# Supplementary Materials

**Anonymous Authors**[1]

## 1. Effect of the regularizers on the Sinkhorn based model

In this section we show the effect of the regularizers in generalizing to unseen distributions and also how well our model learns the Translation law. We show that by just adding the regularizer that enforces the Scaling law, we get better performance in learning the Translation property and generalization to out of sample distributions than the vanilla model which has no regularizers.

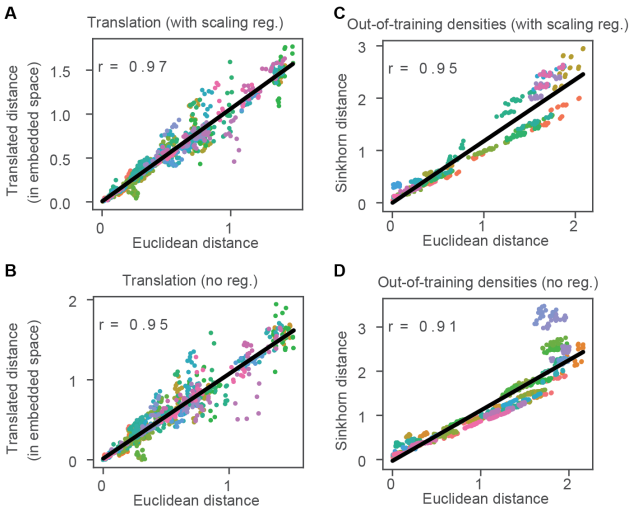


*Figure 1.* Ablation studies showing the effects of various regularizers. Pearson's r correlation to compare distances using model trained with (**A**) scaling regularizer, (**B**) no regularizer. Out of sample distribution generalization plots for models trained with (**C**) scaling regularizer, (**D**) no regularizer.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

## 2. Details on Network Architecture

We use the DeepSets architecture (Zaheer et al., 2017) which basically consists of two blocks of linear layers (with non-linearities between them) $\phi$ and $\rho$. $\phi$ consists of 3 linear layers with hidden sizes $50, 100, 36$ with non-linearity Elu (Clevert et al., 2016). We sum over the outputs of $\phi$ as in (Zaheer et al., 2017) to ensure permutation invariance before passing it to the next network $\rho$. $\rho$ consists of 3 linear layers of hidden sizes $30, 30, 10$ with an output layer of dimension 2. Elu activation is added to each hidden layer. We trained the model for $500$ epochs with Adam optimizer and learning rate $10^{-3}$.

For the CGAN based model, we use the same DeepSets architecture for the encoder. The generator consists of 4 linear layers (with non-linearities between them) with hidden sizes 128, 256, 512, 1024 and then a final linear layer with output size the same as the size of a single input sample. The discriminator has 3 linear layers with hidden sizes 512 (with non-linearities and dropout layers in between) and then a final linear layer which outputs a prediction score.

## 3. Runtime comparison between Sinkhorn and CGAN based models

To validate our claim that the CGAN based Distributional Encoder has better scaling properties compared to the Sinkhorn based model, we perform a training run time comparison experiment between the Sinkhorn and CGAN based models. Specifically, we quantify how increasing the set size will affect the training time for each model. We train each model for 5 epochs on the point cloud dataset with set sizes of $[25, 50, 100, 200, 400, 800, 1600]$, and present the *set size vs. training time* plot in Figure 2.

While the CGAN based model has an almost linear increase in training time as set size increases, the Sinkhorn based model increases exponentially. This result indicates that even though the Sinkhorn based model provides an accurate Wasserstein embedding, it becomes infeasible once the data starts to get more complex (either via increasing set size or increasing the number of features). Indeed, we observe a very similar scaling when the number of features is increased.

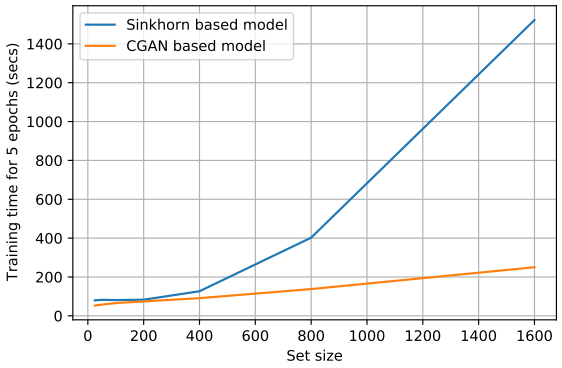


*Figure 2.* Training times for different set sizes to compare scalability of Sinkhorn and CGAN models.

## 4. Experimental results from the CGAN based model

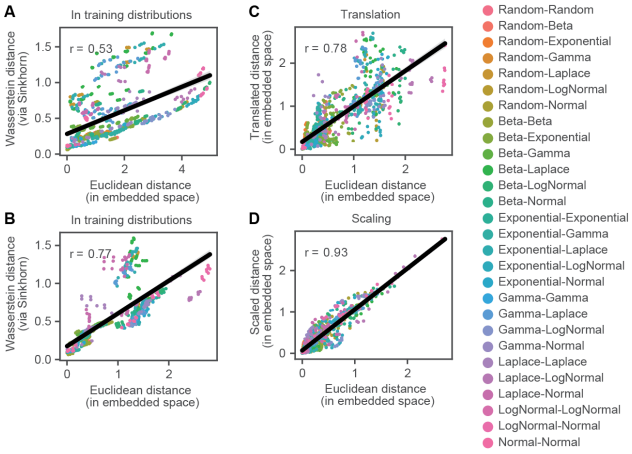


*Figure 3.* Pearson's r correlation to compare distances using CGAN model (**A**) the condition network is kept frozen (untrained) (**B**) trained CGAN model with the OT loss (**C**) CGAN Model with OT loss showing high correlation between distances of translated samples (**D**) CGAN Model with OT loss showing high correlation between distances of scaled samples.

### 4.1. Generation Experiments

We trained the CGAN based Distributional Encoder on samples from two Gaussian distributions with unit variance and means at -1 and 1 respectively. The upper subplot in Figure 4 shows histogram of samples from the two distributions. The lower subplot shows the histograms of generated samples from the encodings of the two distributions and also their interpolations. We can observe that the CGAN based model can generate the given distributions from their encoding. Also, we can see the interpolation from the midpoint resembles the barycenter of the two distributions which is a Gaussian with mean 0 and unit variance.

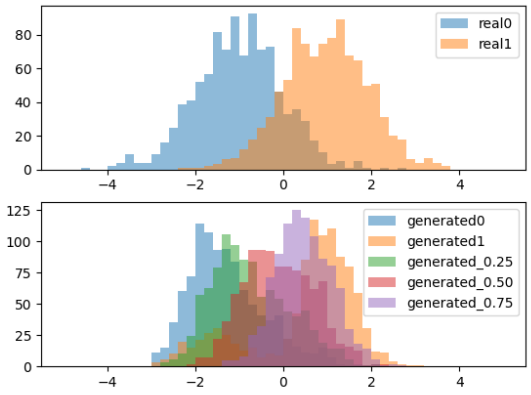


*Figure 4.* Generating from interpolations using the CGAN OT Distributional Encoder. The upper subplot shows the input data: 2 Gaussians centered at -1 and 1 respectively. The lower subplot shows histograms of their generated counterparts in the same color, as well as the histograms of the generated samples from interpolations in the encoding space.

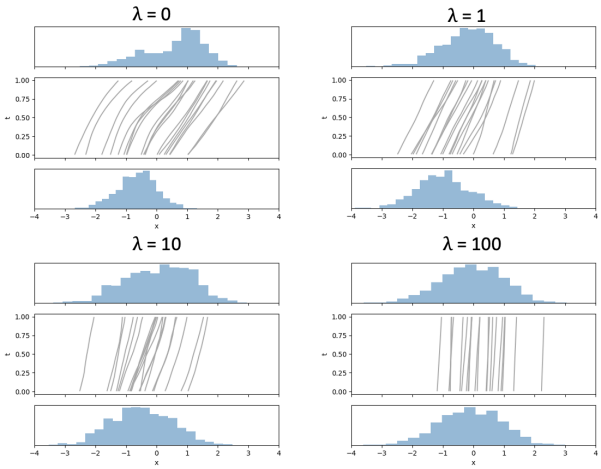


*Figure 5.* Trajectories sampled from the Distributional Encoder trained with CGAN OT loss. The model was trained to interpolate between two distributions: a Gaussian with $\mu = -1$ and $\sigma = 1$ and a Gaussian with $\mu = 1$ and $\sigma = 1$. The model was trained with different OT loss regularization ($\lambda$). Shown are trajectories sampled between the two distributions in the generated space, as well as the start and end distributions (histograms). Increasing $\lambda$ results in stronger OT and thus straighter lines, but increasing it too much breaks the generator.

In Figure 5, we show 20 trajectories between the two Gaussians. Each of these trajectories is obtained by fixing the noise vector and interpolating between the two encodings using 100 steps and generating from cGAN conditioned on the interpolations. We observe the model learns to generate

in almost straight line paths between the two distributions which is the optimal transport between the two distributions. This shows the distributional encoder trained using a cGAN based loss not only learns an encoding space which preserves wasserstein properties but also generates in a way that respects optimal transport, both in an end-to-end manner. We can also observe in the figure that increasing regularization parameter $\lambda$ for OT loss leads to straighter lines but increasing $\lambda$ by too much can break down the generator.

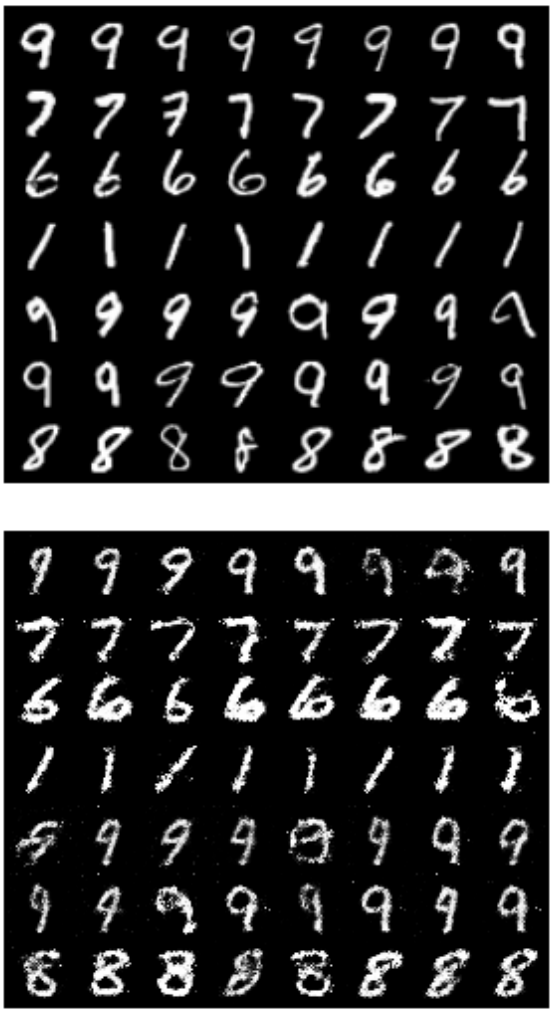

*Figure 6.* The first figure shows a batch of images from MNIST where each row is a set sampled from a particular digit which is encoded into a single vector. The second figure shows the corresponding generated images from cGAN conditioned on the encoding.

To show that the cGAN based Distributional Encoder can also generate samples from more complex distributions, we show a batch of images from MNIST data and the corresponding generated images from cGAN in Figure 6. Each class is assumed to a distribution and each row in the upper figure is a set sampled from a particular class which is fed to the set encoder. We get an encoding for each row and use cGAN to generate the corresponding row in the lower figure using the same number of randomly sampled noise vectors as the set size.

## References

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), 2016.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep Sets, 2017.