Reversible Deep Generative Models for Climate Informatics

David Bayani Jeremy Cohen Elan Rosenfeld

Abstract

A popular emerging approach in deep generative modeling consists of *normalizing flows*, models that learn a sequence of smooth invertible functions from a simple base distribution to a complex target distribution. One exciting possible application of such models is in the nascent field of Climate Informatics, particularly for medium- to long-term weather prediction. In this paper we conduct the first experiments on the feasibility of reversible deep generative models applied to the task of density estimation of climate data. We find that even the simplest normalizing flow models successfully outperform several baselines and show serious potential for future applications in weather prediction and other aspects of Climate Informatics.

1. Introduction

A popular emerging approach in deep generative modeling consists of *normalizing flows*, a technique that involves learning a sequence of smooth invertible functions that map a simple product distribution into an arbitrarily complex target distribution learned from the data. These methods ensure that the log-likelihood is easy to evaluate, facilitating training. Simultaneously, by sampling from the known base distribution and transforming via the inversion of the learned function, generating samples is made simple as well. These models are extremely flexible and powerful, and they have very exciting future potential.

One possible application is in the nascent field of Climate Informatics. Historical weather prediction is based on simple ensembles that jointly utilize statistical regression models and dynamical models that simulate weather outcomes via partial differential equations (PDEs). Climate Informatics is the study of how to improve prediction techniques by incorporating the rapidly developing wealth of machine learning research. While some of these methods lack the explainability of simpler models, superior predictive performance is often worth the trade-off.

Medium- to long-term weather prediction is a notoriously difficult problem. To begin with, there is a severe shortage of climate data, as we have access to only one reality and no method of obtaining counterfactuals. This is partially addressed via simulations, but these simulators are extremely sensitive to fluctuations in initial conditions and frequently are forced to sacrifice accuracy in favor of computability. This shortage of climate data means many popular models are a poor choice for this specific problem. In addition to this, most of the work on these models has been done with the assumption of *stationarity*; while this is reasonable for modeling things like language or natural images, it is certainly not the case with climate.

In general, work on time series prediction attempts to model the autoregressive nature of the data. In contrast, deep generative models and other techniques such as GP regression typically consider each training and test sample to be an i.i.d. draw from an unknown distribution or function. As such, these models are probably not the best suited for regression; they are, however, much more adept at modeling unknown and complex distributions.

With deep generative models, we can better learn underlying representations of the developing Earth climate. This is particularly useful for three reasons: by exploring the learned representations in latent space, we can discover new predictive features of Earth's climate system that can be used to inform the simpler physics and regression models; evaluating the log-likelihood of the output of other models can be used to assess their quality, giving a quantitative benchmark against which to compare various physical simulations; and we can sample from these models to generate a much larger set of plausible observations[1].

2. Background & Related Work

2.1. Climate Informatics

Much of the past successful work in Climate Informatics has utilized Gaussian process regression (Camps-Valls et al., 2016), known in geostatistics as *kriging*, or tensor regression (Bahadori et al., 2014). More recently, He et al. (2019)

[1]There is evidence to suggest that statistical models trained on physical simulations learn an accurate representation of the interactions of Earth's physical systems (Barnston & Livezey, 1987; DelSole & Banerjee, 2017). On the other hand, generating "bad" samples has been shown to produce better results in some settings (Dai et al., 2017).

used a weighted LASSO technique to regress land temperature on sea surface temperature for promising results. Their approach utilized external relational knowledge; the ℓ_1 coefficient regularization penalties were set proportional to the distance between the particular sea and land locations. This demonstrates the potential for improvement to existing techniques when leveraging idiosyncrasies specific to the task at hand.

Clearly, there remains a lot of room for exploration in this field. State of the art results tend to be achieved due to a new collection of data or an ensemble technique involving multiple regression models (Hwang et al., 2018). Much work is still being done towards simply defining an appropriate metric for evaluating model quality (Council, 2010; National Academies of Sciences & Medicine, 2016). Because of this dearth of existing work, we believe a lot can be accomplished by leveraging recent machine learning results to build more accurate statistical models of Earth's weather system.

2.2. Normalizing Flows

The idea of normalizing flows was first suggested by G. Tabak & Vanden-Eijnden (2010), though the concept of iterative Gaussianization for density estimtation came much earlier (Chen & Gopinath, 2001). The recent increase in interest began when Rezende & Mohamed (2016) demonstrated that normalizing flows are an effective tool for achieving complex approximations to the true posterior distribution in variational inference (Kingma & Welling, 2013).

Consider some smooth, invertible function $f : \mathbb{R}^d \to \mathbb{R}^d$ and its inverse. Suppose we draw some random variable from a known base distribution $\boldsymbol{z} \sim q(\boldsymbol{z})$ and apply f to get some new random variable $\boldsymbol{z}' = f(\boldsymbol{z})$. Then there is a simple formula for the density of the transformed variable:

$$q(\boldsymbol{z}') = q(\boldsymbol{z}) \left| \det \frac{\partial f^{-1}}{\partial \boldsymbol{z}'} \right| = q(\boldsymbol{z}) \left| \det \frac{\partial f}{\partial \boldsymbol{z}} \right|^{-1}$$

where the second equality is known as the inverse function theorem. The determinant of the Jacobian of f at the point $\boldsymbol{z}$ represents the local expansion or contraction of density, and the end result is a valid probability distribution, hence the term "normalizing".

In order to ensure that the function is invertible and the determinant of its Jacobian can be efficiently computed, various constraints must be placed on its architecture. This somewhat limits its flexibility, but we can get around this by chaining together several such functions. The flow then becomes:

$$\boldsymbol{z}_K = f_K(f_{K-1}(\dots f_1(\boldsymbol{z}_0)\dots))$$

and log-likelihood evaluation for training can be similarly calculated:

$$\log q_K(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}} \right|$$

Dinh et al. (2014) previously suggested a similar formulation, but restricted their transformative function to ensure a Jacobian determinant of 1, simplifying the above expression. Rezende & Mohamed (2016) present several classes of functions with Jacobians and inverses that are particularly easy to compute. Subsequent work has developed functions that are more expressive, while leveraging parallelizability to allow for efficient inference and sampling.

One such approach that has proven particularly popular is using a bijective autoregressive neural network. Since each dimension only depends on the dimensions before it in the chosen ordering, the Jacobian of this transformation is triangular. As a result, the determinant is simply the product of the diagonal terms and is easy to calculate.

A Masked Autoregressive Flow (Papamakarios et al., 2017) learns a network that maps the observed variables to a distribution over the latent variables. This allows for efficient, parallelizable inference; however, sampling from this model is slow as each dimension of the output must be sampled before the next one can start.

Alternatively, Kingma et al. (2016) suggest an Inverse Autoregressive Flow. This is similar to a MAF, but the neural network is applied to the base distribution rather than the transformed distribution. As a result, sampling can be done easily, but inference requires serial recovery of the base dimensions. Both MAF and IAF make use of a Masked Autoencoder for Distribution Estimation (Germain et al., 2015), which allows for parallel autoregressive neural network evaluation (rather than learning and evaluating a different network for each conditional distribution).

More recent work has utilized Neural Ordinary Differential Equations (Chen et al., 2018), in which a neural network parameterizes the derivative of the hidden state to represent a continuous-depth model—the output is then the solution to an ODE. Extending this further on normalizing flows, Grathwohl et al. (2018) introduce FFJORD, an efficient algorithm for continuous, reversible flow of density.

3. Methods

3.1. Our Approach

In this report we work with z-scored monthly mean sea surface temperature (SST) observations. The dataset does not actually contain the true observations—rather, it consists of the simulated observations from thirteen different dynami-

cal models. The dataset further includes the true observed average land surface temperatures (LST). The most immediate idea would be to regress LST on SST, such as in He et al. (2019); however, we feel the field of existing approaches of this flavor is somewhat saturated—not to say that progress cannot be made, but the recent literature heavily implies such progress can expect to be incremental at best. Because of this, we hoped to find a new angle from which to attack the problem.

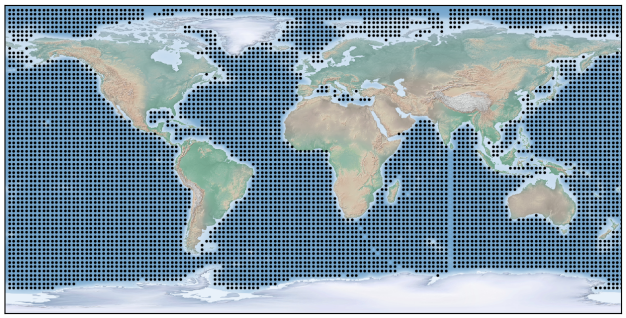

Figure 1. A depiction of the locations for which we have simulated mean monthly observations of SST from 13 models over the period 1849-2005. Notice that some locations that might be expected to have observations are conspicuously absent.

Density Estimation One novel direction we might consider is to throw away the regressands and instead put effort into modeling the distribution of observations. This is quite different from previous approaches, but it offers several distinct and complementary benefits which can further improve the predictive ability of existing models. To begin with, modeling the density of SST is an excellent tool for anomaly detection. If we retain temporal information of our observations, we can segment our dataset into two or more contiguous chunks of observations. We can then learn the distributions of each of these chunks to assess if there is indeed a distributional shift. Knowledge of the details of this shift would potentially be more helpful than the integration technique of standard SARIMA models[2].

The primary appeal of modeling the distribution, however, is the flexibility it gives us in dealing with the difficult realities of climate data collection in the real world. For medium- to long-term weather prediction, our hope is to achieve long-range forecasts that ignore the short-term chaotic nature of weather, while still picking up slower-moving, potentially aperiodic patterns. One way of doing this is to simulate the weather with different initial conditions, to get a sense of the long-range, invariant effects that repeatedly appear.

Unfortunately, we are confined to only a single reality, and therefore we get to see only one true time-series of observations; coming up with different but physically plausible initial conditions can be challenging. A common remedy is to simulate Earth's climate with different dynamical models (as in He et al. (2019)), but these systems are massively complicated and slow and costly to execute. In 2016, more than seven percent of the compute power of the world's 500 most powerful computers was completely dedicated to numerical weather simulation and prediction (Feldman, 2017). If we want to learn these long-range properties of Earth's climate, it would be more efficient to learn a graphical representation of the underlying distribution of SSTs, making sampling new feasible points much easier.

On top of this, missing data is a very common problem in Climate Informatics (Council, 2010; National Academies of Sciences & Medicine, 2016). As further evidence, note that even our dataset is missing observations for a set of locations which should be present, including one entire longitudinal line through Asia (see Figure 1). This exemplifies a typical issue with such a nascent field, which is the lack of standardized data which can serve as a baseline against which all approaches can be equally compared. By learning the distribution of observations, we can sample from the appropriate conditional distribution to fill in these missing values.

These two goals, sampling from the full and conditional distributions of global SST, are the primary focus of this work. Implementing a normalizing flow model for such a dataset is relatively simple, though it remains to show that such a model can better approximate the distribution than existing baselines, and that it can do so without overfitting. Conditional sampling, on the other hand, requires a bit more care.

3.2. Markov Chain Monte Carlo

Ideally, given a normalizing flow model which defines some distribution $p(x)$ for $x \in \mathbb{R}^d$, we might hope to be able to directly find a closed form for the conditional distribution. Consider partitioning an observation x into two disjoint sets of dimensions, x_1 and x_2. With a bit of abuse of notation, we wish to evaluate $p_x(x_2 \mid x_1)$. By Bayes' Law, we have

$$p(x_2 \mid x_1) = \frac{p(x_1, x_2)}{p_x(x_1)} = \frac{p(x)}{\int p(x_1, x_2) dx_2}$$

The numerator is given; it is precisely the value our model is constructed to be able to easily evaluate. The denominator, however, is more troublesome. Because of the arbitrarily complex ways in which a normalizing flow mixes each of the dimensions together at each transformation, there is no

[2]The "I" in SARIMA stands for "integration", which refers to the choice to subtract from each time-series observation the previous observation's value. This helps to remove some of the non-stationarity of the data by modeling differences rather than absolute values, similar to a residual neural network.

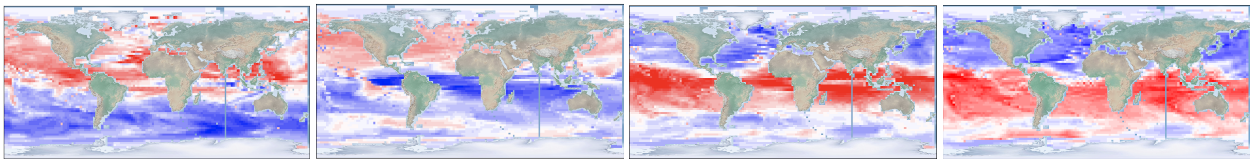

Figure 2. Four observations from the dataset. Each observation consists of sea temperatures at a grid of locations evenly spaced throughout the earth's oceans. We visualize each observation by plotting the temperature of each grid point as a color — red is hotter, blue is colder. Observe that the data exhibits interesting structure: dimensions with the same latitude are likely to be positively correlated.

simple closed form for this integral. It is clear that an analytical result of this expression in the context of normalizing flows is intractable. As a result, direct sampling from this conditional distribution is infeasible.

Instead, we consider sampling via Markov Chain Monte Carlo (MCMC). This family of techniques allows us to sample from a distribution from which sampling might normally be difficult, provided that we are able to evaluate the probability of the desired distribution up to a constant. Note that, for a fixed x_1,

$$p(x_2 \mid x_1) \propto p(x_1, x_2) = p(x)$$

Thus, we are able to evaluate the likelihood up to a constant, and we can therefore consider MCMC for sampling from the conditional.

The simplest choice would be to simply implement Metropolis-Hastings (MH), but in such high dimensions this is unlikely to work. If the proposals are too close to the current point we will never converge to the true distribution; likewise if the proposals are too far they will be rejected far too frequently to ever get anywhere. Additionally, we can reasonably expect the distribution of SSTs to be sharply multimodal, inducing a probability distribution which MH typically struggles to fully explore. Instead, we implement a modified MCMC algorithm known as Hamiltonian Monte Carlo (Neal, 2012; Betancourt, 2017).

3.3. Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) dramatically improves the efficiency and reduces the autocorrelation of the "blind" sampling process of MH by relating the sampling process to a dynamical system; HMC lifts the distribution into a higher dimensional phase-space over which a set of ODEs has been defined whose attractor sets correspond to areas of higher likelihood.

Concretely, HMC starts by mapping a point $\mathbf{x}$ in the d-dimensional sampling space to an element in phase space by concatenating it with d-dimensional momentum vector, m. A joint distribution over the phase space, known as the *canonical distribution* π, is then defined so that $\int_m \pi(x, m)dm = p(x)$, which in turn defines the Hamiltonian H—a useful quantity borrowed from classical mechanics.

Analogous to physical systems, $H(x, m)$ encodes the total energy of the system as the sum of the kinetic and potential energies. We expect the total energy to be constant over time, governed by the following Hamiltonian dynamics:

$$\frac{dx}{dt} = \frac{\partial H}{\partial m} = \frac{\partial K}{\partial m}$$
$$\frac{dm}{dt} = -\frac{\partial H}{\partial m} = -\frac{\partial K}{\partial x} - \frac{\partial V}{\partial x}$$

By design, the attractor sets of this vector field correspond to areas of high likelihood under p; thus, given an initial starting point, following this Hamiltonian leads us to desirable samples. While standard ODE approximation techniques can handle this task, the second-order method of leapfrog integration has proven to be particularly well suited here given the form of this system, providing computational and stability advantages.

Evaluating the Gradient In addition to being able to evaluate the likelihood up to a constant factor, HMC requires that we have access to the gradient of the log-likelihood, $\nabla_{x_2} \log p(x_2 \mid x_1)$. For a typical implementation, we would ideally have either a closed form of this derivative, or we would take advantage of the autodiff feature to evaluate the derivative automatically. In this instance, we neither have a closed-form log-likelihood, nor does our system ever actually evaluate $p(x_2 \mid x_1)$ directly. However, a simple observation means we can still make use of this powerful algorithm:

$$\begin{aligned}\nabla_{x_2} \log p(x_2 \mid x_1) &= \nabla_{x_2}[\log p(x) - \log p(x_1)] \\ &= \nabla_{x_2} \log p(x)\end{aligned}$$

When we evaluate $\log p(x)$ we get $\nabla \log p(x)$ for free, and we can simply drop the dimensions corresponding to the values of x_1 to exactly determine the desired partial derivatives.

4. Experiments

4.1. Dataset

The dataset consists of 24,612 observations; each observation is a vector of dimension 5881, consisting of the sea surface temperature recorded at 5881 distinct sea locations at $2.5^\circ \times 2.5^\circ$ gridded points on the Earth. These grid locations are depicted in Figure 1.

We randomly partitioned the dataset into a training set of size 21,612, a validation set of size 1,000, and a testing set of size 2,000. We then z-scored each of these sets according to the sample mean and variance of the test set.

Figure 2 shows four observations from the dataset visualized as heatmaps—red for positive temperature deviations, blue for negative (i.e., hotter and colder). Note the structured nature of the data: the sea temperatures at grid locations at the same latitude are highly positively correlated. Additionally, the sea temperatures tend to be consistent within each of the the northern and southern hemispheres.

4.2. Multivariate Gaussian Baseline

Our baseline density model was a multivariate Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where the mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ are estimated from the training data.

The unregularized MLE estimator takes $\boldsymbol{\mu}$ as the sample mean $\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$ and $\boldsymbol{\Sigma}$ as the sample covariance matrix $\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T$. However, since there are 5881 dimensions and only 21,612 training points, we suspected that overfitting might be an issue. Therefore we further experimented with three regularization schemes:

1. Use a diagonal Gaussian model instead, i.e. $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}))$.
2. Use a full-covariance Gaussian, but "shrink" the covariance matrix (Schafer & Strimmer, 2005) by taking $\boldsymbol{\Sigma}$ to be a convex combination of the empirical covariance matrix (weight α) and the identity matrix $\boldsymbol{I}$ (weight $1 - \alpha$). Tune α on the validation dataset.
3. Same as above, but with α computed from the training set using the Ledoit-Wolf formula (Ledoit & Wolf, 2012).

The train and test (average) log-likelihoods of these four methods are shown in Table 1. Observe that the full covariance MLE estimator with no regularization slightly overfits. In contrast, the diagonal Gaussian MLE estimator substantially underfits. The best two methods are the two shrinkage estimators for the full-covariance model. Of these, the Ledoit-Wolf method attains a slightly higher test accuracy. For this reason, we use the Ledoit-Wolf estimator as the Gaussian baseline in the remainder of this paper.

Figure 3 shows four samples from the Gaussian baseline. The Gaussian baseline broadly picks up on the pattern that temperatures at similar latitudes are correlated, but it doesn't quite seem to understand that neighboring latitudes should also have low variance. There are several locations with rapidly-changing temperatures across a short distance in the North-South direction, which should be very unlikely based on the original observations.

4.3. Real NVP

LAYERS	TRAIN	TEST
1	-824.07	-1050.58
2	6019.32	5615.01
3	7522.79	7138.43
4	8328.33	7478.42
5	8487.2	7964.5
7	9719.95	**9021.75**
10	9627.52	8260.2

Table 2. Train and test log-likelihoods of Real NVP with 400 hidden units in the neural network of each affine coupling layer, as the number of affine coupling layers is varied. We suspect the decrease from 7 to 10 layers was due to optimization challenges.

Model description Our principal model was Real NVP (Dinh et al., 2016). Real NVP is a reversible generative model which consists of a series of "affine coupling layers," each an invertible transformation whose Jacobian determinant is efficiently computed. Each affine coupling layer $f_k : \mathbb{R}^d \to \mathbb{R}^d$ involves a partitioning of the d dimensions into two index sets $S_1,\ S_2 \subset [d]$ of sizes $|S_1| = d_1$ and $|S_2| = d_2$. The forward equations for an affine coupling layer with input $\mathbf{x} \in \mathbb{R}^d$ and output $\mathbf{y} \in \mathbb{R}^d$ are as follows:

$$\mathbf{y}_{S_1} = \mathbf{x}_{S_1}$$
$$\mathbf{y}_{S_2} = \mathbf{x}_{S_2} \circ \mathbf{a}_k(\mathbf{x}_{S_1}) + \mathbf{b}_k(\mathbf{x}_{S_1})$$

where $\circ$ is element-wise multiplication. Here, $\mathbf{a}_k : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ and $\mathbf{b}_k : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ are functions which determine

SPLIT	SHRINKAGE	LEDOIT-WOLF	FULL COVARIANCE	DIAGONAL COVARIANCE
TRAIN	8921.74	8760.49	9123.27	-8344.77
TEST	7779.99	**7819.00**	7318.10	-8362.4

Table 1. Train and test log-likelihoods of four baseline models on the sea surface temperature dataset. The strongest baseline is LEDOIT-WOLF. Notice that FULL COVARIANCE overfits and DIAGONAL COVARIANCE severely underfits.

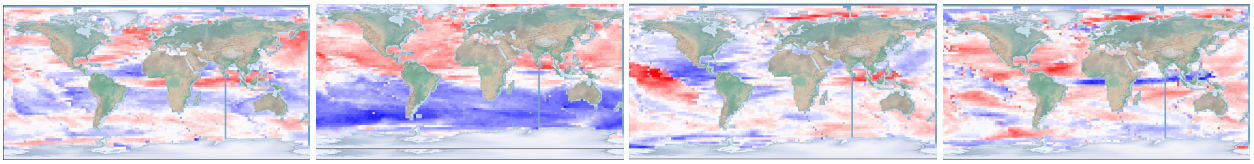

Figure 3. Four samples from a multivariate Gaussian density model. Compare these samples to the real data in Figure 2.

how to shift and scale the dimensions in S_2 using the dimensions in S_1 (they are typically implemented as feed-forward neural networks). Observe that this transformation is intervible! Given some $\mathbf{y}$, we can recover the original $\mathbf{x}$ as follows:

$$\mathbf{x}_{S_1} = \mathbf{y}_{S_1}$$
$$\mathbf{x}_{S_2} = (\mathbf{y}_{S_2} - \mathbf{b}_k(\mathbf{y}_{S_1}))/\mathbf{a}_k(\mathbf{y}_{S_1})$$

The Jacobian determinant of this transformation is also easy to compute. The Jacobian is triangular, and therefore its determinant is simply the product along its diagonal. More concretely, we have

$$\left| det \frac{\partial f}{\partial z} \right| = \exp\left(\sum_k a_k(\mathbf{x}_{S_1}) \right)$$

A single affine coupling layer is not a very expressive transformation, but by composing together a series of affine coupling layers sequentially, we can transform the base density to an arbitrarily complex target density. To ensure that all dimensions are able to affect each other, the partitioning $[d] = S_1 \cup S_2$ needs to change from layer to layer. We followed Dinh et al. (2016) by taking S_1 and S_2 to be the even and odd dimensions, alternating from layer to layer.

HIDDEN UNITS	TRAIN	TEST
100	6645.26	6399.80
200	7844.4	7469.83
400	8487.2	7964.5
800	9606.95	**8665.56**

Table 3. Train and test log-likelihoods of Real NVP with 5 affine coupling layers as the number of hidden units in the neural networks of each affine coupling layer is varied.

Hyperparameters We implemented each $\mathbf{a}_k$ and $\mathbf{b}_k$ as a single neural network with two hidden layers with input dimension d_1 and output dimension $2d_2$. We chose the number of hidden units in each of these layers, as well as the overall number of affine coupling layers, according to performance on a held-out test set.

Table 2 and Table 3 show the performance of the model as these two hyperparameters are varied. We obtained the best test log-likelihood using 7 affine coupling layers with 400 hidden units in each layer of the shift and scale neural networks.

Performance Table 4 shows that the test-set log likelihood of our Real NVP model vastly exceeded the test-set log-likelihood of our Gaussian baseline.

MODEL	TRAIN	TEST
GAUSSIAN	8760.49	7819.00
REAL NVP	9719.95	**9021.75**

Table 4. Train and test log-likelihoods of our Gaussian baseline, and a Real NVP model with 7 affine coupling layers with 400 hidden units each.

Figure 4 shows unconditional samples from our Real NVP density model. We can see that this model estimates the distribution of sea temperatures significantly better—the temperatures at nearby longitudes are more coherent (mirroring the real data) than unconditional samples from the Gaussian baseline.

4.4. Further Analysis of Learned Model

In an attempt to understand and verify the distribution learned by our model, we examine the trained model's likelihood as we perturb the data to become dissimilar to true weather patterns. This process allows us to determine whether the model has learned a distribution which aligns with our external knowledge, at a minimum by placing higher likelihood on observations which we recognize as realistic weather. We go about this analysis by recording the model's response to 1-dimensional perturbations of the test set data. The way in which we perturb the data is carefully chosen to preserve some critical properties of the original data while violating others.

Let D be the test-set prior to z-scoring. For various values of α, we evaluate the model over the following α-dependent datasets:

$$S_{\text{affine}}(\alpha) = \text{Z-score}(\{(1-\alpha)x + \text{mean}(x) | x \in D\})$$

$$S_{\text{const}}(\alpha) = \text{Z-score}(\{x + \alpha | x \in D\})$$

where $\text{mean}(x) = \frac{1}{dim(x)} \sum_{i=1}^{dim(x)} \{x\}_i$ and addition is done independently to each dimension. Thus, S_{affine} is a combination of a true observation and an unrealistic world where all locations have the same temperature, while S_{const} is the result of adding a constant value to all dimensions (which becomes more unrealistic as $|\alpha|$ grows).

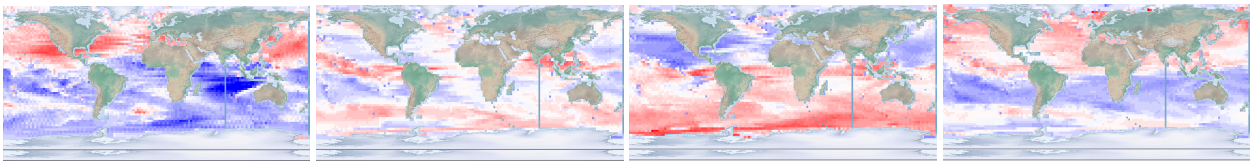

Figure 4. Four samples from a Real NVP density model. Compare these samples to the real data in Figure 2 and to samples from the Gaussian baseline in Figure 3.

While these transformations lose certain properties of real weather, they are designed to retain others: $S_{\text{affine}}(\alpha)$ preserves the per-instance mean temperature while changing inter-location temperature differences, and $S_{\text{const}}(\alpha)$ preserves the difference in adjacent temperatures while modifying the per-instance mean. Notice that by definition, $S_{\text{affine}}(0) = S_{\text{const}}(0) = D$, which is appropriate given α is intended to indicate the degree of displacement from the original test instances.

Figure 6 displays the model's log-likelihoods as we vary α. Note that the log-likelihood is always strictly maximized around $\alpha = 0$, suggesting Real NVP is not simply relying on mean global temperature or local temperature consistency alone to evaluate likelihood. Comparing the two sub-plots, the model is far more sensitive to changes in global mean temperature than changes in local temperature differentials, while being asymmetrically effected by the direction of change.

Summarizing, we see that our model has recovered multiple facets of weather's defining properties, and it assigns higher likelihood to real data than semi-realistic fake data. This provides further evidence that the model has learnt a distribution that meaningfully relates to Earth's climate.

4.5. Imputation

Density estimation is a widely studied task at which deep generative models have been shown to excel. But the classical framework of probabilistic graphical models studies

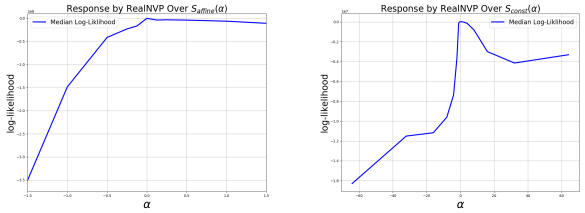

Figure 6. Response plots for RealNVP over expert-chosen subspaces. We report the median to be robust to outliers which are present in the results.

a multitude of other tasks as well. In particular, classical PGMs generally support efficient operations like conditioning. Little previous work has studied the problem of computing conditional probability distributions with reversible deep generative models. We chose to evaluate the feasibility of this idea by applying DGMs to *imputation*.

The task of imputation is this: given a fitted density model p and a data point $\mathbf{x} \in \mathbb{R}^d$, conceal a subset of the dimensions $S_2 \subset [d]$ and try to predict them from the remaining dimensions $\mathbf{x}_{S_1}$. How to make such a prediction $\hat{\mathbf{x}}_{S_2}$, given a fitted density model p? One way is to compute the conditional expectation of the dimensions in S_2, given the dimensions in S_1:

$$\hat{\mathbf{x}}_{S_2} := \mathbb{E}_{\mathbf{X}_{S_2} \sim p(\mathbf{X}_{S_2} | \mathbf{X}_{S_1} = \mathbf{x}_{S_1})}[\mathbf{X}_{S_2}]$$

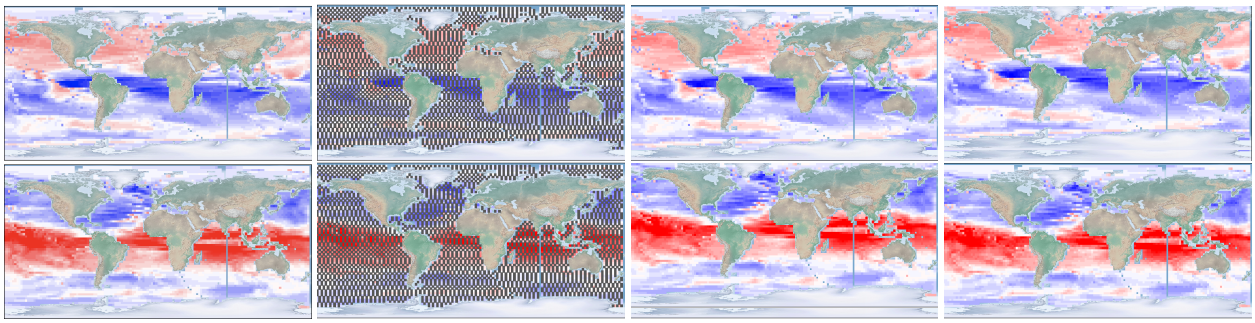

Figure 5. Imputations of two data points. From left to right: the original example, the example with every other dimension hidden (drawn in black), the imputation by the Gaussian baseline, the imputation by the Real NVP with HMC.

In this report, we measure the quality of the imputation $\hat{\mathbf{x}}_{S_2}$ by computing the difference between $\hat{\mathbf{x}}_{S_2}$ and $\mathbf{x}_{S_2}$ in squared ℓ_2 norm: $\|\hat{\mathbf{x}}_{S_2} - \mathbf{x}_{S_2}\|_2^2$.

We chose to alternate in earth grid space between concealed and visible dimensions. This setup is visualized in the second column of Figure 5. The blacked-out dimensions are the ones which we conceal. It is important to note that the model treats each dimension separately and is not given the spatial structure, so the task is indeed challenging.

Gaussian baseline If p is a multivariate Gaussian distribution $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, there is a well-known closed-form solution for the conditional expectation of the dimensions S_2 given the dimensions S_1:

$$\hat{\mathbf{x}}_{S_2} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_{S_1} - \boldsymbol{\mu}_1) \tag{1}$$

The third column of Figure 5 shows two examples of imputation using the Gaussian baseline.

Reversible generative models For a reversible generative model, there is no way to exactly compute the conditional probability distribution of $\mathbf{x}_{S_2}$ given $\mathbf{x}_{S_1}$, nor can we precisely evaluate the expectation of that distribution. Therefore, as discussed above, we approximated the conditional expectation using Hamiltonian Monte Carlo.

Figure 8 illustrates this process for a single data point $\mathbf{x}$. Each of the four subplots in Figure 8 is a single dimension in S_2. The green line is the true, concealed value; the orange line is the Gaussian density that is predicted using the Gaussian baseline and Equation (1); and the blue lines are samples from the conditional distribution of $\mathbf{X}_{S_2}|\mathbf{X}_{S_1} = \mathbf{x}_{S_1}$ under the normalizing flow, sampled using Hamiltonian Monte Carlo. Observe that both the Gaussian baseline and normalizing flow / HMC combination do a decent job at predicting the concealed dimension.

HMC vs MH We found that HMC consistently outperformed the random-walk MH baseline for normalizing flow conditional sampling. Figure 7 shows the log-probability during burn-in of HMC compared to MH across three different settings of the proposal distribution variance. Since each HMC "step" involves a number of gradient evaluations, it would be unfair to compare MH to HMC on a step-by-step basis. Therefore, we instead compare the progress of the two algorithms in terms of the number of function or gradient evaluations. Observe that the HMC chain attains a much higher log-probability and it does so at a faster rate. This is because HMC leverages access to gradient information, whereas random walk MH does not.

Tuning HMC Hamiltonian Monte Carlo needs to be tuned extensively. Theoretically, the step size should be as small as possible (to minimize effects of time discretization) and the number of steps simulated should be very large (to allow the dynamics to fully explore the distribution). Realistically, this would be computationally infeasible, and we instead tune a careful balance.

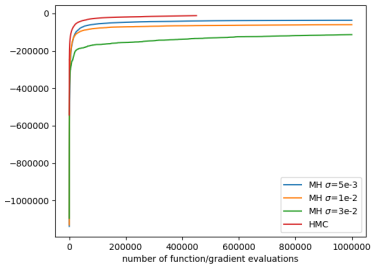

Figure 7. Comparing HMC to random-walk Metropolis Hastings.

Figure 9 shows, for a fixed number of time steps and for four separate settings of the step size, a plot of the log-probability during iterations 250-500 of the burn-in period. Observe that the step size plays a role similar to that of the proposal variance in MH: when the step size is too small (top left), the chain converges to its stationary distribution too slowly — at the end of the burn-in period, the chain is still yet to converge. On the other hand, when the step size is too large (bottom right), no proposals are accepted, and the chain does not move anywhere.

Warm-starting HMC While HMC mixed much faster than MH, it was still far too slow when we initialized the chain randomly (i.e. drawn from a Gaussian distribution). Therefore, we also experimented with warm-starting HMC with a sample from the conditional Gaussian model. This warm-start approach resulted in significantly faster convergence to the stationary distribution.

The top and bottom rows of Figure 10 plot the log-probability of the chain, during the burn-in period, when the chain is warm-started and cold-started, respectively (note the differing scales for the number of iterations). Observe that the warm-start model after just 250 steps attains a log-probability of about 9050, whereas the cold-start model after 7,500 steps only attains a log-probability of -18500.

5. Conclusion

In this paper we have conducted the first experiments on the feasibility of reversible deep generative models applied to the challenging field of Climate Informatics. Despite the small sample size, we found that Real NVP, a relatively simple normalizing flow architecture, significantly outperforms existing baselines for density estimation and, with a few computational tricks, performs comparably in the

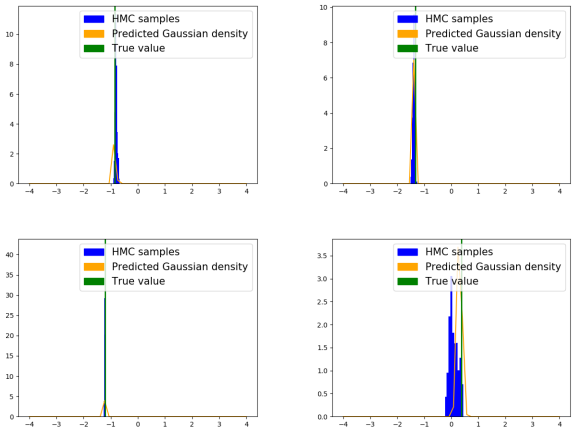

Figure 8. Computing conditional expectations using HMC.

task of imputation. We further provided several compelling arguments for the uses and benefits of effective density estimation: being able to sample from the full or conditional distribution of Earth's SSTs can be thoughtfully leveraged for significantly improved medium- to long-term weather forecasting.

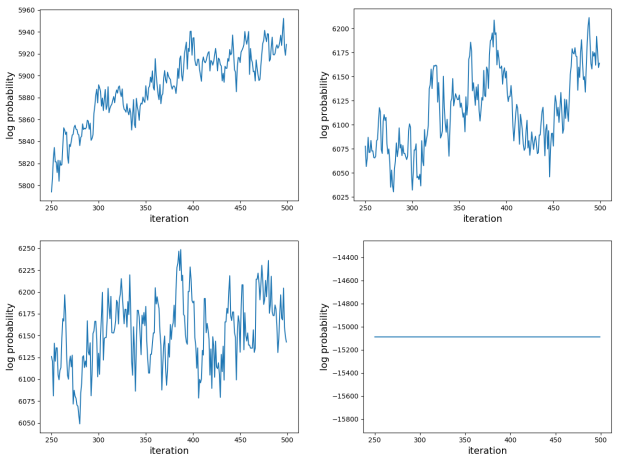

Figure 9. Log-probability of the Markov chain during burn in (iter 250 - 500) as the HMC step size is varied as 1e-4 (top left), 5e-4 (top right), 1e-3 (bottom left), 5e-3 (bottom right).

References

Bahadori, M. T., Yu, Q. R., and Liu, Y. Fast multivariate spatio-temporal analysis via low rank tensor learning. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3491–3499. Curran Associates, Inc., 2014.

Barnston, A. G. and Livezey, R. E. Classification, seasonality and persistence of low-frequency atmospheric circulation patterns. *Monthly Weather Review*, 115(6):1083–1126, 1987. doi: 10.1175/1520-0493(1987)115⟨1083:CSAPOL⟩2.0.CO;2. URL https://doi.org/10.1175/1520-0493(1987)115<1083:CSAPOL>2.0.CO;2.

Betancourt, M. A conceptual introduction to hamiltonian monte carlo, 2017.

Camps-Valls, G., Verrelst, J., Muoz, J., Laparra, V., Mateo, F., and Gomez-Dans, J. A survey on gaussian processes for earth-observation data analysis: A comprehensive investigation. *IEEE Geoscience and Remote Sensing Magazine*, 4:58–78, 06 2016. doi: 10.1109/MGRS.2015.2510084.

Chen, S. S. and Gopinath, R. A. Gaussianization. In Leen, T. K., Dietterich, T. G., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems 13*, pp. 423–429.

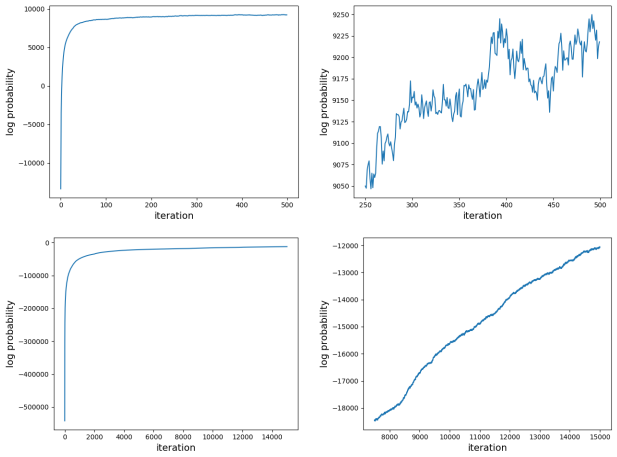

Figure 10. The importance of warm-starting HMC. **Top left:** warm start, burn-in steps 1-500. **Top right:** warm start, burnin-steps 250-500. **Bottom left**: cold start, burnin-in steps 1-15,000. **Bottom right**: cold start, burnin-in steps 7,500-15,000.

MIT Press, 2001. URL http://papers.nips.cc/paper/1856-gaussianization.pdf.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018.

Council, N. R. *Assessment of Intraseasonal to Interannual Climate Prediction and Predictability*. The National Academies Press, Washington, DC, 2010. ISBN 978-0-309-15183-2. doi: 10.17226/12878.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. R. Good semi-supervised learning that requires a bad gan. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6510–6520. Curran Associates, Inc., 2017.

DelSole, T. and Banerjee, A. Statistical seasonal prediction based on regularized regression. *Journal of Climate*, 30(4):1345–1361, 2017. doi: 10.1175/JCLI-D-16-0249.1. URL https://doi.org/10.1175/JCLI-D-16-0249.1.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp, 2016.

Feldman, M. Top500 meanderings: Supercomputers for weather forecasting have come a long way, 2017. URL https://www.top500.org/news/top500-meanderings-supercomputers-for-weather-forecasting-have-come-a-long-way/.

G. Tabak, E. and Vanden-Eijnden, E. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences - COMMUN MATH SCI*, 8, 03 2010. doi: 10.4310/CMS.2010.v8.n1.a11.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, Lille, France, 2015.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models, 2018.

He, S., Li, X., Sivakumar, V., and Banerjee, A. Interpretable predictive modeling for climate variables with weighted lasso. 2019.

Hwang, J., Orenstein, P., Cohen, J., Pfeiffer, K., and Mackey, L. Improving subseasonal forecasting in the western u.s. with machine learning, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29*, pp. 4743–4751, 2016.

Ledoit, O. and Wolf, M. Nonlinear shrinkage estimation of large-dimensional covariance matrices. *The Annals of Statistics*, 2012.

National Academies of Sciences, E. and Medicine. *Next Generation Earth System Prediction: Strategies for Subseasonal to Seasonal Forecasts*. The National Academies Press, Washington, DC, 2016. ISBN 978-0-309-38880-1. doi: 10.17226/21873.

Neal, R. M. Mcmc using hamiltonian dynamics. 2012.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation, 2017.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, Lille, France, 2016.

Schafer, J. and Strimmer, K. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 2005.