A Selecting ϵ for MNIST

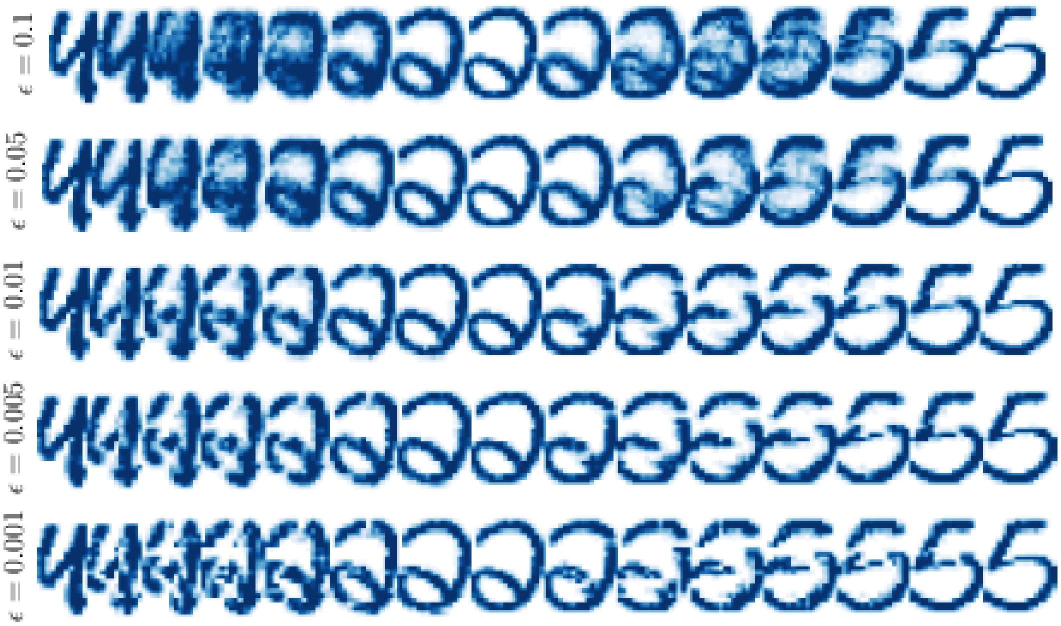

Figure 8: We selected $\epsilon = 10^{-2}$ for our MNIST coupling experiments as it results in transport maps that are not too blurry or sharp.

B Other models for continuous OT

While developing the hyper-network or Meta ICNN in sect. 3.2 for predicting couplings between continuous measures, we considered alternative modeling formulations briefly documented in this section. We finalized only the hyper-network model because it is conceptually the most similar to predicting the optimal dual variables in the continuous setting and results in rapid predictions.

B.1 Optimization-based meta-learning (MAML-inspired)

The model-agnostic meta-learning setup proposed in MAML [Finn et al., 2017] could also be applied in the Meta OT setting to learn an adaptable initial parameterization. In the continuous setting, one initial version would take a parameterized dual potential model $\psi_\varphi(x)$ and seek to learn an initial parameterization φ_0 so that optimizing a loss such as the W2GN loss $\mathcal{L}$ from eq. (12) results in a minimal $\mathcal{L}(\varphi_K)$ after adapting the model for K steps. Formally, this would optimize:

$$\arg\min_{\varphi_0} \mathcal{L}(\varphi_K) \quad \text{where} \quad \varphi_{t+1} = \varphi_t - \nabla_\varphi \mathcal{L}(\varphi_t) \tag{18}$$

Tancik et al. [2021] explores similar learned initializations for coordinate-based neural implicit representations for 2D images, CT scan reconstruction, and 3d shape and scene recovery from 2D observations.

Challenges for Meta OT. The transport maps given by $T = \nabla\psi$ can significantly vary depending on the input measures α, β. We found it difficult to learn an initialization that can be rapidly adapted, and optimizing eq. (18) is more computationally expensive than eq. (17) as it requires unrolling through many evaluations of the transport loss $\mathcal{L}$. And, we found that *only* learning to predict the optimal parameters with eq. (17), conditional on the input measures, and then fine-tuning with W2GN to be stable.

Advantages for Meta OT. Exploring MAML-inspired methods could further incorporate the knowledge that the model's prediction is going to be fine-tuned into the learning process. One promising direction we did not try could be to integrate some of the ideas from LEO [Rusu et al., 2018] and CAVIA [Zintgraf et al., 2019], which propose learn a latent space for the parameters where the initialization is also conditional on the input.

B.2 Neural process

The (conditional) neural process models considered in Garnelo et al. [2018b,a] can also be adapted for the Meta OT setting. In the continuous setting, this would result in a dual potential that is also conditioned on a representation of the input measures, e.g. $\psi_\varphi(x; z)$ where $z := f_\varphi^{\rm emb}(\alpha, \beta)$ is a learned embedding of the input measures that is learned with the parameters of ψ. This could be formulated as

$$\arg\min_\varphi \mathop{\mathbb{E}}_{(\alpha,\beta)\sim\mathcal{D}} \mathcal{L}(\varphi, f_\varphi^{\rm emb}(\alpha, \beta)), \tag{19}$$

where $\mathcal{L}$ modifies the model used in the loss eq. (12) to also be conditioned on the context extracted from the measures.

Challenges for Meta OT. This raises the issue on best-formulating the model to be conditional on the context. One way could be to append z to the input point x in the domain, but if ψ is an input-convex neural network, then the model would only need to be convex with respect to x and not z.

Advantages for Meta OT. A large advantage is that the representation z of the measures α, β would be significantly lower-dimensional than the parameters φ that our Meta OT models are predicting.

C Additional experimental and implementation details

We have attached the Jax source code necessary to run and reproduce all of the experiments in our paper and will open-source all of it. Here is a basic overview of the files:

```
├─meta_ot    Meta OT Python library code
│ ├─conjugate.py    Exact conjugate solver for the continuous setting
│ ├─data.py
│ ├─models.py
│ └─utils.py
├─config    Hydra configuration for the experiments (containing hyper-parameters)
├─train_discrete.py    Train Meta OT models for discrete OT
├─train_color_single.py    Train a single ICNN with W2GN between 2 images (for debugging)
├─train_color_meta.py    Train a Meta ICNN with W2GN
├─plot_mnist.py    Visualize the MNIST couplings
├─plot_world_pair.py    Visualize the spherical couplings
├─eval_color.py    Evaluate the Meta ICNN in the continuous setting
└─eval_discrete.py    Evaluate the Meta ICNN for the discrete tasks
```

Connecting to the data is one difficulty in running the experiments. The easiest experiment to re-run is the MNIST one, which will automatically download the dataset:

```
./train_discrete.py # Train the model, outputting to <exp_dir>
./eval_discrete.py <exp_dir> # Evaluate the learned models
./plot_mnist.py <exp_dir> # Produce further visualizations
```

We lastly summarize the hyper-parameters we used:

C.1 Hyper-parameters

Here we briefly summarize the hyper-parameters we used for training, which we did not extensively tune. In the discrete setting, we use the same hyper-parameters for the MNIST and spherical settings.

Table 3: Discrete OT hyper-parameters.

Name	Value
Batch size	128
Number of training iterations	50000
MLP Hidden Sizes	[1024, 1024, 1024]
Adam learning rate	1e-3

Table 4: Continuous OT hyper-parameters.

Name	Value
Meta batch size (for α, β)	8
Inner batch size (to estimate $\mathcal{L}$)	1024
Cycle loss weight (γ)	3.
Adam learning rate	1e-3
ℓ_2 weight penalty	1e-6
Max grad norm (for clipping)	1.
Number of training iterations	200000
Meta ICNN Encoder	ResNet18
Encoder output size (both measures)	256×2
Meta ICNN Decoder Hidden Sizes	[512]

D Additional color transfer results

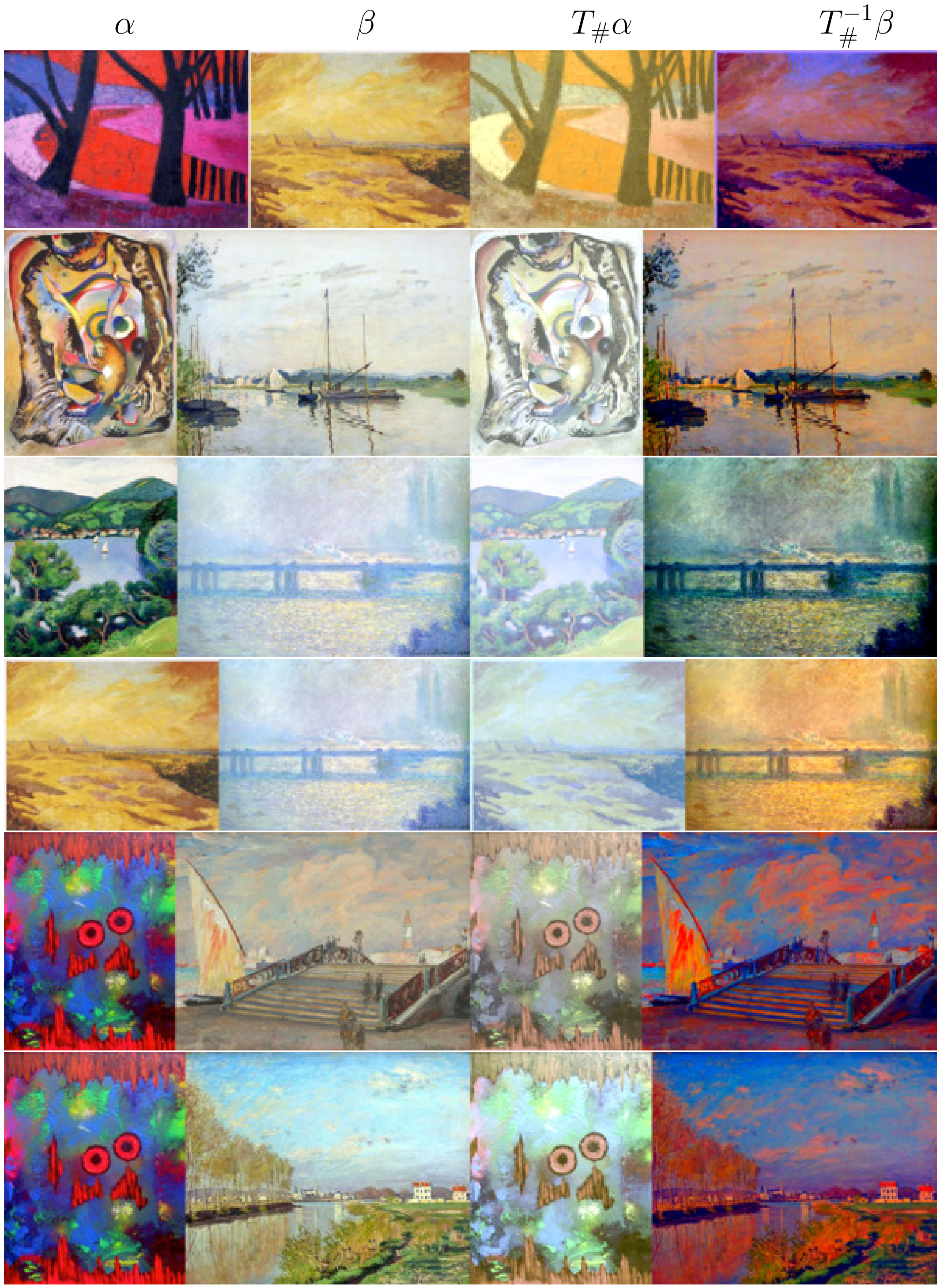

Figure 9: Meta ICNN (initial prediction)

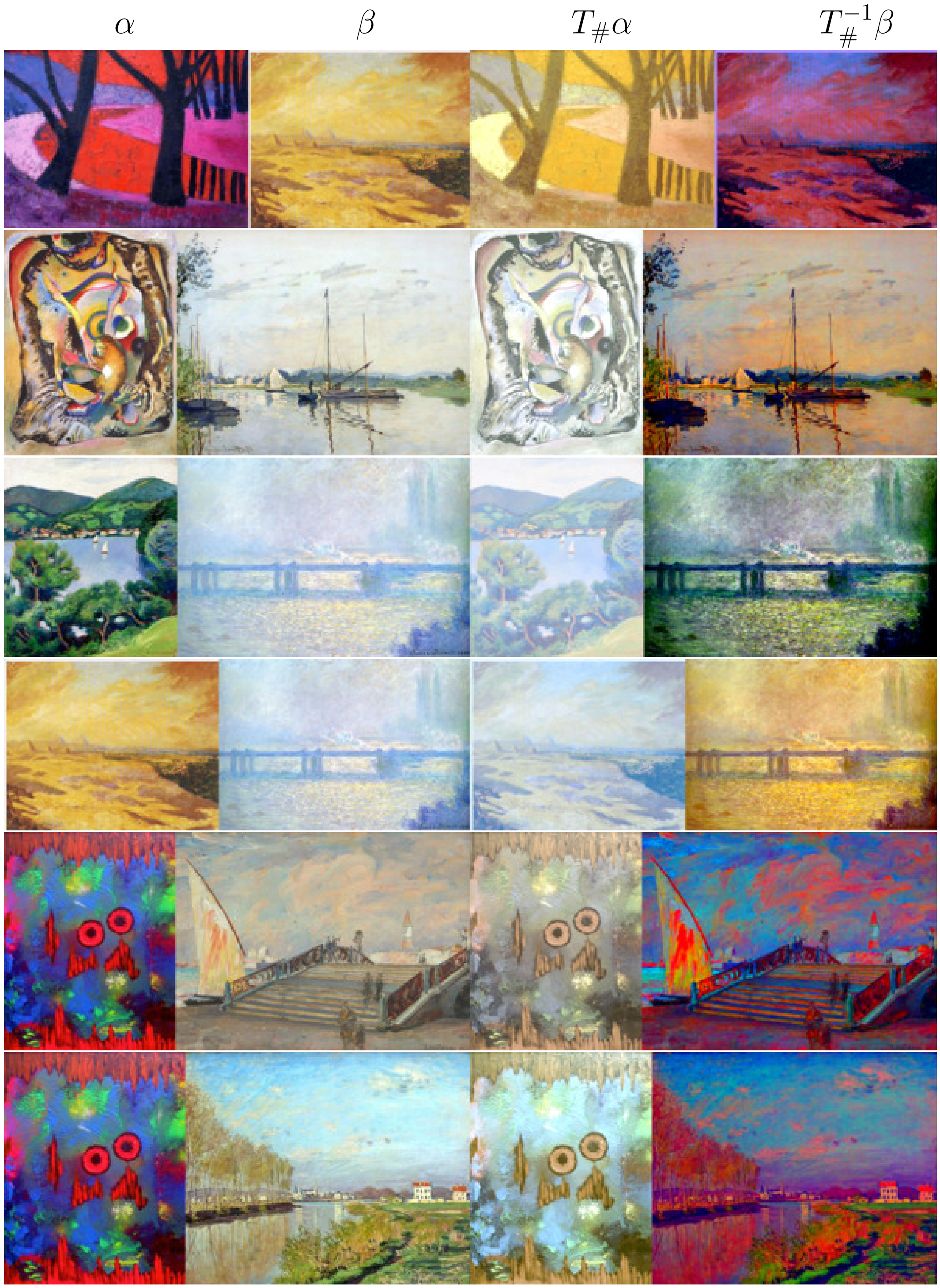

Figure 10: Meta ICNN + W2GN fine-tuning

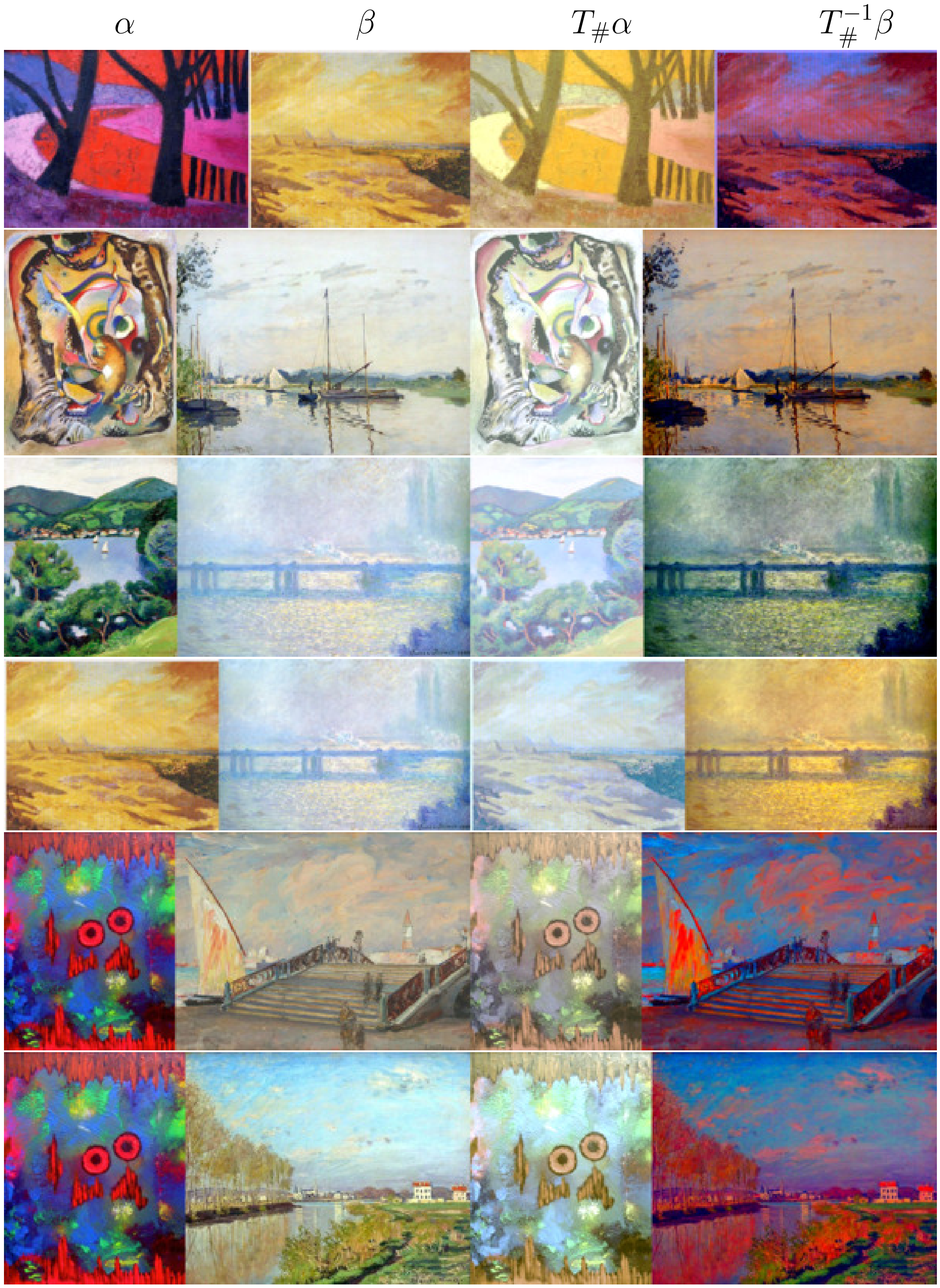

Figure 11: W2GN (final)