FIBER BUNDLE MORPHISMS AS A FRAMEWORK FOR MODELING MANY-TO-MANY MAPS

Elizabeth Coda, Nico Courts, Colby Wight, Loc Truong, WoongJo Choi, Charles Godfrey, Tegan Emerson,[*] Keerti Kappagantula, Henry Kvinge[†]
Pacific Northwest National Lab
Seattle, Washington
{first.last}@pnnl.gov

ABSTRACT

While it is not generally reflected in the 'nice' datasets used for benchmarking machine learning algorithms, the real-world is full of processes that would be best described as many-to-many. That is, a single input can potentially yield many different outputs (whether due to noise, imperfect measurement, or intrinsic stochasticity in the process) and many different inputs can yield the same output (that is, the map is not injective). For example, imagine a sentiment analysis task where, due to linguistic ambiguity, a single statement can have a range of different sentiment interpretations while at the same time many distinct statements can represent the same sentiment. When modeling such a multivalued function $f : X \to Y$, it is frequently useful to be able to model the distribution on $f(x)$ for specific input x as well as the distribution on fiber $f^{-1}(y)$ for specific output y. Such an analysis helps the user (i) better understand the variance intrinsic to the process they are studying and (ii) understand the range of specific input x that can be used to achieve output y. Following existing work which used a fiber bundle framework to better model many-to-one processes, we describe how morphisms of fiber bundles provide a template for building models which naturally capture the structure of many-to-many processes.

1 INTRODUCTION

Variation is ubiquitous in the real-world. This is especially true for physical processes where a single input to the system can produce many different possible outputs. For example, in nearly all scientific fields it is common for an experiment to yield (slightly) different results each time it is repeated, even when all controllable parameters are held fixed. On the other hand, variation in systems can also flow in the other direction. It is common to have many inputs to the system that yield a single output. For example, there are infinitely many images of cats that all map to the label 'cat'. From the perspective of machine learning (ML) we can think of processes that possess both of these properties as being many-to-many. Each input to the system may yield many different outputs and many different inputs can also yield a single output. A non-injective multivalued function underlies such processes.

In this paper we develop a deep learning architecture capable of modeling the distributions found on both images and fibers of a many-to-many map. More precisely, let $f : X \to Y$ be a many-to-many map, so that $f(x)$ and $f^{-1}(y)$ are both sets for any $x \in X$ and $y \in Y$. A probability distribution on X induces probability distributions on both $f(x)$ and $f^{-1}(y)$. Our goal is to build a model that simultaneously learns all of these distributions over the course of training. Such information may not be necessary for simple inference tasks, but modeling the distributions on both images and fibers gives information that can be critical for decision making. For example, understanding the distribution $f^{-1}(y)$ can help a user identify all possible inputs that can be applied to achieve the specific desired output y. On the other hand, being able to understand $f(x)$ for varying x can point to those x that may yield less consistent values in Y.

[*]Dr. Emerson holds joint appointments in the Department of Mathematics at Colorado State University and the Department of Mathematical Sciences at the University of Texas, El Paso.

[†]Dr. Kvinge holds a joint appointment in the Department of Mathematics at the University of Washington.

The present work was inspired by a problem in advanced manufacturing where one wants to understand all the different manufacturing processing parameter settings X that will lead to a single desirable material property $y \in Y$. If f represents the manufacturing process this amounts to understanding $f^{-1}(y)$. At the same time, certain processing parameters in X can yield less inconsistent results (more variation in $f(x)$), which is often useful information when each processing run is time-, labor-, and resource-intensive.

To achieve both of these objectives within a single model architecture we build off the work of Courts & Kvinge (2022) which proposed a fiber bundle framework to model many-to-one maps. We find that while fiber bundles offer a useful template for many-to-one maps, bundle morphisms better capture the structure required for an architecture meant to capture many-to-many maps. In the model that we propose, which we call a *Bundle Morphism Network (BMNet)*, we offload the variability intrinsic to f by lifting it to a new 'bundle morphism' $F : E_X \to E_Y$. We incorporate local trivializations into F so that it is easy to move from X to E_X and Y to E_Y. Finally, the variation that emerges when we consider either $f(x)$ or $f^{-1}(y)$ is encoded in simple distributions (e.g., 1-dimensional Gaussians) on the fibers of E_X and E_Y respectively. The local trivializations, which we build into our model using invertible neural networks, are trained to transform these distributions to the distribution on $f(x)$ and $f^{-1}(y)$.

We test BMNet on several synthetic datasets that represent many-to-many maps with a fiber bundle like flavor. We benchmark BMNet against other many-to-many models as well as their relatives such as conditional GANs (cGANs) (Mirza & Osindero, 2014) and conditional normalizing flow (cNF) models (Winkler et al., 2019). We show that BMNet generally outperforms these models on most of our synthetic datasets suggesting that the next step is to benchmark BMNet against a wider variety of more complicated real-world datasets.

2 Fiber Bundles and Bundle Morphisms

Early in the history of geometry it was recognized that one way to analyze a space is to decompose it into simpler constituent parts. The most familiar of such decompositions is the product, wherein space X is identified as the Cartesian product of two simpler spaces Y and Z, $X \cong Y \times Z$. The torus T is a simple example of a space that is (topologically) the product of two spheres, $T \cong S^1 \times S^1$. Some spaces, however, may appear to be a product space locally and yet fail to be a product space globally due to a 'twist' in the product structure. One of the simplest examples of this is the Möbius band. The notion of a fiber bundle, first introduced in Seifert (1933) and Whitney (1935), is meant to capture the essence of this phenomenon. We include a review of the definition of a fiber bundle in the appendix[1].

As with many structures within topology and geometry, it is useful to identify the maps between fiber bundles that preserve their structure. Let $\mathbf{E_1} = (E_1, B_1, Z_1, \pi_1)$ and $\mathbf{E_2} = (E_2, B_2, Z_2, \pi_2)$ be two fiber bundles. A *bundle morphism* F between total spaces $\mathbf{E_1}$ and $\mathbf{E_2}$ is a continuous map $F : E_1 \to E_2$ and continuous map on base spaces $f : B_1 \to B_2$ such the following diagram commutes.

$$\begin{tikzcd}
E_1 \arrow[r, "F"] \arrow[d, "\pi_1"'] & E_2 \arrow[d, "\pi_2"] \\
B_1 \arrow[r, "f"] & B_2
\end{tikzcd}$$

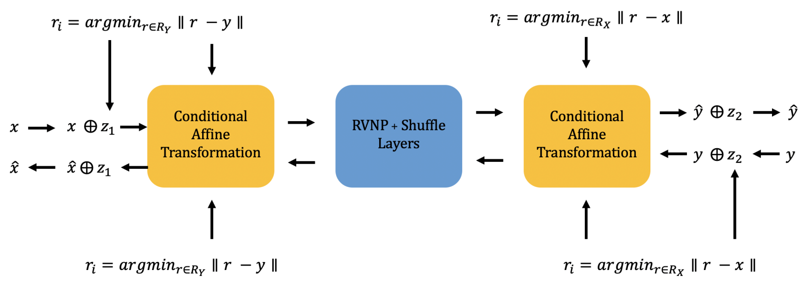
Figure 1: Bundle morphism diagram.

While it is not intended to follow this definition "on the nose", the concept of a bundle morphism informs the structure of the Bundle Morphism Network introduced in Section 3.

3 Bundle Morphism Network

Building off of BundleNet, *Bundle Morphism Network (BMNet)* is a neural network $\Phi : X \times Z_1 \times \mathcal{R}_X \times \mathcal{R}_Y \to Y \times Z_2$ where X is the input space for the task, Y is the output space, Z_1 and Z_2 are called the fibers, and $\mathcal{R}_X = \{r_1^X, \ldots, r_{n_1}^X\}$ and $\mathcal{R}_Y = \{r_1^Y, \ldots, r_{n_2}^Y\}$ are a collection of conditioning vectors representing neighborhoods of X and Y respectively. For fixed $r_i^X \in \mathcal{R}_X, r_j^Y \in \mathcal{R}_Y$

[1]The appendix is available at https://arxiv.org/abs/2203.08189.



$r_i = argmin_{r \in R_Y} \| r - y \|$
$r_i = argmin_{r \in R_X} \| r - x \|$
x
$x \oplus z_1$
$\hat{x}$
$\hat{x} \oplus z_1$
Conditional Affine Transformation
RVNP + Shuffle Layers
Conditional Affine Transformation
$\hat{y} \oplus z_2$
$\hat{y}$
$y \oplus z_2$
y
$r_i = argmin_{r \in R_Y} \| r - y \|$
$r_i = argmin_{r \in R_X} \| r - x \|$

Figure 2: A diagram outlining the architecture of Bundle Morphism Network.

we write $\Phi_{i,j} := \Phi(-,-,r_i^X,r_j^Y)$. $\Phi_{i,j}$ is invertible (we use the `FrEIA` package (Ardizzone et al., 2021) to build Φ in our experiments). For clarity we refer to the process of running the model in the direction $X \times Z_1 \to Y \times Z_2$ as the *forward direction* and running the model in the direction $Y \times Z_2 \to X \times Z_1$ as the *reverse direction*. Z_1 and Z_2 are assumed to follow user chosen distributions $\mathcal{D}_1$ and $\mathcal{D}_2$. In the experiments in this paper we choose to use the uniform distribution on the circle when sampling from Z_1 and Z_2 (justification for this was given in Courts & Kvinge (2022)).

Elements of $\mathcal{R}_X$ and $\mathcal{R}_Y$ consist of representatives of neighborhoods in the support of the training data input distribution (in X) and the training data output distribution (in Y) respectively. Given a training set $D \subset X \times Y$, we cluster both the input from X and the labels from Y and let $\mathcal{R}_X$ and $\mathcal{R}_Y$ be the cluster centroids. Described in the forward direction, the model consists of a conditional affine transformation conditioned on elements of $\mathcal{R}_Y$, several RVNP blocks and coordinate permutation layers (Dinh et al., 2016), and an additional conditional affine transformation conditioned on elements of $\mathcal{R}_X$. These invertible conditional layers correspond to the local trivializations for each "bundle".

During training, a batch consists of training examples $\{(x_i, y_i)\}$ such that all x in the batch belong to the same cluster with center $r_i^X \in R_X$ and all y in the batch belong to the same cluster with center $r_j^Y \in R_Y$. For each x, we sample a z_1 from distribution $\mathcal{D}_1$ on Z_1. To run the model in the forward direction, we condition on r_j^Y and r_i^X and obtain $(\hat{y}, \hat{z}_2) = \Phi(x, z_1, r_i^X, r_j^Y)$. To run the model in the reverse direction, we sample z_2 and obtain $(\hat{x}, \hat{z}_1) = \Phi^{-1}(y, z_2, r_i^X, r_i^Y)$. Once we have a collection of such $\hat{y}$, which we denote by $\hat{S}$, and y, which we denote by S, the loss in the forward direction is

$$\mathcal{L}_{\text{forward}}(\hat{S}, S) = \frac{1}{|\hat{S}|} \sum_{\hat{s} \in \hat{S}} \min_{s \in S} ||\hat{s} - s||^2 + \frac{1}{|S|} \sum_{s \in S} \min_{\hat{s} \in \hat{S}} ||\hat{s} - s||^2.$$

$\mathcal{L}_{\text{reverse}}$ is defined analogously. This is a symmetric version of the mean-squared minimum distance, used to ensure that the learned distribution covers the entirety of the training distribution. Notice that this is different from the loss function in Courts & Kvinge (2022). We found that applying an analogous version of the loss found in that paper was unstable when applied to our problem. The detailed training algorithm is outlined in Algorithm 1.

At inference time, we will only have access to one of x or y. If we wish to run the model in the forward direction and generate the distribution of $f(x)$, we need a conditioning vector $r_j^Y \in \mathcal{R}_Y$, which we cannot assume we have since we do not have a y value paired with x. To get around this problem, we find the k nearest neighbors of x from the training set and randomly sample one (x', y'). We use the corresponding y' from (x', y') to determine r_j^Y. The detailed inference algorithm is outlined in Algorithm 2. Running inference in the reverse direction is analogous.

4 Experiments

We evaluate on three synthetic datasets. The first dataset, *Torus-to-circle I*, is illustrated in Figure 3 and consists of a map from the torus to a circle. In the forward direction, (Figure 3, center), each slice of the torus maps to a random point sampled uniformly from an interval of the circle centered

Algorithm 1 Training Procedure

Input: Training dataset $D = \{(x_i, y_i)\}$, prior distributions $\mathcal{D}_1$ and $\mathcal{D}_2$
Output: Trained model Φ with cluster centers $\mathcal{R}_X$ and $\mathcal{R}_Y$ and updated priors $\mathcal{D}_1$ and $\mathcal{D}_2$

```
 1: Cluster the training data (in $X$) to obtain $n_1$ cluster centers $\mathcal{R}_X$
 2: Cluster the training data (in $Y$) to obtain $n_2$ cluster centers $\mathcal{R}_Y$
 3: for each epoch do
 4:     for $r_i^X \in \mathcal{R}_X$ and $r_j^Y \in \mathcal{R}_Y$ do
 5:         Construct $D_{ij} = \{(x, y) | r_i^X = \arg\min_{r \in \mathcal{R}_X} |r - x|, r_j^Y = \arg\min_{r \in \mathcal{R}_Y} |r - y|\}$
 6:         $X = \text{proj}_x(D_{ij})$ (set projection to 1st coordinate)
 7:         $Y = \text{proj}_y(D_{ij})$ (set projection to 2nd coordinate)
 8:         Sample $Z_1 \sim \mathcal{D}_1^i$ and $Z_2 \sim \mathcal{D}_2^j$
 9:         $(\hat{Y}, \hat{Z}_2) = \Phi(X, Z_1, r_i^X, r_j^Y)$
10:         $(\hat{X}, \hat{Z}_1) = \Phi^{-1}(Y, Z_2, r_i^X, r_j^Y)$
11:         $\mathcal{L} = \mathcal{L}_{forward}(\hat{Y}, Y) + \mathcal{L}_{reverse}(\hat{X}, X)$
12:         Update the weights of $\Phi$
13:         Update the parameters of the prior distributions $\mathcal{D}_1^i$ and $\mathcal{D}_2^j$
14:     end for
15: end for
```

Algorithm 2 Inference Procedure: Forward Direction

Input: Input x, training dataset $D = \{(x_i, y_i)\}$, trained model Φ, with neighborhood centers $\mathcal{R}_X$ and $\mathcal{R}_Y$ and priors $\mathcal{D}_1$ and $\mathcal{D}_2$
Output: A set S of n samples from the distribution $f(x)$

```
 1: $S = \{\}$
 2: $r_i^X = \arg\min_{r \in \mathcal{R}_X} |r - x|$
 3: Construct a dataset $D_x = \{(x_i, y_i) | (x_i, y_i) \in D$ and $x_i \in \text{Neigh}(x)\}$
 4: while $|S| < n$ do
 5:     Sample $y' \sim \text{proj}_y(D_x)$
 6:     $r_j^Y = \arg\min_{r \in \mathcal{R}_Y} |r - y'|$
 7:     Sample $z \sim \mathcal{D}_1^i$
 8:     $(\hat{y}, \hat{z_2}) = \Phi(x, z, r^X, r^Y)$
 9:     $S = S \cup \{\hat{y}\}$
10: end while
```

on the projection of that slice. The second dataset, *Torus-to-circle II* (Figure 4), similarly maps the torus to the circle. In the forward direction a slice of the torus with axis of revolution angle θ is projected to one of two points on the circle with angles $\theta/2$ or $\theta/2 + \pi$, with equal probability. The third dataset, *Möbius-to-circle*, is very similar to *Torus-to-circle I*, with the torus replaced by a Möbius band (Figure 5) . More details on each dataset can be found in the appendix.

We compare BMNet to two other many-to-many models, Augmented CycleGAN (AugCGAN) (Almahairi et al.) and Latent Normalizing Flows for Many-to-Many Cross-Domain Mappings (LNFMM) (Mahajan et al., 2020). Additionally, we include a pair of cGAN models (Mirza & Osindero, 2014) and a pair of cNF models (Winkler et al., 2019). Note that in order to adapt cGANs and cNFs for many-to-many problems, we train one cGAN and cNF for the forward direction and one for the backward direction. We follow the evaluation procedure proposed in Courts & Kvinge (2022). In the forward direction, to evaluate the global distribution, we generate 5000 points and compare to 5000 points from the true distribution. To evaluate local distributions, we sample 15 points U from X and for each $x \in U$, use the model to generate 200 points in Y and compare this to the true distribution defined by the dataset. Evaluation of the backward direction is analogous. To compare true and reconstructed distributions we use a range of different metrics, including the 1-Wasserstein metric, mean squared minimum distance (MSMD), maximum mean discrepancy (MMD), and KL divergence. To obtain confidence intervals, we train each model 5 times and report the mean with 95% confidence intervals.

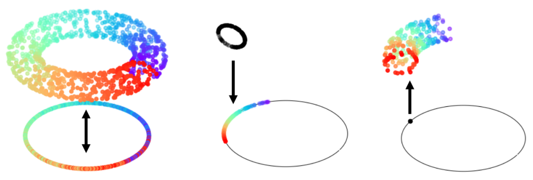


Figure 3: The *Torus-to-circle I* many-to-many dataset (left). Each torus slice maps to a uniformly sampled random point on an interval of the circle (center).

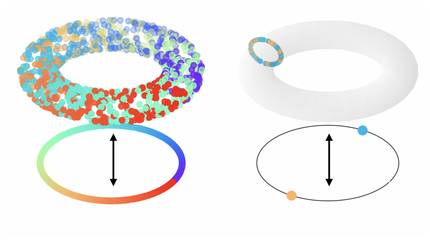


Figure 4: The *Torus-to-circle II* dataset. A point at angle θ of the torus maps to one of two points on the circle, $\theta/2$ and $\theta/2 + \pi$ (right) with equal probability.

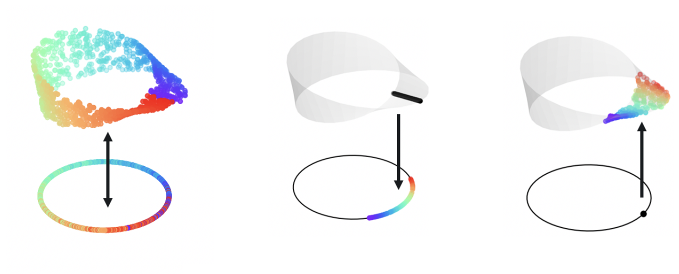


Figure 5: The *Möbius-to-circle* dataset. Each möbius slice maps to a uniformly sampled random point on an interval of the circle (center).

4.1 Results

As can be seen in Table 1, at the global level our model is the best performing model in terms of the Wasserstein-1 distance on most of the datasets. While AugCGAN performs well at the global level, as seen in Table 2, this many-to-many model does not learn the local distributions. The cNF model also generally performs well, particularly at the local level, though requires training a separate model in each direction and does not learn the discrete nature of the *Torus-to-circle II* dataset in the forward direction. This is depicted in the appendix.

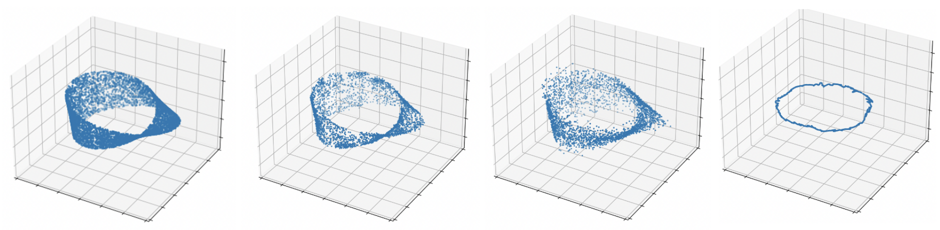


Figure 6: Target global möbius distribution (left) and global distributions generated by our model (left of center), AugCGAN (right of center), and LNFMM (right) for the *Möbius-to-circle* dataset in the reverse direction.

To give a flavor of the difference in reconstructions for models explicitly designed to model many-to-many processes, the reader should consult Figures 6 and 7 which depict the global and local distributions of the *Möbius-to-circle* dataset reconstructed by BMNet, an AugCGAN, and a LNFMM (left to right). Locally, while the AugCGAN does not learn the correct distribution and LNFMM struggles to learn the full range of possible inputs to achieve a given y, our model is able to recreate a reasonable approximation of sections of the möbius band. In general, we see this pattern repeated across datasets in both the forward and backward direction. This suggests to us that the structure of BMNet makes it more capable of handling the problem set forth in this short paper. Full results for each dataset, all models, all metrics, and further visualizations are available in the appendix.

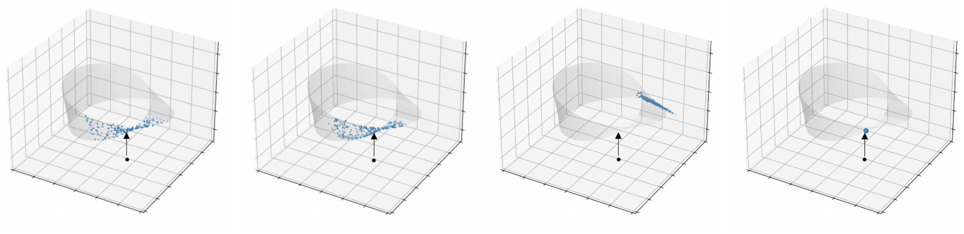


Figure 7: Target local möbius distribution (left) and local distributions generated by our model (left of center), AugCGAN (right of center), and LNFMM (right) for the *Möbius-to-circle* dataset in the reverse direction.

Table 1: The Wasserstein-1 ($\times 10^{-2}$) global metric on all datasets and models.

	BMNet	AugCGAN	LNFMM	cGAN	cNF
Torus-to-circle I: Fwd	1.91 ± 0.34	9.80 ± 0.88	15.01 ± 5.55	4.38 ± 1.61	$\mathbf{1.28 \pm 0.16}$
Torus-to-circle I: Rev	$\mathbf{4.38 \pm 0.57}$	19.13 ± 3.51	25.95 ± 3.16	28.96 ± 3.72	7.10 ± 0.58
Torus-to-circle II: Fwd	$\mathbf{0.38 \pm 0.10}$	0.45 ± 0.05	23.36 ± 5.94	6.29 ± 0.84	27.54 ± 5.49
Torus-to-circle II: Rev	$\mathbf{3.51 \pm 0.17}$	9.48 ± 1.0	20.54 ± 0.35	48.84 ± 12.01	8.13 ± 0.70
Möbius-to-circle: Fwd	1.85 ± 0.39	$\mathbf{0.68 \pm 0.11}$	6.50 ± 2.62	8.49 ± 1.54	2.58 ± 0.59
Möbius-to-circle: Rev	$\mathbf{2.33 \pm 0.19}$	2.37 ± 0.16	23.89 ± 11.95	103.7 ± 4.1	3.36 ± 0.05

Table 2: The Wasserstein-1 ($\times 10^{-2}$) local metric on all datasets and models.

	BMNet	AugCGAN	LNFMM	cGAN	cNF
Torus-to-circle I: Fwd	9.30 ± 1.26	119.9 ± 10.4	20.31 ± 2.22	20.33 ± 2.01	$\mathbf{5.68 \pm 0.72}$
Torus-to-circle I: Rev	$\mathbf{12.01 \pm 0.95}$	123.6 ± 9.9	46.23 ± 0.71	37.48 ± 1.80	12.43 ± 0.28
Torus-to-circle II: Fwd	16.05 ± 4.38	127.1 ± 2.8	31.71 ± 2.29	$\mathbf{9.37 \pm 1.01}$	32.86 ± 2.26
Torus-to-circle II: Rev	$\mathbf{4.49 \pm 0.54}$	116.1 ± 13.7	23.58 ± 0.15	56.39 ± 5.73	10.36 ± 0.43
Möbius-to-circle: Fwd	9.48 ± 0.98	87.3 ± 13.41	16.67 ± 2.08	42.63 ± 3.71	$\mathbf{6.08 \pm 0.93}$
Möbius-to-circle: Rev	8.77 ± 0.79	94.32 ± 14.2	42.78 ± 2.03	117.17 ± 1.84	$\mathbf{8.60 \pm 0.38}$

5 CONCLUSION

Many-to-many processes are common in nature. While there are a range of deep learning-based frameworks that can be used to solve simple tasks related to these processes, network architectures for more comprehensive modeling have until now remained limited. Guided by the concept of a bundle morphism, in this paper we introduce the first model architecture explicitly designed to capture the more nuanced aspects of many-to-many processes, providing the capability to model not only the distribution as a whole, but also the distributions of both the image of points and their fibers. The appendix is available at https://arxiv.org/abs/2203.08189.

ACKNOWLEDGMENTS

This research was supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative via the Laboratory Directed Research and Development (LDRD) investments at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830.

REFERENCES

Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data.

Lynton Ardizzone, Jakob Kruse, Peter Steinbach, Till Bungert, Peter Sorrenson, Alexandre René, and Johnson Yue. Framework for easily invertible architectures, 2021. URL `https://github.com/VLL-HD/FrEIA`.

Nico Courts and Henry Kvinge. Bundle networks: Fiber bundles, local trivializations, and a generative approach to exploring many-to-one maps. 2022.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016. URL `http://arxiv.org/abs/1605.08803`.

Shweta Mahajan, Iryna Gurevych, and Stefan Roth. Latent normalizing flows for many-to-many cross-domain mappings. 2020.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

H. Seifert. Topologie Dreidimensionaler Gefaserter Räume. *Acta Mathematica*, 60(none):147 – 238, 1933. doi: 10.1007/BF02398271. URL `https://doi.org/10.1007/BF02398271`.

Hassler Whitney. Sphere-spaces. *Proceedings of the National Academy of Sciences*, 21(7):464–468, 1935. ISSN 0027-8424. doi: 10.1073/pnas.21.7.464. URL `https://www.pnas.org/content/21/7/464`.

Christina Winkler, Daniel E. Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019.