\left(\prod_{\ell=i}^{L} J_{\sigma_\ell}(z_{\ell-1})\right)\right) W + V^\top V. \tag{14}$$

Now, one needs to ensure the Positive Semidefiniteness (PSD). Each term in this expression is PSD:

- The activation functions σ_ℓ are element-wise increasing, making their Jacobians J_{σ_ℓ} diagonal with non-negative entries.
- The term $\prod_{\ell=i}^{L} J_{\sigma_\ell}(z_{\ell-1})$ is a product of diagonal, non-negative matrices, which remains PSD.
- The sum of these products $\sum_{i=1}^{L}\left(\prod_{\ell=i}^{L} J_{\sigma_\ell}(z_{\ell-1})\right)$ is a sum of PSD matrices, which remains PSD.
- The term $W^\top\left(\sum_{i=1}^{L}\left(\prod_{\ell=i}^{L} J_{\sigma_\ell}(z_{\ell-1})\right)\right)W$ is a quadratic form and is PSD, as it is a similarity transformation of a PSD matrix.
- The term $V^\top V$ is always PSD, as it is a Gram matrix.

Since the sum of PSD matrices remains PSD, we conclude that C-MGN ensures a positive semidefinite Jacobian, guaranteeing monotonicity.

4.2 Modular Monotone Gradient Network (M-MGN)

The **M-MGN** adopts a modular structure where each component contributes to the overall gradient function while maintaining the PSD property. It is defined as:

$$\text{M-MGN}(x) = a + V^\top V x + \sum_{k=1}^{K} s_k(z_k) W_k^\top \sigma_k(z_k), \tag{15}$$

where $s_k(z_k)$ are nonnegative convex scalar functions applied to activation outputs. To compute its Jacobian, we differentiate each term:

1. Linear term:
$$\frac{\partial}{\partial x}(V^\top V x) = V^\top V.$$
2. Nonlinear sum term: The gradient of the sum is:
$$\sum_{k=1}^{K}\left[s_k(z_k)W_k^\top J_{\sigma_k}(z_k)W_k + \left(W_k^\top \sigma_k(z_k)\right)\left(W_k^\top \sigma_k(z_k)\right)^\top\right].$$

Thus, the final Jacobian is:

$$J_{\text{M-MGN}}(x) = V^\top V + \sum_{k=1}^{K}\left[s_k(z_k)W_k^\top J_{\sigma_k}(z_k)W_k + \left(W_k^\top \sigma_k(z_k)\right)\left(W_k^\top \sigma_k(z_k)\right)^\top\right]. \tag{16}$$

Now we need to ensure Positive Semidefiniteness (PSD) We verify that each term is PSD:

- $V^\top V$ is always PSD (Gram matrix property).
- Since $s_k(z_k)$ is convex and nonnegative, it ensures that $s_k(z_k)J_{\sigma_k}(z_k)$ remains non-negative.
- The quadratic form $W_k^\top J_{\sigma_k}(z_k)W_k$ is PSD whenever $J_{\sigma_k}(z_k)$ is diagonal non-negative.
- The outer product term $\left(W_k^\top \sigma_k(z_k)\right)\left(W_k^\top \sigma_k(z_k)\right)^\top$ is always PSD.

Since the sum of PSD matrices remains PSD, we conclude that M-MGN guarantees a positive semidefinite Jacobian, ensuring monotonicity.

5 Experiments

We decided to try different applications of these models. Code for these experiments is available at https://github.com/emilio-pcrd/generative_modelling.
One first experiment has been on predicting the gradient of a 2D differentiable functions.

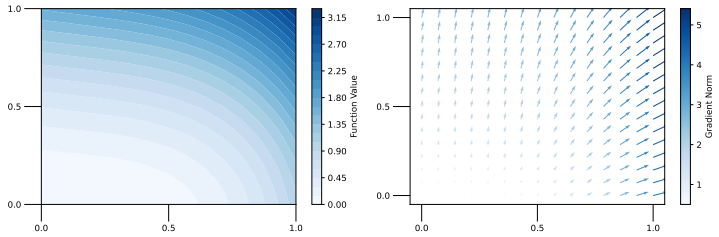
5.1 Gradient Field

The purpose here is to approximate the gradient of a convex function f. Let f be $f(x) = x_1^4 + \frac{x_2}{2} + \frac{x_1 x_2}{2} + \frac{3x_2^2}{2} - \frac{x_2^3}{3}$, where $x = [x_1, x_2]^T$. We want to approximate $\nabla f(x) = [4x_1^3 + \frac{x_2}{2}, \frac{1}{2} + \frac{x_1}{2} + 3x_2 - x_2^2]$ with both methods of the paper. We trained both networks of 90 learnable parameters during 100 epoch, with 1 milion training datapoints from the unit square, and use a learning rate of $3e^{-4}$ with an Adam optimizer. Figure 1a show the expected function f. Figure **??** show the ℓ_2 error map between true and predicted gradient for C-MGN, and the predicted Gradient norm. These trainings lead to a MAE loss of order $1e^{-5}$.

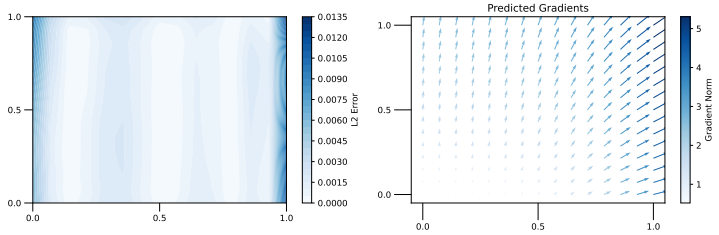
Another toy example has been with another convex function: $h(x) = \log(\exp(5x_1) + \exp(2x_2))$. With the exactly same previous settings, we also had really good results. Figures of this experiment can be found in Appendix A.

5.2 Optimal Coupling

In this experiment, we consider the Monge problem, with two different dispositives: a first experiment has been done with two Gaussian distributions P_X and P_Y, with the Euclidian cost (Peyré and Cuturi [2020], Santambrogio [2015]). As said in the paper, the Wasserstein metric gives a closed form for the optimal cost. Thus, we used this metric in order to compare the transport quality of both models. Our data is in 2D, sampled from a multivariate Gaussian distribution. We train during 500 epochs, with $lr = 3e^{-4}$ with Adam. The loss used here is the Sinkhorn distance (Cuturi [2013]). We also tried to transport a Gaussian to a banana shape distribution. Figure 2 show the transport of both Gaussian distribution to the target one (Figure 2a with Gaussian, 2b with banana shape). C-MGN achieved a 0.12 Wesserstein score. We can see that the model is well working.

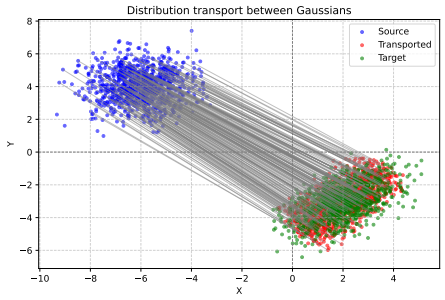


(a) True function f value and gradient norm.

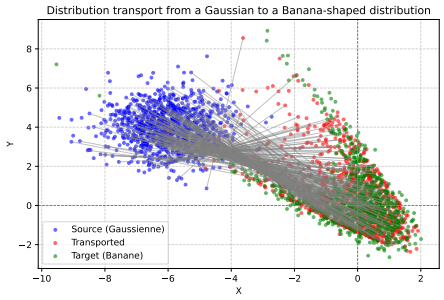


(b) Predicted values. On the left, the error ℓ_2 norm of the model prediction, and on the right, the gradient norm predictions, which are very close to the true ones.

Figure 1: Comparison between true and predicted function values and gradient norms.



(a) Distribution transport between two Gaussians using the C-MGN model. This achieves a 0.12 Wasserstein score.



(b) Distribution transport from Gaussian to Banana distribution using the M-MGN model. This achieves a 0.17 Wasserstein score (0.19 for C-MGN).

Figure 2: Comparison of distribution transport using C-MGN and M-MGN models.

5.3 Optimal Coupling with Large-Scale Datasets

For these last experiments, we used the MNIST dataset and CIFAR-10.
Our goal for this first experiment is to transport numbers to another numbers; for instance, transport digit 1 to digit 2, digit 2 to 3, 3 to 4, etc. Both models has difficulty to learn this transportation. However, this lead to good results for 7 to 8 and 8 to 9. We thus use data of dim 784, where the target digit is the original data plus one. Figure 7 of Appendix B show results of this experiment.

Our goal for this second experiment is to generate MNIST numbers from a Standard Gaussian distribution in high dimension. To do that, we generate random normal points of dimension 784 (28×28), and flatten the MNIST images as target points.
For the C-MGN model, the embedded dimension is fixed to 256, with 10 layers. We train it during 200 epochs, with a StepLR scheduler on the learning rate, decreasing it by two each 50 epochs. Results can be found in Figure 3. Models learn to generate digits from the initial Gaussian distribution. This leaded to similar results than little Variatonal Auto-Encoders.

In this third experiment, we investigated the ability of our models to colorize grayscale images from the CIFAR-10 dataset. To construct the dataset, we converted the original colored images to

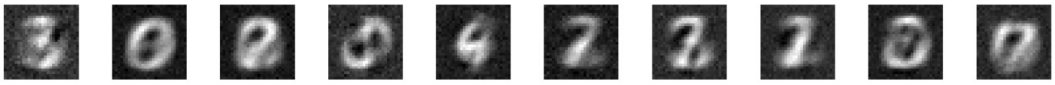


Figure 3: Digit generation using optimal coupling techniques. This experiment was conducted with C-MGN. M-MGN did not achieved better results.

grayscale, using them as input data, while the original colored versions served as the target outputs. The models were trained for 300 epochs using a Sinkhorn loss, which aligns with the principles of Optimal Transport to learn meaningful color mappings. The results, showcasing the effectiveness of our method, are presented in Figure 4.

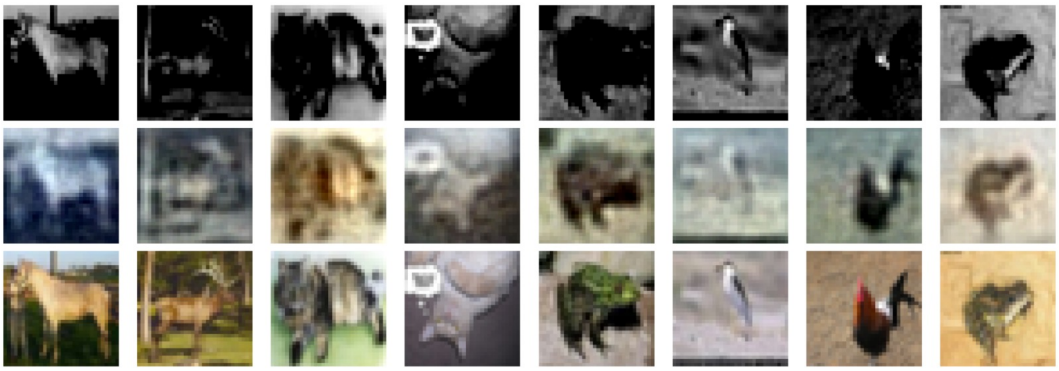


Figure 4: Examples of Colorization generation from gray-scale CIFAR-10 images (on test set). Top row is our input images, middle row show our colored generation, and bottom row show target images.

In this experiment, we aimed to modify the coloration of an image, following the approach outlined in the paper. Specifically, we modeled the pixel colors of the target image using a multivariate Gaussian distribution fitted to its pixel values. The target image's coloration is influenced by the desired transformation of the original image. For example, converting a daytime scene to nighttime requires a cooler, darker color palette in the target image. During training, we learn the model to map the original images to the target multivariate Gaussian distribution from a unique image. Figure 5 show the result of color domain adaptation on an image of Toulouse city in France.

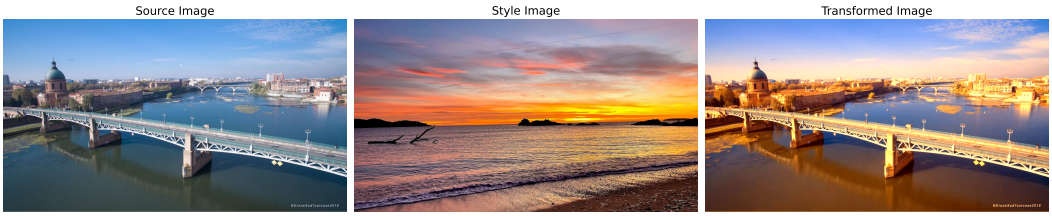


Figure 5: City of Toulouse, France. Color generation using C-MGN.

The rest of the experiments can be found Figure 8 in Appendix C.

It outline example results of this method on two cities of France, Marseille and Toulouse, using a sunset image, in order to have warm colors.

6 Conclusion

In this project, we investigated model architectures originally designed to estimate gradients of convex functions, and explored how this perspective could be applied to generative modeling. Rather than directly learning functions, we focused on learning gradient fields, which provides a theoretical link between convex optimization and deep learning.

We began by rigorously re-deriving the theoretical foundations presented in the original paper we reviewed, ensuring a solid understanding of the underlying mathematical principles. This theoretical framework was then consolidated through several toy examples, including gradient estimation on synthetic functions and optimal transport between simple distributions.

Building on these foundations, we implemented a generative modeling framework that maps Gaussian noise to structured data, with experiments on the MNIST dataset. We also applied the method to image colorization on CIFAR-10, translating grayscale images into color using learned transport maps. Finally, we reproduced a domain adaptation task involving image translation from daytime to sunset, further validating the versatility of the approach.

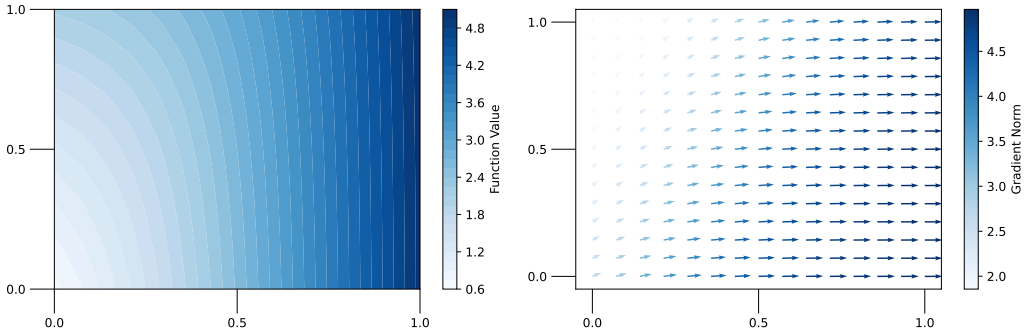
As a perspective for future work, we note that since the operator W used in our framework is linear, it could be replaced or extended with a convolutional operator. This would allow the model to better capture local pixel correlations, potentially improving transport quality and leading to more realistic image generation, while still preserving the gradient-of-convex-function structure.

This project contributes to bridging the gap between convex optimization, optimal transport, and deep learning, by combining rigorous theoretical analysis with practical implementations in generative modeling.

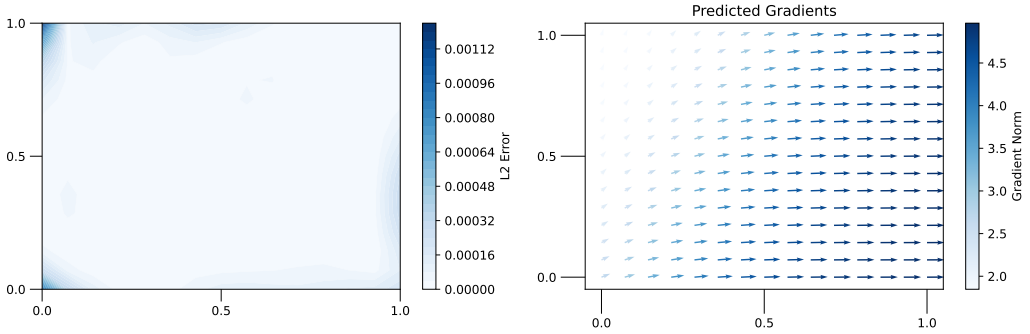
Appendix

A Gradient Field results

Parameters used: 100 epochs with a learning rate settle to $3e^{-4}$, Adam optimizer, number of modules $= 4$. We want to estimate $J_h(x) = [\frac{5\exp(5x_1)}{\exp(5x_1)+\exp(2x_2)}, \frac{2\exp(2x_2)}{\exp(5x_1)+\exp(2x_2)}]$. Figure 6 are the results of gradient fields estimation with our tested function.



(a) This is the function value and gradient norm of h.



(b) Caption for second subfigure

Figure 6: Comparison between true and predicted function values and gradient norms.

B Optimal Coupling - MNIST

This figure show the transport from one digit to another.

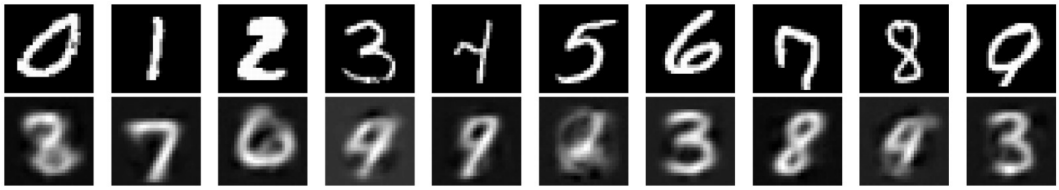
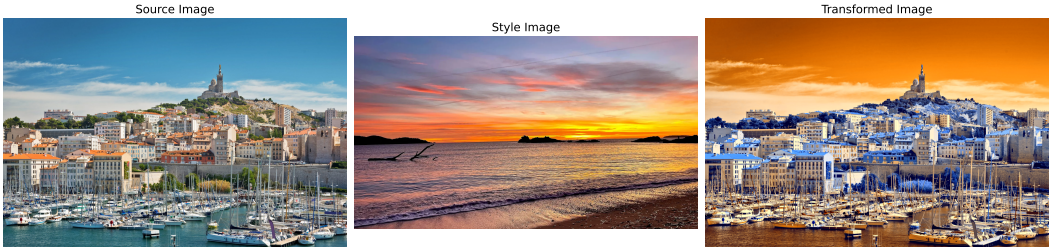
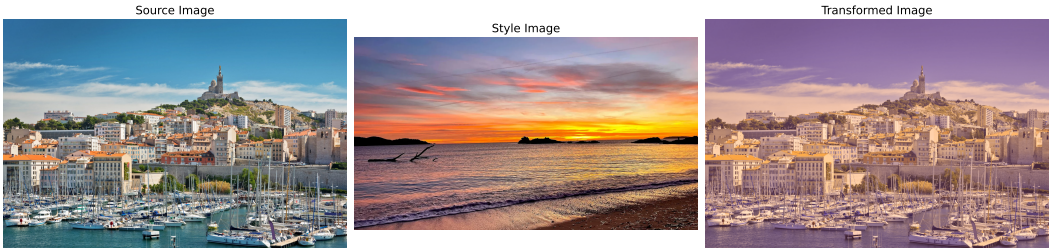


Figure 7: transport from 1 to 2, 2 to 3, etc. We can observe than the transport of 7 and 8 is working well, whereas the other ones have difficulty. The model used here is the Cascade version. The loss used is Sinkhorn.

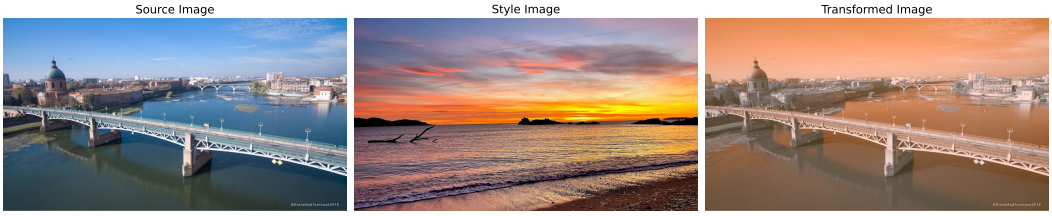
C Recolorization Experiment



(a) City of Marseille, France. Color generation using C-MGN.



(b) City of Marseille, France. Color generation using M-MGN.



(c) City of Toulouse, France. Color generation using M-MGN.

Figure 8: Comparison of colorization transport using C-MGN and M-MGN models. Figures (a) and (b) show the colorization results for the city of Marseille using C-MGN and M-MGN, respectively. Similarly, Figure (d) depict the result for the city of Toulouse for Modular-MGN. Our observations indicate that the Cascade model outperforms the Modular model in terms of colorization quality.

References

Y. Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44:375–417, 1991. URL https://api.semanticscholar.org/CorpusID:123428953.

S. Chaudhari, S. Pranav, and J. M. F. Moura. Learning gradients of convex functions with monotone gradient networks, 2023. URL https://arxiv.org/abs/2301.10862.

S. Chaudhari, S. Pranav, and J. M. F. Moura. Gradient networks, 2025. URL https://arxiv.org/abs/2404.07361.

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances, 2013. URL https://arxiv.org/abs/1306.0895.

A. Korotin, L. Li, A. Genevay, J. Solomon, A. Filippov, and E. Burnaev. Do neural optimal transport solvers work? a continuous wasserstein-2 benchmark, 2021. URL https://arxiv.org/abs/2106.01954.

G. Peyré and M. Cuturi. Computational optimal transport, 2020. URL https://arxiv.org/abs/1803.00567.

F. Santambrogio. Optimal transport for applied mathematicians. 2015. URL http://www.math.toronto.edu/mccann/assignments/477/Santambrogio15.pdf.