Appendix

A. Training of the GW GAN with Fixed Adversary

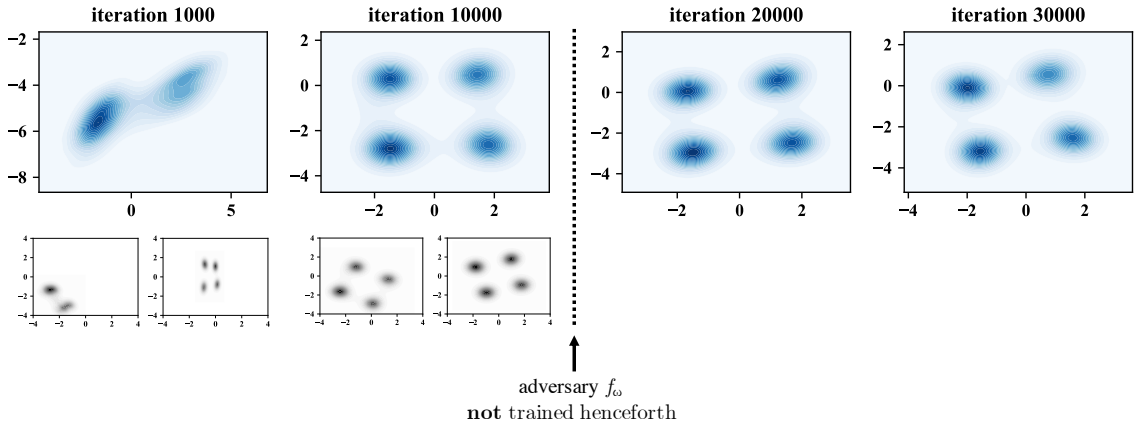

Figure 6. Results demonstrating the consistency of the GW training objective when fixing adversary f_ω. Even after holding the adversary f_ω fixed and stopping its training after 10000 iterations, the generator does not diverge and remains consistent. All plots display 1000 samples.

B. Influence of Generator Constraints on the GW GAN

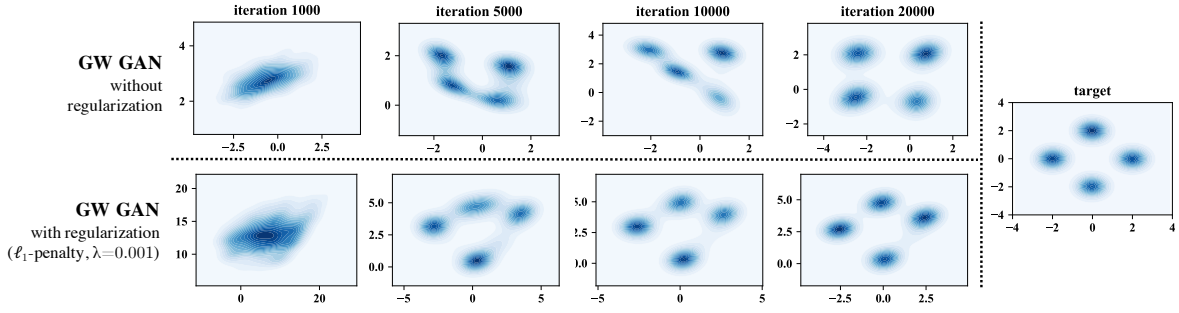

Figure 7. The GW GAN recovers relational and geometric properties of the reference distribution. Global aspects can be determined via constraints of the generator g_θ. When learning a mixture of four Gaussians we can enforce centering around the origin by penalizing the ℓ_1-norm of the generated samples. The results show training the GW GAN with and without a ℓ_1-penalty. While the GW GAN recovers all modes in both cases, learning with ℓ_1-regularization centers the resulting distribution around the origin and determines its orientation in the Euclidean plane. All plots display 1000 samples.

C. Influence of the Adversary

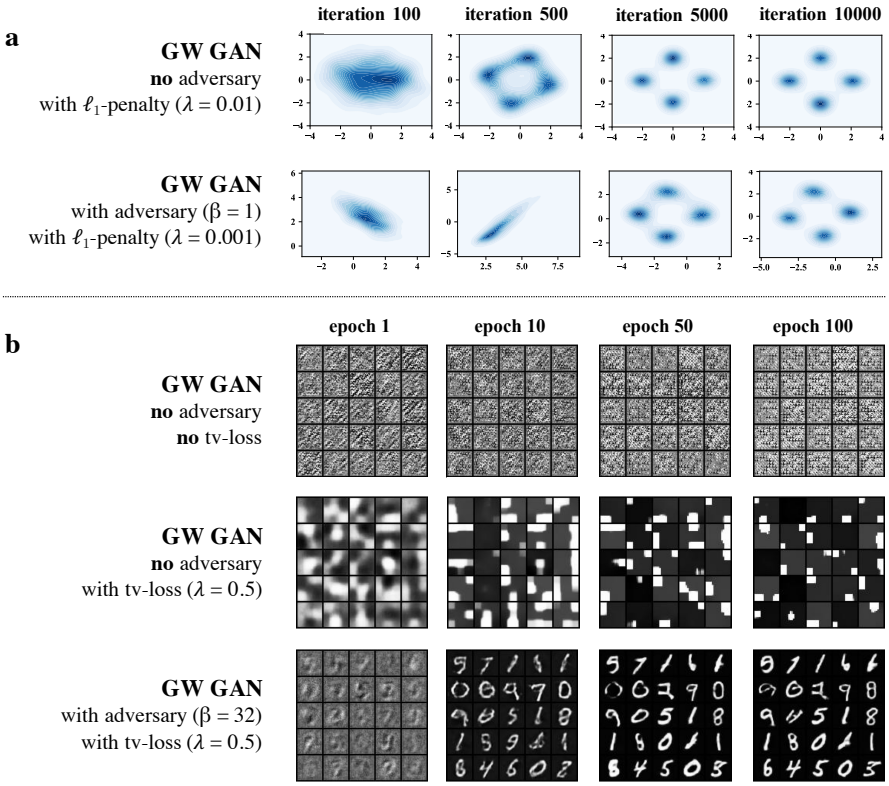

Figure 8. Learning the intra-space metrics adversarially is crucial for high dimensional applications. **a.** For simple applications such as generating 2D Gaussian distributions, the GW GAN performs well irrespective of the use of an adversary. **b.** However, for higher dimensional inputs, such as generating MNIST digits, the use of the adversary is crucial. Without the adversary, the GW GAN is not able to recover the underlying target distributions, while the total variation regularization (tv-loss) effects a clustering of local intensities.

D. Comparison of the Effectiveness of Orthogonal Regularization Approaches

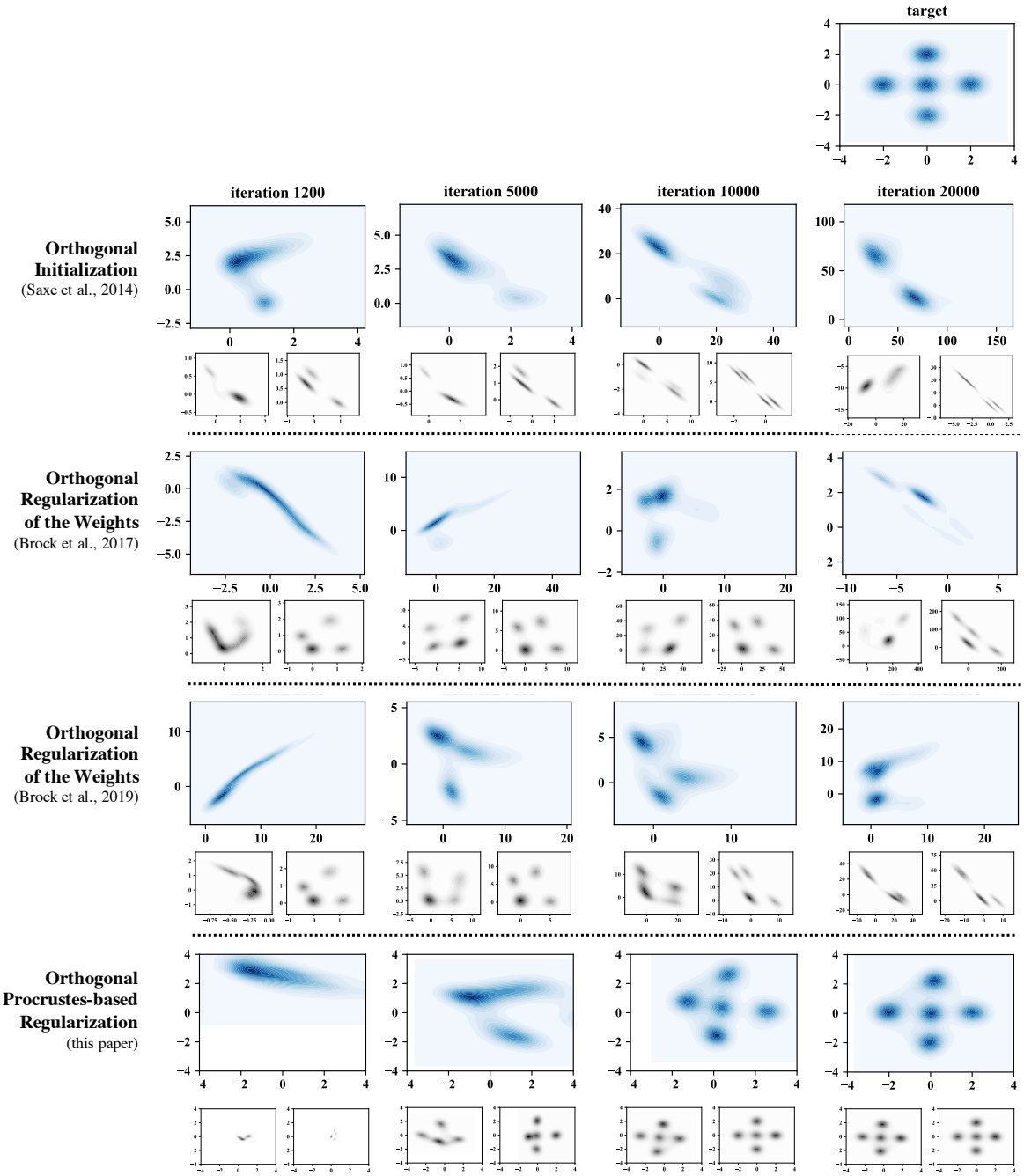

Figure 9. To avoid arbitrary distortion of the space by the adversary, we propose to regularize f_ω by (approximately) enforcing it to define a unitary transformation. We compare different methods of orthogonal regularization of neural networks; including random initialization of the network's weights at the beginning of the training (Saxe et al., 2014), and layerwise orthogonality constraints which penalize deviations of the weights from orthogonality ($R_\beta(W_k) := \beta\|W_k^\top W_k - I\|_{\mathrm{F}}^2$) (Brock et al., 2017). Brock et al. (2019) remove the diagonal terms from the regularization, which enforces the weights to be orthogonal but does not constrain their norms ($R_\beta(W_k) := \beta\|W_k^\top W_k \odot (1 - I)\|_{\mathrm{F}}^2$). The approaches of Saxe et al. (2014); Brock et al. (2017; 2019) are not able to tightly constrain f_ω. As a result, the adversary is able to stretch the space and thus maximize the Gromov-Wasserstein distance. Only the orthogonal Procrustes-based orthogonality approach introduced in this paper is able to effectively regularize adversary f_ω preventing arbitrary distortions of the intra-space distances in the feature space. All plots display 1000 samples.

E. Comparison of the GW GAN with Salimans et al. (2018)

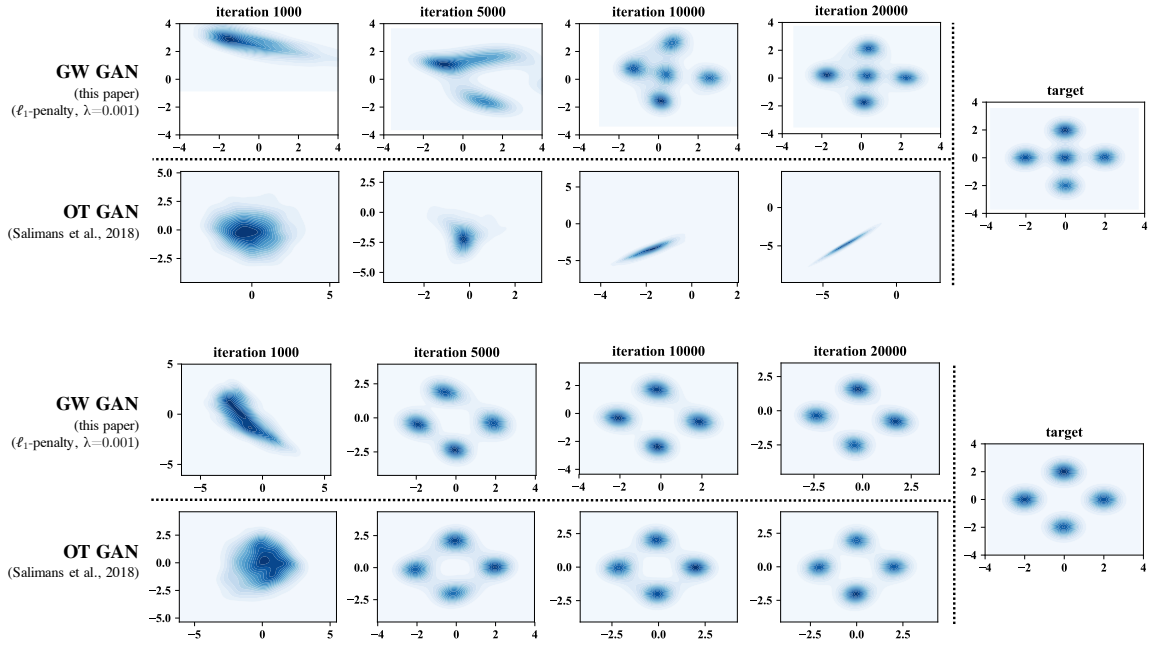

Figure 10. Comparison of the performance of the GW GAN and an OT based generative model proposed by Salimans et al. (2018) (OT GAN). The GW GAN learns mixture of Gaussians with differing number of modes and arrangements. The approach of Salimans et al. (2018) is not able to recover a mixture of five Gaussians with a centering distribution. In the case of a mixture of four Gaussian distributions, both models are able to recover the reference distribution in a similar number of iterations. All plots display 1000 samples.

F. Comparison of Training Times

Model	**Average Training Time** (Seconds per Epoch)
WASSERSTEIN GAN with Gradient Penalty (Gulrajani et al., 2017)	17.57 ± 2.07
SINKHORN GAN (Genevay et al., 2018) (default configuration, $\epsilon = 0.1$)	145.52 ± 1.90
SINKHORN GAN (Genevay et al., 2018) (default configuration, $\epsilon = 0.005$)	153.86 ± 1.64
GW GAN (this paper, $\epsilon = 0.005$)	156.62 ± 1.06

Table 1. Training time comparisons of PyTorch implementations of different GAN architectures. The generative models were trained on generating MNIST digits and their average training time per epoch was recorded. All experiments were performed on a single GPU for consistency.

G. Training Details of the Style Adversary

We introduce a novel framework which allows a modular application of style transfer tasks by integrating a *style adversary* into the architecture of the Gromov-Wasserstein GAN. In order to demonstrate the practicability of this modular framework, we learn MNIST digits and enforce their font style to be bold via additional design constraints. The style adversary is parametrized by a binary classifier trained on handwritten letters of the EMNIST dataset (Cohen et al., 2017) which were assigned bold and thin class labels based on the letterwise ℓ_1-norm of each image. As the style adversary is trained based on a different dataset, it is independent of the original learning task. The binary classifier is parametrized by a convolutional neural network and trained by computing a binary cross-entropy loss. The dataset, classification results of bold and thin letters as well as the loss curve of training the binary classifier are shown in figure 11.

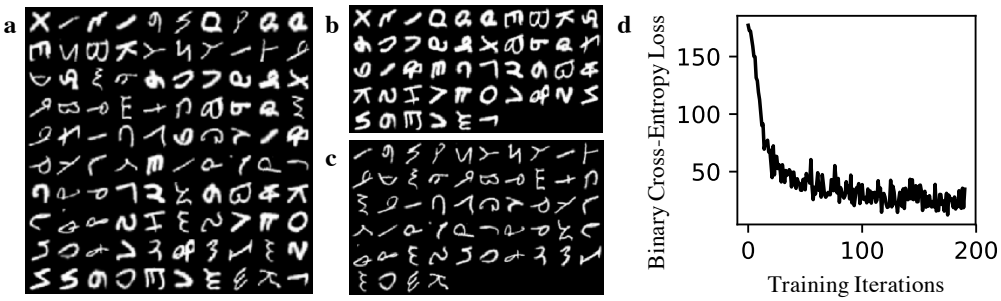

Figure 11. Training of a binary classifier to discriminate between bold and thin letters. **a.** Training set of the EMNIST dataset including bold and thin letters. Output of the trained network of letters labelled as **b.** bold and **c.** thin. **d.** Loss curve corresponding to the training of the binary classifier.