

LEARNING GRADIENTS OF CONVEX FUNCTIONS WITH MONOTONE GRADIENT NETWORKS

*Shreyas Chaudhari**

*Srinivasa Pranav**

José M.F. Moura

Electrical and Computer Engineering, Carnegie Mellon University

ABSTRACT

While much effort has been devoted to deriving and analyzing effective convex formulations of signal processing problems, the gradients of convex functions also have critical applications ranging from gradient-based optimization to optimal transport. Recent works have explored data-driven methods for learning convex objective functions, but learning their monotone gradients is seldom studied. In this work, we propose C-MGN and M-MGN, two monotone gradient neural network architectures for directly learning the gradients of convex functions. We show that, compared to state of the art methods, our networks are easier to train, learn monotone gradient fields more accurately, and use significantly fewer parameters. We further demonstrate their ability to learn optimal transport mappings to augment driving image data.

Index Terms— Convex Functions, Monotone Gradient, Neural Network, Optimal Transport

1. INTRODUCTION

Convex functions have been studied and celebrated for their amenable analytic properties, relative ease of optimization, and plethora of applications. When finding solutions to signal processing problems, convex formulations enable us to easily augment objective functions and incorporate prior domain knowledge regarding the structure of the solution. However, for complex problems where prior domain knowledge is either lacking or insufficient, deep learning approaches are attractive alternatives that rely on purely data-driven, nonconvex, and overparameterized problem formulations. Deep neural networks have achieved state of the art performance on a variety of image and speech processing tasks at the cost of sacrificing many benefits of convex optimization: computational efficiency, interpretability, and theoretical guarantees. Thus, even in the age of deep learning, convex optimization methods offer significant value.

Formulating convex optimization problems is an active area of research that permeates nearly all of signal process-

ing, from source localization in communications to image deblurring [3]. However, it is often a laborious process that involves manually designing suitable convex objectives and associated convex constraints. Perhaps more important than the objective function itself is the *gradient* of the function, since most convex problems are solved using computationally frugal gradient-based methods. Monotone gradient maps of convex functions also have critical applications in domains including gradient-based optimization, generalized linear models, linear inverse problems, and optimal transport. Therefore, in this work, we propose to *learn* the gradient of convex functions in a data-driven manner using deep learning. Our approach is a fundamental step toward blending strengths of both deep learning and convex optimization and offers a wide array of applications in data science and signal processing.

Contributions: We propose two neural network architectures for learning gradients of convex functions, i.e., monotone gradient functions [4]. To the best of our knowledge, we are the first to propose a method for *directly* parameterizing and learning monotone gradients of convex functions, without first learning the underlying convex function or its Hessian. In contrast to current methods, our networks are considerably easier to train and generalize to high-dimensional problem settings. In this work, we show empirically the efficacy of our approach on a set of standard problems and an image color domain adaptation task.

2. RELATED WORK

2.1. Learning Loss Functions and Regularizers

Parameterized, monotone gradient functions are useful when we want to optimize input data to minimize a desired loss function that is difficult to express analytically. Recent works explored *learning* parameters of an objective function during training and then using it to optimize an input at inference time. The adversarial method for training a regularizer in [5] entails a nonconvex optimization problem to generate predictions at inference time. In contrast, an Input Convex Neural Network (ICNN) [6] constrains the learned objective function to be convex with respect to its input. To optimize a proposed input using gradient descent updates, the ICNN must be differentiated at inference time. In this work, we avoid the ex-

* Authors contributed equally. Authors partially supported by NSF Graduate Research Fellowships (GRFP; Grants DGE1745016, DGE2140739) and XSEDE [1] Allocation ELE220003 on PSC Bridges-2 system [2]. S. Pranav partially supported by an ARCS Fellowship. Authors thank Tyler Vuong for insightful discussions.

tra differentiation step by directly learning the gradient of the convex objective. Furthermore, ICNNs are infamously hard to train [7] as most weights must be positive and nonlinearities must be convex, monotonically increasing functions.

2.2. Monotone Gradient Solutions to Optimal Transport Problems

Directly learning the monotone gradient of a convex function is fruitful even when the convex function itself is unknown. Monotone gradients are significant in optimal transport (Monge) problems that arise in numerous scenarios, including domain adaptation [8, 9]. For example, computer vision algorithms used in autonomous vehicle applications require large amounts of human-annotated training images that may be too expensive or infeasible to obtain for a wide variety of factors like location, lighting, and weather conditions. One promising generative approach to augmenting training data involves solving an optimal transport problem to transform existing annotated road images into similar images under different visual settings. For example, daytime images can be mapped to realistic twilight images of the same scene by solving an optimal transport problem between estimated daytime and nighttime image distributions. Formally, one can solve

$$\inf_{g: g(x) \sim p_Y} \mathbb{E}_{x \sim p_X} [c(x, g(x))] \quad (1)$$

for a given cost function c in order to determine an invertible mapping function g that transforms a random variable $x \sim p_X$ to $y \sim p_Y$. Choosing the cost function to be squared Euclidean distance, Brenier’s theorem [10, 11] states that the unique optimal mapping function g that solves Eqn. 1 is a monotone gradient of a convex function. Brenier’s theorem inspired several recent normalizing flow works [12, 13] which attempt to optimally transport one probability distribution to another using the gradient of a learnable convex function. These methods train an ICNN [6] for the sole purpose of extracting its gradient map using automatic differentiation. These approaches avoid directly parameterizing the gradient map, are computationally inefficient, and are subject to the difficulties of training ICNNs.

Input Convex Gradient Networks (ICGNs) [7] take an indirect and more complex approach to alleviate the issues associated with ICNNs: they use a standard neural network to learn factors of a positive semidefinite Hessian matrix and numerically approximate its integral. This computationally expensive approach is not feasible for high-dimensional problems and requires that the neural network always satisfies a partial differential equation. Currently, only zero hidden layer perceptrons are known to always satisfy this condition. While related theoretical analysis is sparse, the symmetry of a scalar-valued function’s second derivatives reveals that approximating the gradient of any function using a standard multilayer perceptron with more than one hidden layer requires only representing one feature in the first hidden layer [14]. In this

work, we propose two specific neural network architectures designed with explicit weight-tying that avoid this limitation.

3. PROBLEM STATEMENT

We learn a monotone gradient function $g(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that $g(\mathbf{x}) = \nabla f(\mathbf{x})$, for some convex, twice differentiable function $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$. A differentiable function $f(\mathbf{x})$ is convex if and only if its gradient $g(\mathbf{x})$ is monotone [15]:

$$\langle g(\mathbf{x}) - g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq 0 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \quad (2)$$

Since this condition is difficult to enforce in practice, we rely on a twice differentiable function $f(\mathbf{x})$ being convex if and only if its Hessian is positive semidefinite (PSD) [15]:

$$\mathbf{H}_f(\mathbf{x}) = \mathbf{J}_g(\mathbf{x}) \succeq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \quad (3)$$

This condition subsumes the requirement of a symmetric Jacobian \mathbf{J}_g . Thus, parameterizing $g(\mathbf{x})$ with a neural network requires the Jacobian of the neural network to be PSD everywhere. **Note:** in this work, Jacobians of neural networks are always computed with respect to their *inputs*.

4. PROPOSED APPROACHES

We propose two monotone gradient neural network (MGN) architectures to learn gradients of convex functions: C-MGN and M-MGN. They are motivated by two different approaches that ensure a neural network’s Jacobian (with respect to its input) is PSD everywhere.

4.1. Cascaded Network: C-MGN

We propose a *cascaded* monotone gradient network (C-MGN):

$$\mathbf{z}_0 = \mathbf{W}\mathbf{x} + \mathbf{b}_0 \quad (4)$$

$$\mathbf{z}_\ell = \mathbf{W}\mathbf{x} + \sigma_\ell(\mathbf{z}_{\ell-1}) + \mathbf{b}_\ell \quad (5)$$

$$C\text{-MGN}(\mathbf{x}) = \mathbf{W}^\top \sigma_L(\mathbf{z}_{L-1}) + \mathbf{V}^\top \mathbf{V}\mathbf{x} + \mathbf{b}_L \quad (6)$$

Layer outputs \mathbf{z}_ℓ , biases \mathbf{b}_ℓ , and activation functions σ_ℓ can differ for each layer ℓ , but all L layers share weight matrix \mathbf{W} . Prop. 1 shows that the proposed C-MGN is monotone for popular activation functions, e.g., tanh, sigmoid, and softplus.

Proposition 1. *If $\forall \ell \sigma_\ell(\cdot)$ is a differentiable, element-wise activation function that is monotonically increasing, then $\forall \mathbf{x} \mathbf{J}_{C\text{-MGN}}(\mathbf{x}) \succeq 0$.*

Proof. The Jacobian of C-MGN(\mathbf{x}) with respect to \mathbf{x} is:

$$\mathbf{J}_{C\text{-MGN}}(\mathbf{x}) = \mathbf{W}^\top \left(\sum_{\ell=1}^L \prod_{i=\ell}^L \mathbf{J}_{\sigma_i}(\mathbf{z}_{i-1}) \right) \mathbf{W} + \mathbf{V}^\top \mathbf{V} \quad (7)$$

If each $\sigma_i(\cdot)$ is a monotonically increasing, element-wise function, then each \mathbf{J}_{σ_i} is diagonal and non-negative. Therefore, $\sum_{\ell=1}^L \prod_{i=\ell}^L \mathbf{J}_{\sigma_i}(\mathbf{z}_{i-1})$ is a diagonal, non-negative matrix and is PSD. Since $\mathbf{W}^\top \mathbf{A} \mathbf{W} \succeq 0$ for any $\mathbf{A} \succeq 0$ and the sum of PSD matrices is also PSD, $\forall \mathbf{x} \mathbf{J}_{C\text{-MGN}}(\mathbf{x}) \succeq 0$. \square

4.2. Modular Network: M-MGN

We define a modular monotone gradient network (M-MGN):

$$\mathbf{z}_k = \mathbf{W}_k \mathbf{x} + \mathbf{b}_k \quad (8)$$

$$\text{M-MGN}(\mathbf{x}) = \mathbf{a} + \mathbf{V}^\top \mathbf{V} \mathbf{x} + \sum_{k=1}^K s_k(\mathbf{z}_k) \mathbf{W}_k^\top \sigma_k(\mathbf{z}_k) \quad (9)$$

where \mathbf{a} is a bias parameter and the number of modules K can be tuned based on the application. The activation function σ_k , weight matrix \mathbf{W}_k , and bias \mathbf{b}_k can differ for each network module. Prop. 2 relates scalar-valued functions s_k to σ_k .

Proposition 2. If $\forall k s_k(\mathbf{x})$ is a convex, twice differentiable, nonnegative scalar function and $\sigma_k(\cdot) = \nabla s_k(\cdot)$, then $\forall \mathbf{x} \mathbf{J}_{\text{M-MGN}}(\mathbf{x}) \succeq 0$.

Proof. The Jacobian of M-MGN(\mathbf{x}) with respect to \mathbf{x} is:

$$\begin{aligned} \mathbf{J}_{\text{M-MGN}}(\mathbf{x}) &= \mathbf{V}^\top \mathbf{V} + \sum_{k=1}^K \left[s_k(\mathbf{z}_k) \mathbf{W}_k^\top \mathbf{J}_{\sigma_k}(\mathbf{z}_k) \mathbf{W}_k \right. \\ &\quad \left. + (\mathbf{W}_k^\top \sigma_k(\mathbf{z}_k)) (\mathbf{W}_k^\top \sigma_k(\mathbf{z}_k))^\top \right] \end{aligned} \quad (10)$$

Observe that convex $s_k(\cdot)$ implies $\mathbf{J}_{\sigma_k}(\mathbf{x}) \succeq 0$. Since $s_k(\cdot)$ is a scalar, nonnegative function and $\mathbf{A} \succeq 0$ implies $\mathbf{B}^\top \mathbf{A} \mathbf{B} \succeq 0$, the first term in the summation is PSD. The second term in the summation and $\mathbf{V}^\top \mathbf{V}$ are PSD as they are Gram matrices. Sums of PSD matrices are PSD, therefore $\mathbf{J}_{\text{M-MGN}} \succeq 0$. \square

Note that requiring $s_k(\mathbf{x}) \geq 0$ is not restrictive. For example, if x_i denotes the i^{th} component of \mathbf{x} , then $s_k(\mathbf{x}) = \log \cosh(x_1) + \dots + \log \cosh(x_n)$ yields the element-wise activation $\sigma_k(\mathbf{x}) = \tanh(\mathbf{x})$. Similar functions $s_k(\cdot)$ can be easily found when $\sigma_k(\cdot)$ is the element-wise softplus or sigmoid functions. Furthermore, the activations $\sigma_k(\cdot)$ in the M-MGN are not restricted to be element-wise.

4.3. Discussion

The main distinction between the aforementioned approaches is that C-MGN represents monotone gradients via a single *deep* network and M-MGN via several *parallel* networks. Additionally, note that both C-MGN and M-MGN can easily be generalized to incorporate convolutional layers. The \mathbf{W} in C-MGN and each \mathbf{W}_k in M-MGN can be replaced with any linear operator while maintaining a PSD Jacobian – assuming the transposed weight matrices are replaced with the adjoint operator. Furthermore, the network outputs can be easily adjusted to represent gradients of strongly convex functions. A twice differentiable function $f(\mathbf{x})$ is *strongly* convex if there exists $\gamma > 0$ such that $\mathbf{H}_f(\mathbf{x}) \succeq \gamma \mathbf{I}$. By adding a scalar multiple of the input to the output, we can ensure that the learned gradients correspond to strongly convex functions and that the

network is invertible [4, 12]. In particular, this allows us to use our network architecture as a normalizing flow.

Since non-negative diagonal matrices are closed under matrix multiplication, C-MGN’s Jacobian would remain PSD even if different learnable, non-negative diagonal matrices were composed after (to the left of) each \mathbf{W} and $\sigma_\ell(\cdot)$. The same applies to M-MGN modules where the activations $\sigma_k(\cdot)$ operate element-wise. In practice, we observe that these non-negative diagonal matrices (which scale dimensions) avoid training difficulties faced by non-negative dense matrices in ICNNs (which perform conical combinations) [6].

5. EXPERIMENTS

5.1. Gradient Field

We compare our proposed models to ICGNs [7] and ICNNs [6] using the standard problem in [7]: estimating the gradient field of $f(\mathbf{x})$ over the unit square, where $\mathbf{x} = [x_1 \ x_2]^\top$ and

$$f(\mathbf{x}) = \mathbf{x}_1^4 + \frac{\mathbf{x}_2}{2} + \frac{\mathbf{x}_1 \mathbf{x}_2}{2} + \frac{3\mathbf{x}_2^2}{2} - \frac{\mathbf{x}_2^3}{3} \quad (11)$$

We adopt the procedure used in [7] and train all models using 1 million points randomly sampled from the unit square and mean absolute error loss. All models were trained using the same batch size, learning rate, and number of epochs. Fig. 1 illustrates the gradient field of $f(\mathbf{x})$ and the ℓ_2 -norm errors between true and predicted gradients. We note that our proposed C-MGN and M-MGN more accurately learn $\nabla f(\mathbf{x})$ while requiring considerably fewer parameters.

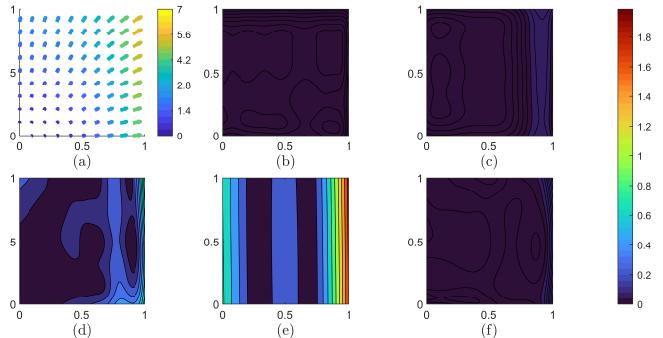


Fig. 1: (a) Gradient of $f(\mathbf{x})$ (Eqn. 11) - arrow color corresponds to ℓ_2 norm of $\nabla f(\mathbf{x})$. ℓ_2 -error maps between true and predicted gradient for: (b) C-MGN* (c) M-MGN* (d) ICGN (e) ICNN with 78 parameters (f) ICNN with 163 parameters

Table 1: Results for estimating $\nabla f(\mathbf{x})$ from Eq. 11. *Our methods.

	ICGN [7]	ICNN [6]	C-MGN*	M-MGN*
Parameters	15	78	163	14
MSE (dB)	-15.00	-4.15	-30.88	-39.10
				-32.31

As shown in Table 1, our proposed C-MGN achieves a ≈ 10 dB lower MSE than an ICNN while using less than 10% of the number of parameters.

5.2. Optimal Coupling

We consider the Monge problem in Eqn. 1 with Gaussian data distribution p_X , standard Gaussian prior p_Y , and Euclidean transport cost c . For this restricted Gaussian case, the Wasserstein distance metric gives a closed form solution for the optimal cost. We study cases where the data samples lie in $d = 2$ and 16 dimensions, as in [12], and compare our results with Convex Potential Flows (CP Flows) [12], Invertible Autoregressive Flows (IAFs) [16] and Cholesky-based whitening transform (WT). All models are trained to minimize KL-divergence with the standard normal prior distribution and rely on the change-of-variable formula for probability densities, as discussed in [12]. For results summarized in Table 2, a lower negative log-likelihood (NLL) signifies that model outputs better fit the standard normal distribution; a lower cost means that data points are moved less when transforming them to fit the standard normal prior. We observe that for $d = 2$, both C-MGN and M-MGN achieve a lower NLL than CP Flow and IAF. Furthermore, Fig. 2 shows how our methods reduce the original data points’ movement and incur the smallest transport cost. Our methods are competitive with the others when $d = 16$, as they achieve lower NLL than CP Flow and a significantly lower cost than IAF.

Table 2: Gaussian optimal coupling results (lower values are better). The optimal costs are $d = 2$ and $d = 16$ are 3.71; and 211.97 respectively. *Our methods.

	$d = 2$		$d = 16$	
	NLL	Cost	NLL	Cost
Whitening Transform	2.83	4.13	22.70	235.28
IAF [16]	3.00	4.04	22.53	234.82
CP Flow [12]	3.10	3.47	22.71	210.10
C-MGN*	2.86	3.30	22.67	212.66
M-MGN*	2.87	3.21	22.61	211.84

5.3. Color Domain Adaptation for Autonomous Driving

We trained C-MGN and M-MGN to map daytime road scenes to identical scenes at sunset time, thereby affordably augmenting existing human-annotated image datasets. First, we fit a multivariate Gaussian distribution to the pixel colors present in a target image. Then, similar to Section 5.2, we train our network to learn a mapping from source training image pixels to the target multivariate Gaussian by minimizing the NLL of the mapped image pixel colors. We train our models using a single target image of a sunset and a single training image of a road from Dark Zurich [17, 18] dataset’s validation set. Fig. 3 shows that both architectures are able to achieve training and

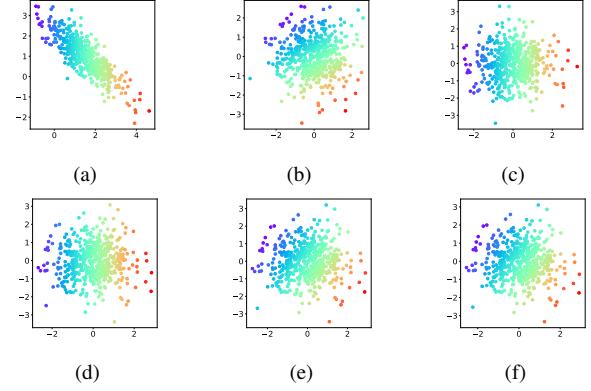


Fig. 2: (a) Original data points (colored according to x-axis), Data points after being transported by: (b) WT, (c) IAF [16], (d) CP Flow [12] (e) C-MGN*, (f) M-MGN*. *Our methods.

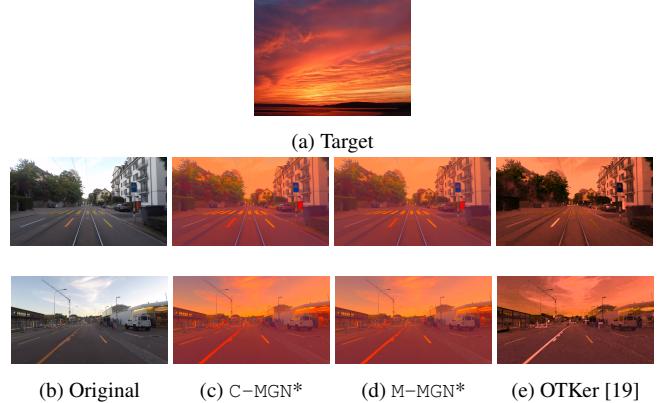


Fig. 3: Color domain adaptation results. Top image: target sunset image. First row: the single training image. Second row: an example test image. *Our methods.

test results comparable to a baseline that solves a convex relaxation of the optimal transport problem with added regularizers and Gaussian kernel functions [19].

6. CONCLUSION

We propose two monotone gradient network architectures, C-MGN and M-MGN, for learning the gradients of convex functions. To the best of our knowledge, this is the first work to directly parameterize and learn gradients of convex functions, without learning the underlying convex function or its Hessian. We show that our networks are guaranteed to represent the gradient of a convex function, suitable for high-dimensional problems, and straightforward to train. We demonstrate that the proposed architectures can learn monotone gradients more accurately and with far fewer parameters than state of the art methods.

7. REFERENCES

- [1] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. Scott, and N. Wilkins-Diehr, “Xsede: Accelerating scientific discovery,” *Computing in Science & Engineering*, vol. 16, no. 05, pp. 62–74, sep 2014.
- [2] Shawn T. Brown, Paola Buitrago, Edward Hanna, Sergiu Sanielevici, Robin Scibek, and Nicholas A. Nystrom, “Bridges-2: A platform for rapidly-evolving and data intensive research,” in *Practice and Experience in Advanced Research Computing*, New York, NY, USA, 2021, PEARC ’21, Association for Computing Machinery.
- [3] Daniel P Palomar and Yonina C Eldar, *Convex optimization in signal processing and communications*, Cambridge University Press, 2010.
- [4] R Tyrrell Rockafellar, *Convex analysis*, vol. 18, Princeton University Press, 1970.
- [5] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb, “Adversarial regularizers in inverse problems,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [6] Brandon Amos, Lei Xu, and J Zico Kolter, “Input convex neural networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 146–155.
- [7] Jack Richter-Powell, Jonathan Lorraine, and Brandon Amos, “Input convex gradient networks,” *arXiv preprint arXiv:2111.12187*, 2021.
- [8] R Flamary, N Courty, D Tuia, and A Rakotomamonjy, “Optimal transport for domain adaptation,” *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 1, 2016.
- [9] Ievgen Redko, Nicolas Courty, Rémi Flamary, and Devis Tuia, “Optimal transport for multi-source domain adaptation under target shift,” in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 849–858.
- [10] Yann Brenier, “Polar factorization and monotone rearrangement of vector-valued functions,” *Communications on Pure and Applied Mathematics*, vol. 44, no. 4, pp. 375–417, 1991.
- [11] Filippo Santambrogio, “Optimal transport for applied mathematicians,” *Birkhäuser, NY*, vol. 55, no. 58-63, pp. 94, 2015.
- [12] Chin-Wei Huang, Ricky TQ Chen, Christos Tsirigotis, and Aaron Courville, “Convex potential flows: Universal probability distributions with optimal transport and convex optimization,” 2020.
- [13] Ashok Makkluva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee, “Optimal transport mapping via input convex neural networks,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 6672–6681.
- [14] Saeed Saremi, “On approximating ∇f with neural networks,” *arXiv preprint arXiv:1910.12744*, 2019.
- [15] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.
- [16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling, “Improved variational inference with inverse autoregressive flow,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [17] Christos Sakaridis, Dengxin Dai, and Luc Van Gool, “Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [18] Christos Sakaridis, Dengxin Dai, and Luc Van Gool, “Map-guided curriculum domain adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [19] Michaël Perrot, Nicolas Courty, Rémi Flamary, and Amaury Habrard, “Mapping estimation for discrete optimal transport,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.