2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

LEARNING GENERAL TRANSFORMATIONS OF DATA FOR OUT-OF-SAMPLE EXTENSIONS

Matthew Amodio[1] *David van Dijk*[3] *Guy Wolf*[4] *Smita Krishnaswamy*[1,2]

[1]Yale University, Depts. of Computer Science; [2] Genetics; [3] Cardio. Medicine, New Haven, CT, USA
[4]Université de Montréal, Dept. of Math. & Stat.; Mila – Quebec AI Institute, Montréal, QC, Canada

ABSTRACT

While generative models such as GANs have been successful at mapping from noise to specific distributions of data, or more generally from one distribution of data to another, they cannot isolate the transformation that is occurring and apply it to a new distribution not seen in training. Thus, they memorize the domain of the transformation, and cannot generalize the transformation *out of sample*. To address this, we propose a new neural network called a *Neuron Transformation Network* (NTNet) that isolates the signal representing the transformation itself from the other signals representing internal distribution variation. This signal can then be removed from a new dataset distributed differently from the original one trained on. We demonstrate the effectiveness of our NTNet on more than a dozen synthetic and biomedical single-cell RNA sequencing datasets, where the NTNet is able to learn the data transformation performed by genetic and drug perturbations on one sample of cells and successfully apply it to another sample of cells to predict treatment outcome.

1. INTRODUCTION

A great deal of attention has been given to the problem of mapping one distribution to another by the deep learning community in recent years, with an emphasis towards models based on the generative adversarial network (GAN) framework [1].

Despite the amount of work on GANs, a key limitation of theirs has received considerably less attention. That limitation is that models used to learn a mapping from one distribution to another like the CycleGAN is that they are only able to operate meaningfully strictly **within the trained distribution** [2]. When the training distribution is representative of the test distribution, and when model evaluation only measures performance on points reasonably close to this distribution, this limitation may not be readily apparent. However, this becomes increasingly problematic in any application where obtaining training data for all parts of the test space is infeasible. As this problem has been increasingly emphasized recently, we propose to address it with our novel framework called a *Neuron Transformation Network* (NTNet) that can generate transformations outside the training space by isolating the signal associated with the transformation from the training data.

The key problem with previous frameworks is that they have been trained *to map from a specific source distribution to a specific target distribution*, and not trained to identify some signal as associated with the transformation and other signal as idiosyncratic to the target distribution. In a trivial example, a model trained to map samples from Uniform(0, 1) to Uniform(-1, 0) could either learn a function algebraically identical to $f(x) = -x$ or it could "specifically learn" the Uniform(-1, 0) distribution and exclusively output values near that range. In the latter case, it would perfectly model the output when given the input it was trained on, but its behavior would be unpredictable when given points outside of the Uniform(0, 1) range.

Learning to model just the signal in the data associated with the *transformation* instead of modeling the *entire training data* is not just an abstract concept, but is essential in the context of biomedical data. In clinical trials, for example, it is naive to build models that assume the patients enrolled in the trial are a representative sample of the population of interest. Also, many researchers test effects of drugs in vitro, on mice, or on a small group of volunteers, and then need the results to generalize to humans or a large population. The prospect of performing tests on the target population is infeasible, sometimes for ethical reasons and sometimes for financial reasons. Our goal is to present a deep learning framework that could offer a key benefit in this situation, by allowing researchers to make predictions on sample datasets on which they did not administer drug treatments.

We achieve this goal by specifically modeling the transformation associated with the *treatment* in the biological setting. The NTNet is based on an autoencoder framework which learns encodings of both an in-sample source (before treatment) distribution and in-sample target (after treatment) dis-

Work supported by IVADO (l'institut de valorisation des données) [*G.W.*]; Chan-Zuckerberg Initiative grants 182702 & CZF2019-002440 [*S.K.*]; and NIH grants R01GM135929 & R01GM130847 [*G.W., S.K.*]. The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies. Correspondence to: Smita Krishnaswamy < smita.krishnaswamy@yale.edu>

978-1-7281-6662-9/20/$31.00

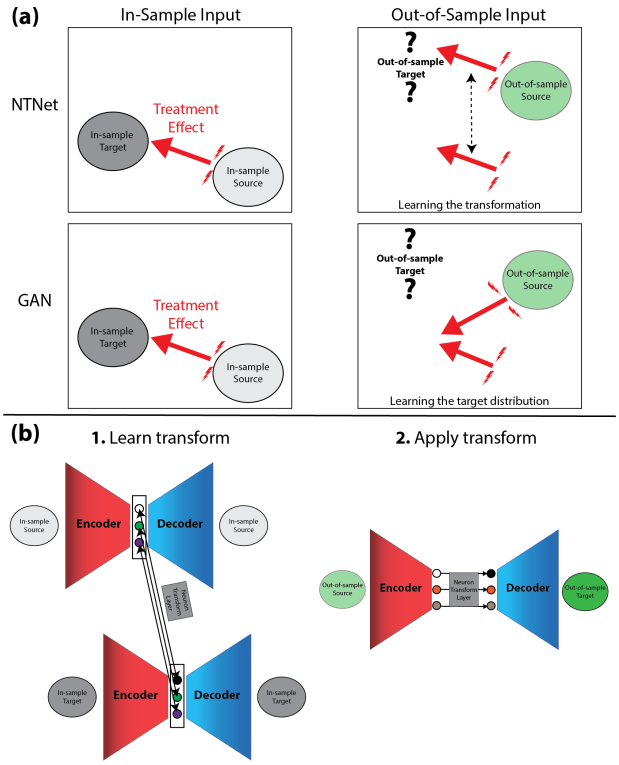

Fig. 1. (a) Learning generalized transformations requires going beyond learning to map to a specific target data distribution. (b) The Neuron Transform Network (NTNet) architecture. The network consists of a standard encoder-decoder pair with our novel Neuron Transform layer in the latent space learned by the model.

tribution. It then extracts these encodings and performs a discretized distribution transformation that takes the source encodings to the target encodings. The transformed encodings are then cascaded from latent space to data space by the decoder. Thus the autoencoder is not trained to transform the data directly, but instead we derive a transformation based on its learned representation of before- and after-treatment data.

We motivate using the latent space of an autoencoder for this transformation by noting that previous work has provided evidence that an autoencoder learns disentangled representations of the data, allowing relatively simple latent space transformations that would otherwise be infeasible in the original ambient space [3, 4]. This property explains why latent vector arithmetic is effective. In language processing, words are embedded in a latent space where a meaningful transformation such as changing the gender of a word is a constant vector in this space. The NTNet is an extension in complexity of simple vector arithmetic, because instead of transforming a single point into another single point, it transforms an entire distribution into another distribution.

We define the terminology used throughout the manuscript here both in the context of machine learning and real-world biological language. We have a (pre-treatment) in-sample source and a (post-treatment) in-sample target, and we want to use the information implied about the treatment to predict the result of applying it to a distinct untreated, out-of-sample source. In other words, there is a transformation and four data distributions that exist in this context:

- A treatment, or transformation, we are modeling
- In-sample source: a known starting distribution
- In-sample target: the in-sample source after it has been given the treatment
- Out-of-sample source: distributed differently from the in-sample source, available at training
- Out-of-sample target: the unknown result of applying the treatment to the out-of-sample source

Given that they are derived from samples collected on different experimental subjects (mice, humans, etc.), we know the differences between the in-sample source and out-of-sample source are going to be (a) large and (b) systematic. An effective generalizable model must be robust to these challenges. By learning to only remove the signal separating the in-sample source and target in the latent space, the NTNet isolates just the effect of the transformation. Thus, when fed the out-of-sample source, the NTNet will only apply the changes truly associated with the underlying transformation, disentangled from the variation specific to the in-sample training data.

We present results on a large set of synthetic and real biomedical datasets pertaining to many different types of treatments and on a wide variety of cellular populations. Our results indicate the NTNet significantly out-performs GANs and other generative models when their learned transformations are evaluated on out-of-sample points not seen during training.

2. MODEL

Let $S \in \mathbb{R}^{n_S \times d}, T \in \mathbb{R}^{n_T \times d}$ represent d-dimensional in-sample source and in-sample target distributions, known inputs and outputs of a treatment with n_S and n_T observations. Let $X \in \mathbb{R}^{n_X \times d}$ be an out-of-sample source distribution with n_X observations, an input to a treatment with an unknown output. We seek a transformation such that

1. when applied to S it produces a distribution approximately equivalent to T, and
2. when applied to X it only induces variation that transforms S to T.

While GANs learn a transformation with the first property, they fail at the second property due to the fact that T is the

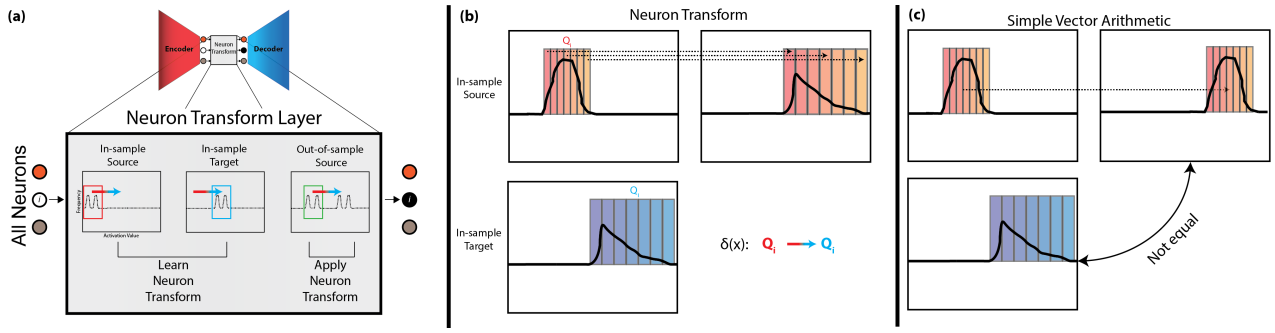

Fig. 2. (a) A schematic of the Neuron Transform layer. The NTNet is trained with all distributions together, and then after training each neuron's activations are decomposed into the three separate distributions. The transform is learned on in-sample source and target distributions and applied to out-of-sample source distribution. (b) The Neuron Transform operation, which is a discretized quantile shift, and is performed on percentiles of each neuron's distribution of activations. (c) Simple vector arithmetic would be unable to align these distributions.

only target data we have for training, and thus the generator learns to push data towards T. Therefore, instead of learning a complex transformation parameterized by a neural network, we learn a simpler transformation on a *space learned by a neural network* (summarized in Figure 1).

We first train an encoder/decoder pair E/D to learn a non-linear mapping of the data into a neuron space decomposed into high-level features such that it can also decode from that space, i.e., the standard autoencoder objective L:

$$L(S,T,X) = \text{MSE}\left[(S,T,X), D(E(S,T,X))\right]$$

where MSE is the mean-squared error. The autoencoder is trained on all three data distributions S, T, and X and thus learns to model their joint manifold. Then, without further training, we separately extract the internal activations of the n-dimensional encoder output for inputs from S and from T.

The NTNet transforms its out-of-sample input by applying the encoder, our NeuronTransform layer, and then decoding the edited internal representation: $\hat{X} = D(NT(E(X)))$. Before detailing the NeuronTransform layer, we emphasize that the NTNet is run in two distinct modes, one for training and one for inference. During training, it is a standard autoencoder and the NeuronTransform layer is simply replaced by the identity function. In this setting, the output of the NTNet is untransformed data and is only used in the MSE training objective. During inference, however, the output of the encoder h is transformed by the NeuronTransform layer into $\hat{h}$, and the decoder cascades this modified latent space representation through its learned nonlinear mapping, eventually resulting in a transformed point in the data space.

The NeuronTransform layer learns a delta function for each neuron, which is piecewise linear over the percentiles of the source distribution. Let $E(S) = \{a_1^S, ..., a_n^S\}$ where a_i^S is the distribution of activations for neuron i for the distribution S. Let $E(T) = \{a_1^T, ..., a_n^T\}$ and $E(X) = \{a_1^X, ..., a_n^X\}$ be defined analogously. Then, the transform is defined as

$$NT(a_{ij}^X) = a_{ij}^X + \delta(a_{ij}^X) \tag{1}$$

where a_{ij}^X is the activation for neuron i and point j from the out-of-sample distribution X. The $\delta(a_{ij}^X)$ function calculates the appropriate edit by locating where a_{ij}^X is in the distribution of a_i^S:

$$\delta(a_{ij}^X) = \left(\frac{a_{ij}^X - p_{ik}^S}{p_{i(k+1)}^S - p_{ik}^S} \cdot (p_{i(k+1)}^T - p_{ik}^T)\right) + p_{ik}^T - a_{ij}^X$$
$$k := \text{index satisfying } p_{ik}^S < a_{ij}^X < p_{i(k+1)}^S \tag{2}$$

where p_{ik}^S, p_{ik}^T are the k^{th} percentiles of p_i^S and p_i^T, respectively. This δ function defines a discretized, monotonic transform that transports the histogram of activations for S into the histogram of activations for T.

Figure 2 graphically illustrates the NeuronTransform layer. Figure 2(a) shows the distribution of activations for a given neuron i being separated out into histograms for the in-sample source, in-sample target, and out-of-sample source, respectively. The δ function is learned to transport the histogram of activations for the in-sample source to the histogram of activations for the in-sample target, and then applied to the out-of-sample source. Figure 2(b) illustrates the δ function, which discretizes the distributions and then learns a shift for each quantile separately.

Using this operation provides several important advantages. First, by using a transformation as expressive as this one that operates on whole distributions, we can approximately align any two distributions without making simplifying assumptions. For example, a transformation that assumes that each distribution is Gaussian will fail if one is bimodal. This is shown in Figure 2(b), where the in-sample source distribution has a different shape from the in-sample target distribution, so no single constant shift will align the two (in other words, simple vector

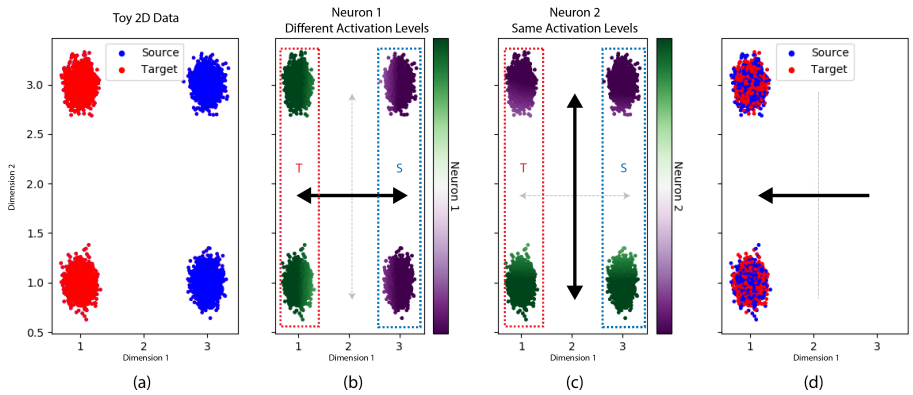

Fig. 3. (a) A toy example with a two-neuron embedding layer. (b) Neuron 1 isolates an axis of variation *separating* S and T. Since the distributions of activations are completely non-overlapping for the source and target, they will be aligned by the transform layer and this axis of variation is removed by the transformation. (c) Neuron 2 isolates an axis of variation *in common* to S and T. Since the distributions of activations are already identical between the source and target, the transform layer produces no change when applied to this neuron, and this axis of variation is unaltered by the NTNet. (d) The result is that the source and target are fully aligned.

arithmetic would not work here). Also, the monotonicity of the transform guarantees it will not introduce artifacts such as flipping the identity of a point by taking it from a low quantile pre-transformation to a high quantile post-transformation.

Additionally, the NTNet benefits from a robust, stable training regime. It trains an autoencoder to learn how the source and target are encoded, and then after training, it derives the transformation implied by the difference between the two. The robustness of this training regime is in stark contrast to the notoriously tricky to train GANs. Adversarial discriminators suffer from oscillating optimization dynamics, uninterpretable losses, and most debilitatingly, mode collapse [5, 6].

3. EXPERIMENTS

In this section, we compare our NTNet to alternative transformation methods on extrapolation tasks across a wide spectrum of artificial and real datasets.

3.1. Methods

We compare to several alternative methods for generating transformations:

- GAN: a standard GAN with in-sample source as input to the generator and in-sample target as input to the discriminator
- CycleGAN: a GAN with added cycle-consistency loss [2]
- OutofSampleCycleGAN: a CycleGAN with both in-sample and out-of-sample source distributions as input to the generator and the in-sample target as input to the discriminator (since the out-of-sample source, but not the out-of-sample target, is available at training time)
- IdentityCycleGAN: CycleGAN with same-domain identity constraint, commonly used with the cycle-consistency constraint
- ResNetCycleGAN: a CycleGAN parameterized as a residual network, to illustrate that generating residuals adversarially is not the same thing as our editing process
- cGAN: a method we devised which uses a conditional GAN creatively to attempt out-of-sample generalization by learning a conditional generator with the labels for the in-sample source data receiving label [0 0], the in-sample target data receiving label [1 0], the out-of-sample source data (available at training) receiving label [0 1], and the predicted out-of-sample target data (not available at training) being defined as the output with label [1 1]
- scGen: a variation autoencoder (VAE) approach that uses a single, constant shift as a latent transformation for all points, everywhere in the space [7]

In each of the experiments, we have one pair of known source and target distributions, plus a third distribution which is a source. We withhold the target for this last distribution and exclusively use it to measure the accuracy of the learned transformation on the first pair of distributions.

To evaluate performance on the task of generalization, we utilize two measures of accuracy. First, for our artificial data experiments, we have known ground truth values for the post-transformation points, since we ourselves created the transformation function. In those experiments, we thus use

Experiment	Treatment	In-sample population	Out-of-sample population
Lupus	Stimulus with Inf-β	CD4+ T Cells, CD8+ T cells, CD14+ Monocytes, FCGR3A+ Monocytes, NK Cells	B cells, Dendritic cells
Mouse	Stimulus with Hpoly	Endocrine cells, Goblet cells, Stem cells, Tuft cells	Enterocytes, TA cells
T Cell	T cell activation	Patient A bone marrow, lung, lymph nodes blood	Patient B bone marrow, lung, lymph nodes blood
ALL	Stimulus with drug Das	Basal population	Bez-stimulated population
Dengue	Stimulus with dengue	Patient A blood	Patient B blood

Table 1. A summary of the treatment, in-sample population, and out-of-sample population for each of the biological applications.

a pointwise mean-squared error score between each model's predicted values and the true values. In the real-world cases, however, we only have ground-truth distributions, with no known pointwise correspondences. This is because the measurements are destructive, and thus the individual cells in each population are not the same (though they are draws from the same distribution) and there is no pointwise correspondence, only a distributional correspondence between the populations. Thus, we measure the accuracy of each model's predicted distribution with the ground truth distribution using the maximum mean discrepancy (MMD) between the two.

Many models have sensitive hyperparameters or architecture design choices that must be tuned for each dataset. The NTNet, however, uses the same network architecture for all experiments in the paper, showing the robustness and versatility of the model. That architecture consists of seven layers: a 400-neuron fully connected (400FC) layer, a 200FC layer, a 100FC layer, our novel Neuron Transform layer, a 200FC layer, and a 400FC layer. This final layer connects to the output layer, whose dimensionality depended on the space of the original data in each experiment. Each FC layer used a leaky ReLU activation, with no additional normalization techniques used. The optimizer was Adam with momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$. Training was done with a fixed batch size 50 in all cases. All models were run on a single GPU and written in Tensorflow.

3.2. Artificial Data

We first use a trivial example with 2D data to illustrate how the NTNet decomposes the overall variation into variation "separating the source and target distributions" from all other variation in the data. In Figure 3, we show a simple two-dimensional mixture of Gaussians with two belonging to the source distribution and two belonging to the target distribution. We use an autoencoder with just two neurons in the bottleneck layer.

In the NTNet's learned representation for this toy dataset, the first neuron separates the source and target along the x-axis, while the second neuron is the same for both the source and the target (variation along the y-axis). The neuron transform layer aligns the source and target activations for the first neuron but produces no change in the second neuron, as the two activation distributions are already identical.

Moving past this illustrative case, we now describe the synthetic data we use as a first set of quantitative experiments. We generate distributions out of various numbers of random Gaussians and subject them to a random transformation function. These transformations apply a random shift to each low-dimensional Gaussian G_i and then project them into 100-dimensional space with a random matrix to obtain a transformed Gaussian $\hat{G}_i$. In our first case (*Artificial1*), our known source consists of two Gaussians. The other source distribution (whose target is withheld and not seen during training) consists of the same two Gaussians plus a third one that is exclusively out-of-sample. Then, on datasets *Artificial2-Artificial4*, we test the models on transformations of increasing complexity, incrementing the number of Gaussians each time. On datasets *Artificial5-Artificial8*, we add further complexity to the last case by incrementing the number of out-of-sample Gaussians, as well. In each case, a random transformation matrix is generated for each Gaussian. The accuracy on out-of-sample data is reported with a mean and standard deviation across five runs (Table 2).

3.3. Biological Experiments

In this section, we feature real-world biological applications, where the transformations between the datasets represent the actual effects of important biological processes. All of the data we examine are from publicly available experiments [8, 9, 10, 11, 12].

The general setup of these experiments is that there is a public dataset whose purpose was to measure the effects of a treatment on several distinct cellular populations. We take some populations' pre-treatment and post-treatment measurements and use them to derive the transformation. We then apply the derived transformation to a different, distinct population. We compare the predicted distribution to the held-out, post-treatment measurement for this different population. The treatment and populations for each experiment are summarized in Table 1. The quantification results of these experiments are reported in Table 3. We see that in each case, the NTNet outperforms all competitors.

	Artificial1	Artificial2	Artificial3	Artificial4	Artificial5	Artificial6	Artificial7	Artificial8
NTNet	**0.017 +/- 0.006**	**0.073 +/- 0.015**	**0.098 +/- 0.019**	**0.060 +/- 0.006**	**0.052 +/- 0.006**	**0.068 +/- 0.009**	**0.063 +/- 0.012**	**0.057 +/- 0.009**
scGen	1.970 +/- 1.363	1.001 +/- 0.075	0.627 +/- 0.030	0.888 +/- 0.015	0.964 +/- 0.024	0.940 +/- 0.032	0.912 +/- 0.022	0.923 +/- 0.015
GAN	4.966 +/- 2.856	0.795 +/- 0.320	0.643 +/- 0.105	2.833 +/- 2.342	1.623 +/- 1.718	0.878 +/- 0.419	3.070 +/- 1.352	2.680 +/- 1.027
CycleGAN	0.673 +/- 0.127	0.300 +/- 0.041	0.277 +/- 0.066	0.730 +/- 0.099	0.588 +/- 0.259	0.572 +/- 0.274	0.681 +/- 0.153	0.640 +/- 0.230
OutOfSample CycleGAN	7.183 +/- 5.557	0.478 +/- 0.297	1.039 +/- 0.456	1.049 +/- 1.012	2.030 +/- 1.093	3.355 +/- 0.155	2.275 +/- 1.095	1.655 +/- 0.966
ResNet CycleGAN	0.063 +/- 0.018	0.477 +/- 0.156	0.479 +/- 0.186	0.959 +/- 1.043	0.135 +/- 0.029	0.378 +/- 0.208	0.175 +/- 0.037	0.667 +/- 0.292
Identity CycleGAN	0.061 +/- 0.019	0.213 +/- 0.021	0.317 +/- 0.096	0.261 +/- 0.018	0.297 +/- 0.098	0.310 +/- 0.052	0.273 +/- 0.035	0.384 +/- 0.259
cGAN	4.433 +/- 0.791	1.458 +/- 0.117	1.515 +/- 0.051	2.500 +/- 0.055	2.706 +/- 0.160	2.509 +/- 0.211	2.681 +/- 0.041	2.592 +/- 0.023

Table 2. MSE scores on artificial data across all models. The mean and standard deviation across five runs is reported.

	Lupus	Mouse	T Cell Activation	ALL	Dengue
NTNet	**0.016 +/- 0.002**	**0.037 +/- 0.000**	**0.258 +/- 0.016**	**0.003 +/- 0.000**	**0.045 +/- 0.004**
scGen	0.438 +/- 0.056	0.085 +/- 0.005	0.796 +/- 0.029	0.219 +/- 0.004	0.120 +/- 0.003
GAN	0.147 +/- 0.022	0.259 +/- 0.042	0.602 +/- 0.035	0.069 +/- 0.005	0.163 +/- 0.017
CycleGAN	0.115 +/- 0.008	0.133 +/- 0.018	0.418 +/- 0.046	0.061 +/- 0.004	0.119 +/- 0.012
OutOfSampleCycleGAN	0.189 +/- 0.027	0.352 +/- 0.106	0.363 +/- 0.006	0.051 +/- 0.008	0.128 +/- 0.008
ResNetCycleGAN	0.196 +/- 0.086	0.141 +/- 0.035	0.343 +/- 0.045	0.127 +/- 0.026	0.121 +/- 0.015
IdentityCycleGAN	0.262 +/- 0.027	0.128 +/- 0.006	0.416 +/- 0.014	0.077 +/- 0.002	0.142 +/- 0.027
cGAN	0.156 +/- 0.011	0.177 +/- 0.044	0.773 +/- 0.108	0.019 +/- 0.005	0.104 +/- 0.012

Table 3. MMD scores on real-world biological data across all models. The mean and standard deviation across five runs is reported.

4. DISCUSSION

In this paper, we identified the problem of learning generalizable transformations, inspired by real-world biological experimental settings. While our NTNet succeed, we note one requirement: we do need sufficient out-of-sample source data for the autoencoder to learn the full data manifold. We consider it an interesting future line of work to also be able to succeed at out-of-sample generation without this requirement. However, when this is available, we show that our NTNet results in more realistic transformations and successfully predicts the outcome of treatment effects in a wide variety of biological data.

5. REFERENCES

[1] Yi et al., "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2849–2857.

[2] Zh et al., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[3] Vincent et. al, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[4] Wang et al., "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 490–497.

[5] Li et al., "Towards understanding the dynamics of generative adversarial networks," *arXiv preprint arXiv:1706.09884*, 2017.

[6] Srivastava et al., "Veegan: Reducing mode collapse in gans using implicit variational learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 3308–3318.

[7] Lotfollahi et al., "scgen predicts single-cell perturbation responses.," *Nature methods*, vol. 16, no. 8, pp. 715–721, 2019.

[8] Kang et al., "Multiplexed droplet single-cell rna-sequencing using natural genetic variation," *Nature biotechnology*, vol. 36, no. 1, pp. 89, 2018.

[9] Haber et al., "A single-cell survey of the small intestinal epithelium," *Nature*, vol. 551, no. 7680, pp. 333, 2017.

[10] Stubbington et al., "Single-cell transcriptomics to explore the immune system in health and disease," *Science*, vol. 358, no. 6359, pp. 58–63, 2017.

[11] Anchang et al., "Drug-nem," *Proceedings of the National Academy of Sciences*, vol. 115, no. 18, 2018.

[12] Amodio et al., "Exploring single-cell data with deep multitasking neural networks," *Nature Methods*, vol. 16, pp. 1139–1145, 2019.