Transformation Autoregressive Networks

Junier B Oliva [1 2] **Avinava Dubey** [2] **Manzil Zaheer** [2]
Barnabás Póczos [2] **Ruslan Salakhutdinov** [2] **Eric P Xing** [2] **Jeff Schneider** [2]

Abstract

The fundamental task of general density estimation $p(x)$ has been of keen interest to machine learning. In this work, we attempt to systematically characterize methods for density estimation. Broadly speaking, most of the existing methods can be categorized into either using: *a*) autoregressive models to estimate the conditional factors of the chain rule, $p(x_i \,|\, x_{i-1}, \ldots)$; or *b*) non-linear transformations of variables of a simple base distribution. Based on the study of the characteristics of these categories, we propose multiple novel methods for each category. For example we propose RNN based transformations to model non-Markovian dependencies. Further, through a comprehensive study over both real world and synthetic data, we show that jointly leveraging transformations of variables and autoregressive conditional models, results in a considerable improvement in performance. We illustrate the use of our models in outlier detection and image modeling. Finally we introduce a novel data driven framework for learning a family of distributions.

1. Introduction

Density estimation is at the core of a multitude of machine learning applications. However, this fundamental task is difficult in the general setting due to issues like the curse of dimensionality. Furthermore, for general data, unlike spatial/temporal data, we do not have known correlations a priori among covariates that may be exploited. For example, image data has known correlations among neighboring pixels that may be hard-coded into a model through convolutions, whereas one must find such correlations in a data-driven fashion with general data.

[1]Computer Science Department, University of North Carolina, Chapel Hill, NC 27599 (Work completed while at CMU.) [2]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213. Correspondence to: Junier Oliva <joliva@cs.unc.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

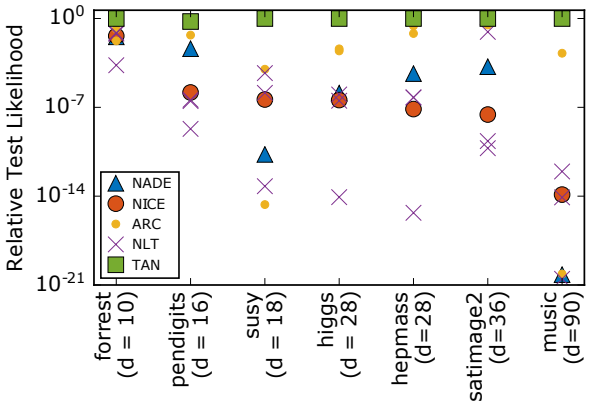

Figure 1. The proposed *TAN* models for density estimation, which jointly leverages non-linear transformation and autoregressive conditionals, shows considerable improvement over other methods across datasets of varying dimensions. The scatter plots shows that only utilizing autoregressive conditionals (ARC) without transformations (*e.g.* existing works like NADE (Uria et al., 2014) and other variants) or only using non-linear transformation (NLT) with simple restricted conditionals (*e.g.* existing works like NICE (Dinh et al., 2014) and other variants) is not sufficient for all datasets.

In order to model high dimensional data, the main challenge lies in constructing models that are flexible enough while having tractable learning algorithms. A variety of diverse solutions exploiting different aspects of the problems have been proposed in the literature. A large number of methods have considered auto-regressive models to estimate the conditional factors $p(x_i|x_{i-1}, \ldots, x_1)$, for $i \in \{1, \ldots, d\}$ in the chain rule (Larochelle & Murray, 2011; Uria et al., 2013; 2016; Germain et al., 2015; Gregor et al., 2014). While some methods directly model the conditionals $p(x_i|x_{i-1}, \ldots)$ using sophisticated semiparametric density estimates, other methods apply sophisticated transformations of variables $x \mapsto z$ and take the conditionals over z to be a restricted, often independent base distribution $p(z_i|z_{i-1}, \ldots) \approx f(z_i)$ (Dinh et al., 2014; 2016). Further related works are discussed in Sec. 3. However, looking across a diverse set of dataset, as in Fig. 1, neither of these approaches have the flexibility required to accurately model real world data.

In this paper we take a step back and start from the basics. If we only model the conditionals, the conditional factors $p(x_i|x_{i-1}, \ldots)$, may become increasingly complicated as i increases to d. On the other hand if we use a complex

transformation with restricted conditionals then the transformation has to ensure that the transformed variables are independent. This requirement of independence on the transformed variables can be very restrictive. Now note that the transformed space is homeomorphic to the original space and a simple relationship between the density of the two spaces exists through the Jacobian. Thus, we can employ conditional modeling on the transformed variables to alleviate the independence requirement, while being able to recover density in the original space in a straightforward fashion. In other words, we propose *transformation autoregressive networks* (TANs) which composes the complex transformations and autoregressive modeling of the conditionals. The composition not only increases the flexibility of the model but also reduces the expressibility power needed from each of the individual components. This leads to an improved performance as can be seen from Fig. 1.

In particular, first we propose two flexible autoregressive models for modeling conditional distributions: the linear autoregressive model (LAM), and the recurrent autoregressive model (RAM) (Sec. 2.1). Secondly, we introduce several novel transformations of variables: 1) an efficient method for learning a linear transformation on covariates; 2) an invertible RNN-based transformation that directly acts on covariates; 3) an additive RNN-base transformation (Sec. 2.2). Extensive experiments on both synthetic (Sec. 4.1) and real-world (Sec. 4.2) datasets show the power of TANs for capturing complex dependencies between the covariates. We run an ablation study to demonstrate contributions of various components in TAN Sec. 4.3, Moreover, we show that the learned model can be used for anomaly detection (Sec. 4.4) and learning a family of distributions (Sec. 4.5).

2. Transformation Autoregressive Networks

As mentioned above, TANs are composed of two modules: *a*) an autoregressive module for modeling conditional factors and *b*) transformations of variables. We first introduce our two proposed autoregressive models to estimate the conditional distribution of input covariates $x \in \mathbb{R}^d$. Later, we show how to use such models over transformation $z = q(x)$, while renormalizing to obtain density values for x.

2.1. Autoregressive Models

Autoregressive models decompose density estimation of a multivariate variable $x \in \mathbb{R}^d$ into multiple conditional tasks on a growing set of inputs through the chain rule:

$$p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p(x_i \,|\, x_{i-1}, \ldots, x_1). \tag{1}$$

That is, autoregressive models will estimate the d conditional distributions $p(x_i|x_{i-1}, \ldots)$. A class of autoregressive models can be defined by approximating conditional distributions through a mixture model, $\mathrm{MM}(\theta(x_{i-1}, \ldots, x_1))$, with parameters θ depending on $x_{i-1}, \ldots, x_1$:

$$p(x_i|x_{i-1}, \ldots, x_1) = p(x_i \,|\, \mathrm{MM}(\theta(x_{i-1}, \ldots, x_1)), \tag{2}$$

$$\theta(x_{i-1}, \ldots, x_1) = f\left(h_i\right) \tag{3}$$

$$h_i = g_i\left(x_{i-1}, \ldots, x_1\right), \tag{4}$$

where $f(\cdot)$ is a fully connected network that may use a element-wise non-linearity on inputs, and $g_i(\cdot)$ is some general mapping that computes a hidden state of features, $h_i \in \mathbb{R}^p$, which help in modeling the conditional distribution of $x_i \,|\, x_{i-1}, \ldots, x_1$. One can control the flexibility of the model through g_i. It is important to be powerful enough to model our covariates while still generalizing. In order to achieve this we propose two methods for modeling g_i.

Linear Autoregressive Model (LAM): This uses a straightforward linear map as g_i in (4):

$$g_i\left(x_{i-1}, \ldots, x_1\right) = W^{(i)} x_{<i} + b, \tag{5}$$

where $W^{(i)} \in \mathbb{R}^{p \times (i-1)}$, $b \in \mathbb{R}^p$, and $x_{<i} = (x_{i-1}, \ldots, x_1)^T$. Notwithstanding the simple form of (5), the resulting model is quite flexible as it may model consecutive conditional problems $p(x_i|x_{i-1}, \ldots, x_1)$ and $p(x_{i+1}|x_i, \ldots, x_1)$ very differently owing to different $W^{(i)}$s.

Recurrent Autoregressive Model (RAM): This features a recurrent relation between g_i's. As the set of covariates is progressively fed into g_i's, it is natural to consider a hidden state evolving according to an RNN recurrence relationship:

$$h_i = g\left(x_{i-1}, g(x_{i-2}, \ldots, x_1)\right) = g\left(x_{i-1}, h_{i-1}\right). \tag{6}$$

In this case $g(x, s)$ is a RNN function for updating one's state based on an input x and prior state s. In the case of gated-RNNs, the model will be able to scan through previously seen dimensions remembering and forgetting information as needed for conditional densities without making any strong Markovian assumptions.

Both LAM and RAM are flexible and able to adjust the hidden states, h_i in (4), to model the distinct conditional tasks $p(x_i|x_{i-1}, \ldots)$. There is a trade-off of added flexibility and transferred information between the two models. LAM treats the conditional tasks for $p(x_i|x_{i-1}, \ldots)$ and $p(x_{i+1}|x_i, \ldots)$ in a largely independent fashion. This makes for a very flexible model, however the parameter size is also large and there is no sharing of information among the conditional tasks. On the other hand, RAM provides a framework for transfer learning among the conditional tasks by allowing the hidden state h_i to evolve through the distinct conditional tasks. This leads to fewer parameters and more sharing of information in respective tasks, but also yields less flexibility since conditional estimates are tied, and may only change in a smooth fashion.

2.2. Transformations

Next we introduce the second module of TANs i.e. the transformations. When using an invertible transformation of variables $z = (q_1(x), \ldots, q_d(x)) \in \mathbb{R}^d$, one can establish a relationship between the pdf of x and z as:

$$p(x_1, \ldots, x_d) = \left|\det \frac{\mathrm{d}q}{\mathrm{d}x}\right| \prod_{i=1}^{d} p\left(z_i \,|\, z_{i-1}, \ldots, z_1\right), \quad (7)$$

where $\left|\det \frac{\mathrm{d}q}{\mathrm{d}x}\right|$ is the Jacobian of the transformation. For analytical and computational considerations, we require transformations to be invertible, efficient to compute and invert, and have a structured Jacobian matrix. In order to meet these criteria we propose the following transformations.

Linear Transformation: It is an affine map of the form:

$$z = Ax + b, \quad (8)$$

where we take A to be invertible. Note that even though this linear transformation is simple, it includes permutations, and may also perform a PCA-like transformation, capturing coarse and highly varied features of the data before moving to more fine grained details. In order to not incur a high cost for updates, we wish to compute the determinant of the Jacobian efficiently. Thus, we propose to directly work over an LU decomposition $A = LU$ where L is a lower triangular matrix with unit diagonals and U is a upper triangular matrix with arbitrary diagonals. As a function of L, U we have that $\det \frac{\mathrm{d}z}{\mathrm{d}x} = \prod_{i=1}^{d} U_{ii}$; hence we may efficiently optimize the parameters of the linear map. Furthermore, inverting our mapping is also efficient through solving two triangular matrix equations.

Recurrent Transformation: Recurrent neural networks are also a natural choice for variable transformations. Due to their dependence on only previously seen dimensions, RNN transformations have triangular Jacobians, leading to simple determinants. Furthermore, with an invertible output unit, their inversion is also straight-forward. We consider the following form to an RNN transformation:

$$z_i = r_\alpha\left(yx_i + w^T s_{i-1} + b\right), \; s_i = r\left(ux_i + v^T s_{i-1} + a\right), \quad (9)$$

where r_α is a leaky ReLU unit $r_\alpha(t) = \mathbb{I}\{t < 0\}\alpha t + \mathbb{I}\{t \geq 0\}t$, r is a standard ReLU unit, $s \in \mathbb{R}^\rho$ is the hidden state y, u, b, a are scalars, and $w, v \in \mathbb{R}^\rho$ are vectors. As compared to the linear transformation, the recurrent transformation is able to transform the input with different dynamics depending on its values. Inverting (9) is a matter of inverting outputs and updating the hidden state (where the initial state s_0 is known and constant):

$$\begin{aligned} x_i &= \frac{1}{y}\left(r_\alpha^{-1}\left(z_i^{(r)}\right) - w^T s_{i-1} - b\right), \\ s_i &= r\left(ux_i + v^T s_{i-1} + a\right). \end{aligned} \quad (10)$$

Furthermore, the determinant of the Jacobian for (9) is the product of diagonal terms:

$$\det \frac{\mathrm{d}z}{\mathrm{d}x} = y^d \prod_{i=1}^{d} r'_\alpha\left(yx_i + w^T s_{i-1} + b\right), \quad (11)$$

where $r'_\alpha(t) = \mathbb{I}\{t > 0\} + \alpha\mathbb{I}\{t < 0\}$.

Recurrent Shift Transformation: It is worth noting that the rescaling brought on by the recurrent transformation effectively incurs a penalty through the log of the determinant (11). However, one can still perform a transformation that depends on the values of covariates through a shift operation. In particular, we propose an additive shift based on a recurrent function on prior dimensions:

$$z_i = x_i + m(s_{i-1}), \quad s_i = g(x_i, s_{i-1}), \quad (12)$$

where g is recurrent function for updating states, and m is a fully connected network. Inversion proceeds as before:

$$x_i = z_i - m(s_{i-1}), \quad s_i = g(x_i, s_{i-1}). \quad (13)$$

The Jacobian is again lower triangular, however due to the additive nature of (12), we have a unit diagonal. Thus, $\det \frac{\mathrm{d}z}{\mathrm{d}x} = 1$. One interpretation of this transformation is that one can shift the value of x_k based on $x_{k-1}, x_{k-2}, \ldots$ for better conditional density estimation without any penalty coming from the determinant term in (7).

Composing Transformations: Lastly, we considering stacking (i.e. composing) several transformations $q = q^{(1)} \circ \ldots \circ q^{(T)}$ and renormalizing:

$$\begin{aligned} p(x_1, \ldots, x_d) = &\prod_{t=1}^{T} \left|\det \frac{\mathrm{d}q^{(t)}}{\mathrm{d}q^{(t-1)}}\right| \\ &\prod_{i=1}^{d} p\left(q_i(x) \,|\, q_{i-1}(x), \ldots, q_1(x)\right), \end{aligned} \quad (14)$$

where we take $q^{(0)}$ to be x. We note that composing several transformations together allows one to leverage the respective strengths of each transformation. Moreover, inserting a reversal mapping $(x_1, \ldots, x_d \mapsto x_d, \ldots, x_1)$ as one of the q_is yields bidirectional relationships.

2.3. Combined Approach

We combine the use of both transformations of variables and rich autoregressive models by: 1) writing the density of inputs, $p(x)$, as a normalized density of a transformation: $p(q(x))$ (14). Then we estimate the conditionals of $p(q(x))$ using an autoregressive model, *i.e.*, to learn our model we minimize the negative log likelihood:

$$\begin{aligned} &-\log p(x_1, \ldots, x_d) = \\ &-\sum_{t=1}^{T} \log\left|\det \frac{\mathrm{d}q^{(t)}}{\mathrm{d}q^{(t-1)}}\right| - \sum_{i=1}^{d} \log p\left(q_i(x) \,|\, h_i\right), \end{aligned} \quad (15)$$

which is obtained by substituting (2) into (14) with h_i as defined in (4).

3. Related Works

Nonparametric density estimation has been a well studied problem in statistics and machine learning (Wasserman, 2007). Unfortunately, nonparametric approaches like kernel density estimation suffer greatly from the curse of dimensionality and do not perform well when data does not have a small number of dimensions ($d \lesssim 3$). To alleviate this, several semiparametric approaches have been explored. Such approaches include forest density estimation (Liu et al., 2011), which assumes that the data has a forest (i.e. a collection of trees) structured graph. This assumption leads to a density which factorizes in a first order Markovian fashion

through a tree traversal of the graph. Another common semiparametric approach is to use a nonparanormal type model (Liu et al., 2009). This approach uses a Gaussian copula with a rank-based transformation and a sparse precision matrix. While both approaches are well-understood theoretically, their strong assumptions lead to inflexible models.

In order to provide greater flexibility with semiparametric models, recent work has employed deep learning for density estimation. The use of neural networks for density estimation dates back to Bishop (1994) and has seen success in speech (Zen & Senior, 2014; Uria, 2015), music (Boulanger-Lewandowski et al., 2012), etc. Typically such approaches use a network to learn the parameters of a parametric model for data. Recent work has also explored the application of deep learning to build density estimates in image data (Oord et al., 2016; Dinh et al., 2016). However, such approaches are heavily reliant on exploiting structure in neighboring pixels, often subsampling, reshaping or re-ordering data, and using convolutions to take advantage of neighboring correlations. Modern approaches for general density estimation in real-valued data include Uria et al. (2013; 2016); Germain et al. (2015); Gregor et al. (2014); Dinh et al. (2014); Kingma et al. (2016); Papamakarios et al. (2017).

NADE (Uria et al., 2013) is an RBM-inspired density estimator with a weight-sharing scheme across conditional densities on covariates. It may be written as a special case of LAM (5) with tied weights:

$$q_i\left(x_{i-1}, \ldots, x_1\right) = W_{<i} x_{<i} + b, \tag{16}$$

where $W_{<i} \in \mathbb{R}^{p \times i-1}$ is the weight matrix compose of the first $i-1$ columns of a shared matrix $W = (w_1, \ldots w_d)$. We note also that LAM and NADE are both related to fully visible sigmoid belief networks (Frey, 1998; Neal, 1992).

Even though the weight-sharing scheme in (16) reduces the number of parameters, it also limits the types of distributions one can model. Roughly speaking, the NADE weight-sharing scheme makes it difficult to adjust conditional distributions when expanding the conditioning set with a covariate that has a small information gain. We illustrate this by considering a simple 3-dimensional distribution: $x_1 \sim \mathcal{N}(0,1)$, $x_2 \sim \mathcal{N}(\text{sign}(x_1), \epsilon)$, $x_3 \sim \mathcal{N}\left(\mathbb{I}\left\{|x_1| < C_{0.5}\right\}, \epsilon\right)$, where $C_{0.5}$ is the 50% confidence interval of a standard Gaussian distribution, and $\epsilon > 0$ is some small constant. That is, x_2, and x_3 are marginally distributed as an equi-weighted bimodal mixture of Gaussian with means $-1, 1$ and $0, 1$, respectively. Due to NADE's weight-sharing linear model, it will be difficult to adjust h_2 and h_3 jointly to correctly model x_2 and x_3 respectively. However, given their additional flexibility, both LAM and RAM are able to adjust hidden states to remember and transform features as needed.

NICE (Dinh et al., 2014) and its successor Real NVP (Dinh et al., 2016) models assume that data is drawn from a latent independent Gaussian space and transformed. The transformation uses several "additive coupling" shifting on the second half of dimensions, using the first half of dimensions. For example NICE's additive coupling proceeds by splitting inputs into halves $x = (x_{<d/2}, x_{\geq d/2})$, and transforming the second half as an additive function of the first half:

$$z = \left(x_{<d/2}, x_{\geq d/2} + m(x_{<d/2})\right), \tag{17}$$

where $m(\cdot)$ is the output of a fully connected network. Inversion is simply a matter of subtraction $x = \left(z_{<d/2}, z_{\geq d/2} - m(z_{<d/2})\right)$. The full transformation is the result of stacking several of these additive coupling layers together followed by a final rescaling operation. Furthermore, as with the RNN shift transformation, the additive nature of (17) yields a simple determinant, $\det \frac{\mathrm{d}z}{\mathrm{d}x} = 1$.

MAF (Papamakarios et al., 2017) identified that Gaussian conditional autoregressive models for density estimation can be seen as transformations. This enabled them to stack multiple autoregressive models that increases flexibility. However, stacking Gaussian conditional autoregressive models amounts to just stacking shift and scale transformations. Unlike MAF, in the TAN framework we not only propose novel and more complex equivalence like Recurrent Transformation (Sec. 2.2), but also systematically composing stacks of such transformations with flexible autoregressive models.

There are several methods for obtaining samples from an unknown distribution that by-pass density estimation. For instance, generative adversarial networks (GANs) apply a (typically noninvertible) transformation of variables to a base distribution by optimizing a minimax loss (Goodfellow, 2016; Kingma et al., 2016). Samples can also be obtain from methods that compose graphical models with deep networks (Johnson et al., 2016; Al-Shedivat et al., 2017). Furthermore, one can also obtain samples with only limited information about the density of interest using methods such as Markov chain Monte Carlo (Neal, 1993), Hamiltonian Monte Carlo (Neal, 2010), stochastic variants (Dubey et al., 2016), etc.

4. Experiments

We now present empirical studies for our TAN framework in order to establish (i) the superiority of TANs over one-prong approaches (Sec. 4.1), (ii) that TANs are accurate on real world datasets (Sec. 4.2), (iii) the importance of various components of TANs, (iv) that TANs are easily amenable to various tasks (Sec. 4.4), such as learning a parametric family of distributions and being able to generalize over unseen parameter values (Sec. 4.5).

Methods We study the performance of various instantiation of TANs using different combinations of conditional models $p\left(q_i(x) \,|\, h_i\right)$ and various transformations $q(\cdot)$. In particular the following conditional models were considered: `LAM`, `RAM`, `Tied`, `MultiInd`, and `SingleInd`. Here, `LAM`, `RAM`, and `Tied` are as described in equations (5), (6), and (16), respectively. `MultiInd` takes $p\left(q_i(x) \,|\, h_i\right)$ to be $p\left(q_i(x) \,|\, \text{MM}(\theta_i)\right)$, that is we shall use d distinct independent mixtures to model the transformed covariates.

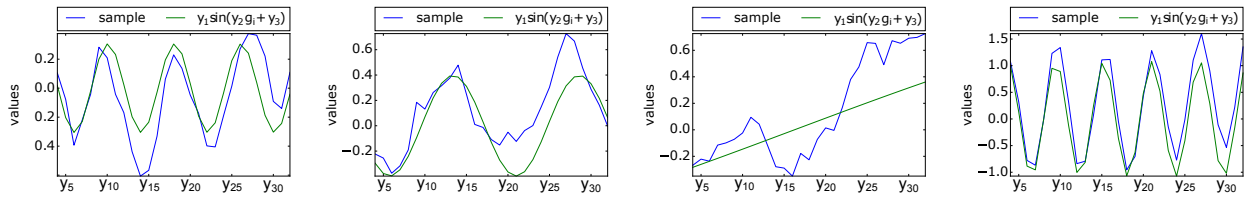

Figure 2. `RNN+4xSRNN+Re & RAM` model samples. Each plot shows a single sample. We plot the sample values of unpermuted dimensions $y_4, \ldots, y_{32} \,|\, y_1, y_2, y_3$ in blue and the expected value of these dimensions (i.e. without the Markovian noise) in green. One may see that the model is able to correctly capture both the sinusoidal and random walk behavior of our data.

Similarly, `SingleInd` takes $p\left(q_i(x) \,|\, h_i\right)$ to be $p\left(q_i(x)\right)$, the density of a standard single component. For transformations we considered: `None`, `RNN`, `2xRNN`, `4xAdd+Re`, `4xSRNN+Re`, `RNN+4xAdd+Re`, and `RNN+4xSRNN+Re`. `None` indicates that no transformation of variables was performed. `RNN` and `2xRNN` perform a single recurrent transformation (9), and two recurrent transformations with a reversal permutation in between, respectively. Following (Dinh et al., 2014), `4xAdd+Re` performs four additive coupling transformations (17) with reversal permutations in between followed by a final element-wise rescaling: $x \mapsto x * \exp(s)$, where s is a learned variable. Similarly, `4xSRNN+Re`, instead performs four recurrent shift transformations (12). `RNN+4xAdd+Re`, and `RNN+4xSRNN+Re` are as before, but performing an initial recurrent transformation. Furthermore, we also considered performing an initial linear transformation (8). We flag this by prepending an `L` to the transformation; e.g. `L RNN` denotes a linear transformation followed by a recurrent transformation.

Implementation Models were implemented in Tensorflow (Abadi et al., 2016)[1]. Both RAM conditional models as well as the RNN shift transformation make use of the standard `GRUCell` GRU implementation. We take the mixture models of conditionals (2) to be mixtures of 40 Gaussians. We optimize all models using the `AdamOptimizer` (Kingma & Ba, 2014) with an initial learning rate of 0.005. Training consisted of $30\,000$ iterations, with mini-batches of size 256. The learning rate was decreased by a factor of 0.1, or 0.5 (chosen via a validation set) every $5\,000$ iterations. Gradient clipping with a norm of 1 was used. After training, the best iteration according to the validation set loss was used to produce the test set results.

4.1. Synthetic

To showcase the strengths of TANs and short-comings of only conditional models & only transformations, we carefully construct two synthetic datasets

Data Generation Our first dataset, consisting of a Markovian structure that features several exploitable correlations among covariates, is constructed as: $y_1, y_2, y_3 \sim \mathcal{N}(0, 1)$ and $y_i \,|\, y_{i-1}, \ldots, y_1 \sim f(i, y_1, y_2, y_3) + \epsilon_i$ for $i > 3$ where $\epsilon_i \sim \mathcal{N}(\epsilon_{i-1}, \sigma)$, $f(i, y_1, y_2, x_3) = y_1 \sin(y_2 g_i + y_3)$, and g_i's are equi-spaced points on the unit interval. That is, instances are sampled using random draws of amplitude, frequency, and shift covariates y_1, y_2, y_3, which determine the mean of the other covariates, $y_1 \sin(y_2 g_i + y_3)$, stemming from function evaluations on a grid, and random noise ϵ_i with a Gaussian random walk. The resulting instances contain many correlations as visualized in Fig. 2. To further exemplify the importance of employing conditional and transformations in tandem, we construct a second dataset with much fewer correlations. In particular, we use a star-structured graphical model where fringe nodes are very uninformative of each-other and estimating the distribution of the fringe vertices are difficult without conditioning on all the center nodes. To construct the dataset: divide the covariates into disjoint center and vertex sets $C = \{1, \ldots, 4\}$, $V = \{5, \ldots, d\}$ respectively. For center nodes $j \in C$, $y_j \sim \mathcal{N}(0, 1)$. Then, for $j \in V$, $y_j \sim \mathcal{N}(f_j(w_j^T y_C), \sigma)$ where f_j is a fixed step function with 32 intervals, $w_j \in \mathbb{R}^4$ is a fixed vector, and $y_C = (y_1, y_2, y_3, y_4)$. In both datasets, to test robustness to correlations from distant (by index) covariates, we observe covariates that are shuffled using a fixed permutation π chosen ahead of time: $x = (y_{\pi_1}, \ldots, y_{\pi_d})$. We take $d = 32$, and the number of training instances to be $100\,000$.

Observations We detail the mean log-likelihoods on a test set for TANs using various combinations of conditional models and transformations in Appendix, Tab. 2 and Tab. 3 respectively. We see that both `LAM` and `RAM` conditionals are providing most of the top models. We observe good samples from the best performing model as shown in Fig. 2. Particularly in second dataset, simpler conditional methods are unable to model the data well, suggesting that the complicated dependencies need a two-prong TAN approach. We observe a similar pattern when learning over the star data with $d = 128$ (see Appendix, Tab. 4).

4.2. Efficacy on Real World Data

We performed several real-world data experiments and compared to several state-of-the-art density estimation methods to substantially improved performance of TAN.

Datasets We carefully followed (Papamakarios et al., 2017) and code (MAF Git Repository) to ensure that we operated over the same instances and covariates for each of the datasets considered in (Papamakarios et al., 2017). Specifically we performed unconditional density estimation on four datasets from UCI machine learning repository[2]: `power`:

[1] See `https://github.com/lupalab/tan`.

[2] `http://archive.ics.uci.edu/ml/`

Table 1. Average test log-likelihood comparison of TANs with baselines MADE, Real NVP, MAF as reported by (Papamakarios et al., 2017). For TANs the best model is picked using validation dataset and are reported here. Parenthesized numbers indicate number of transformations used. Standard errors with 2σ are shown. Largest values per dataset are shown in **bold**.

	POWER d=6; N=2,049,280	GAS d=8; N=1,052,065	HEPMASS d=21; N=525,123	MINIBOONE d=43; N=36,488	BSDS300 d=63; N=1,300,000
MADE	-3.08 ± 0.03	3.56 ± 0.04	-20.98 ± 0.02	-15.59 ± 0.50	148.85 ± 0.28
MADE MoG	0.40 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
Real NVP (5)	-0.02 ± 0.01	4.78 ± 1.80	-19.62 ± 0.02	-13.55 ± 0.49	152.97 ± 0.28
Real NVP (10)	0.17 ± 0.01	8.33 ± 0.14	-18.71 ± 0.02	-13.84 ± 0.52	153.28 ± 1.78
MAF (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	-11.75 ± 0.44	155.69 ± 0.28
MAF (10)	0.24 ± 0.01	10.08 ± 0.02	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
TAN	**0.60 ± 0.01** 5x L+ReLU+SRNN+Re & RAM	**12.06 ± 0.02** 5x L+ReLU+SRNN+Re & RAM	**−13.78 ± 0.02** 5x L+ReLU+SRNN+Re & RAM	**−11.01 ± 0.48** 4xSRNN + Re & RAM	**159.80 ± 0.07** 5x L+ReLU+SRNN+Re & RAM

Containing electric power consumption in a household over 47 months. `gas`: Readings of 16 chemical sensors exposed to gas mixtures. `hepmass`: Describing Monte Carlo simulations for high energy physics experiments. `minibone`: Containing examples of electron neutrino and muon neutrino. We also used `BSDS300` which were obtained from extracting random 8×8 monochrome patches from the BSDS300 datasets of natural images (Martin et al., 2001). These are multivariate datasets from a varied set of sources meant to provide a broad picture of performance across different domains. Here, we used a batch size of 1024 with 60K training iterations. We saw great performance by using multiple successions of a linear transformation, followed by an element-wise leaky transformation (as in eq. 9), a recurrent shift transformation (12), and an element-wise rescale transformation. Thus in addition, we used a model with 5 such stacked transformations (`5x L+ReLU+SRNN+Re`). Further, to demonstrate that our proposed models can even be used to model high dimensional data and produce coherent samples, we consider image modeling task, treating each image as a flattened vector. We consider 28×28 grayscale images of MNIST digits and 32×32 natural colored images of CIFAR-10. Following Dinh et al. (2014), we dequantize pixel values by adding noise and rescaling.

Metric We use the average test log-likelihoods of the best TAN model selected using a validation set and compare to values reported by (Papamakarios et al., 2017) for MADE (Germain et al., 2015), Real NVP (Dinh et al., 2016), and MAF (Papamakarios et al., 2017) methods for each dataset. For images, we use transformed version of test log-likelihood, called bits per pixel, which is more popular. In order to calculate bits per pixel, we need to convert the densities returned by a model back to image space in the range [0, 255], for which we use the same logit mapping provided in Papamakarios et al. (2017, Appendix E.2).

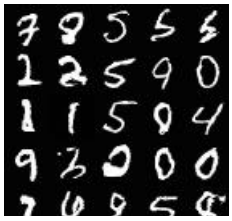

Figure 3. Samples from best TAN model.

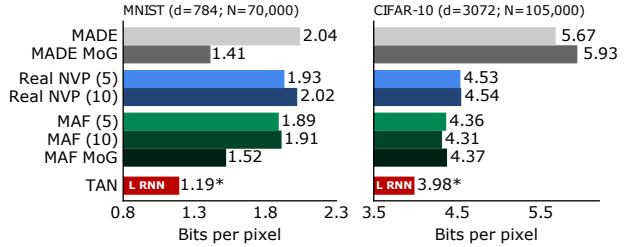

Figure 4. Bits per pixel for models (lower is better) using logit transforms on MNIST & CIFAR-10. MADE, Real NVP, and MAF values are as reported by (Papamakarios et al., 2017). The best achieved value is denoted by *.

Observations Tab. 1 and Fig. 4 shows our results on various multivariate datasets and images respectively, with error bars computed over 5 runs. As can be seen, our TAN models are considerably outperforming other state-of-the-art methods across all multivariate as well as image datasets, justifying our claim of utilizing both complex transformations and conditionals. Furthermore, we plot samples for MNIST case in Fig. 3. We see that TAN is able to capture the structure of digits with very few artifacts in samples, which is also reflected in the likelihoods.

4.3. Ablation Study

To study how different components of the models affect the log-likelihood, we perform a comprehensive ablation study across different datasets.

Datasets We used multiple datasets from the UCI machine learning repository[3] and Stony Brook outlier detection datasets collection (ODDS)[4] to evaluate log-likelihoods on test data. Broadly, the datasets can be divided into: **Particle acceleration**: `higgs`, `hepmass`, and `susy` datasets where generated for high-energy physics experiments using Monte Carlo simulations; **Music**: The `music` dataset contains timbre features from the million song dataset of mostly

[3] `http://archive.ics.uci.edu/ml/`
[4] `http://odds.cs.stonybrook.edu`

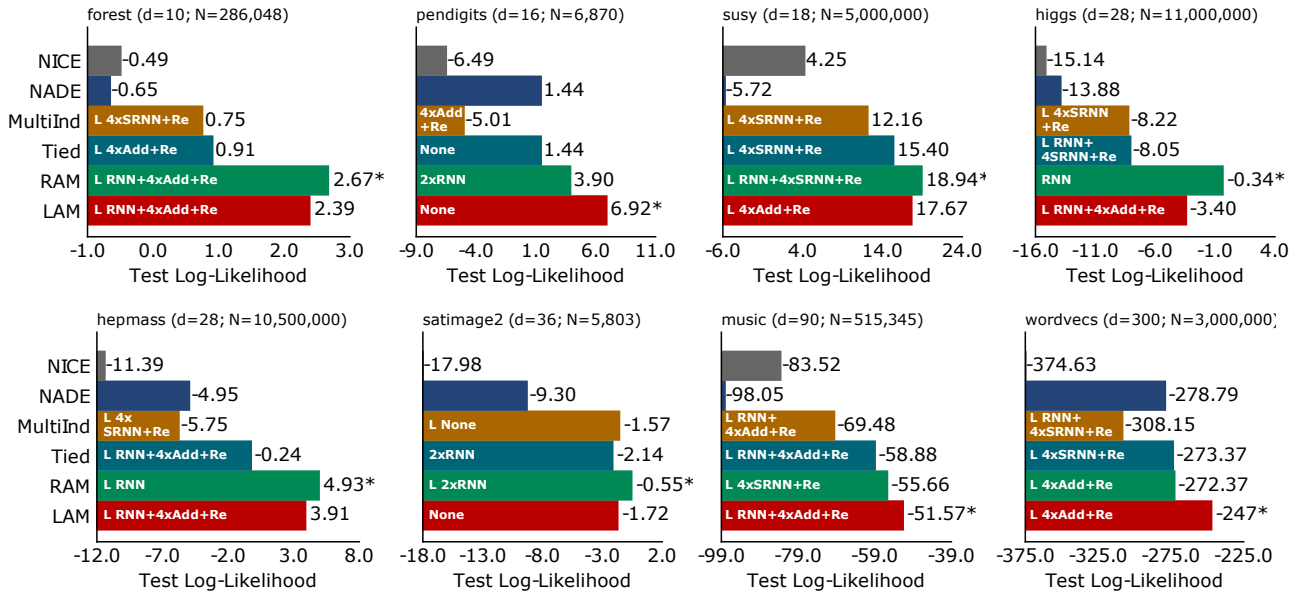

Figure 5. Ablation Study of various components TAN. For each dataset and each conditional model, top transformations is selected using log-likelihoods on a validation set. The picked transformation is reported within the bars for each conditional. $*$ denotes the best model for each dataset picked by validation. Simple conditional `MultiInd`, always lags behind sophisticated conditionals such as `LAM` & `RAM`.

commercial western song tracks from the year 1922 to 2011; (Bertin-Mahieux et al., 2011). **Word2Vec**: `wordvecs` consists of 3 million words from a Google News corpus. Each word represented as a 300 dimensional vector trained using a word2vec model[5]. **ODDS datasets**: We used several ODDS datasets–`forest`, `pendigits`, `satimage2`. These are multivariate datasets from varied set of sources meant to provide a broad picture of performance across anomaly detection tasks. To not penalize models for low likelihoods on outliers in ODDS, we removed anomalies from test sets.

As noted in (Dinh et al., 2014), data degeneracies and other corner-cases may lead to arbitrarily low negative log-likelihoods. Thus, we remove discrete features, standardize, and add Gaussian noise (stddev of 0.01) to training sets.

Observations We report average test log-likelihoods in Fig. 5 for each dataset and conditional model for the top transformations picked on a validation dataset. The tables with test log-likelihoods for all combinations of conditional models and transformations for each dataset is in Appendix Tab. 6-12. We observe that the best performing models in real-world datasets are those that incorporate a flexible transformation *and* conditional model. In fact, the best model in each of the datasets considered always has `LAM` or `RAM` autoregressive components. Each row of these tables show that using a complex conditional is always better than using restricted, independent conditionals. Similarly, each column of the table shows that for a given conditional, it is better to pick a complex transformation rather than having no transformation. It is interesting to note that many of these top models also contain a linear transformation. Of course, linear transformations of variables are common to most parametric models, however they have been under-explored in the context of autoregressive density estimation. Our methodology for efficiently learning linear transformations coupled with their strong empirical performance encourages their inclusion in autoregressive models for most datasets.

Finally, we pick the "overall" winning combination of transformations and conditionals. For this we compute the fraction of the top likelihood achieved by each transformation t and conditional model m for dataset D: $s(t, m, D) = \exp(l_{t,m,D}) / \max_{a,b} \exp(l_{a,b,D})$, where $l_{t,m,D}$ is the test log-likelihood for t, m on D. We then average S over the datasets: $S(t, m) = \frac{1}{T} \sum_D S(t, m, D)$, where T is the total number of datasets and reported all these score in Appendix Tab. 5. This provides a summary of which models performed better over multiple datasets. In other words, the closer this score is to 1 for a model means the more datasets for which the model is the best performer. We see that `RAM` conditional with `L RNN` transformation, and `LAM` conditional with `L RNN+4xAdd+Re` were the two best performers.

4.4. Anomaly Detection

Next, we apply density estimates to anomaly detection. Typically anomalies or outliers are data-points that are unlikely given a dataset. In terms of density estimations, such a task is framed by identifying which instances in a dataset have a low corresponding density. That is, we shall label an instance x, as an anomaly if $\hat{p}(x) \leq t$, where $t \geq 0$ is some threshold and $\hat{p}$ is the density estimate based on training data. Note that this approach is trained in an unsupervised fashion. Density estimates were evaluated on test data with

[5] https://code.google.com/archive/p/word2vec/

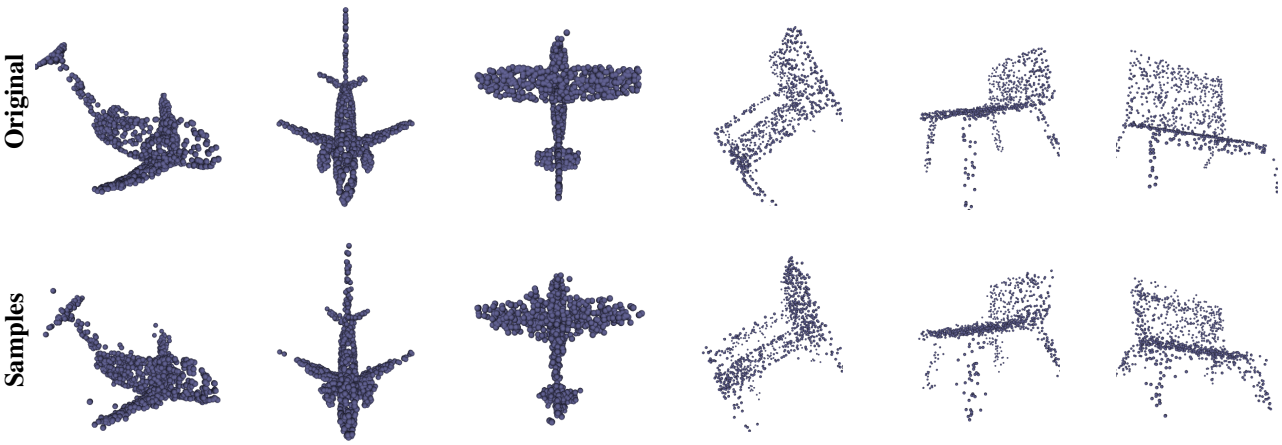

Figure 6. Qualitative samples obtained from TANs for the task of learning parametric family of distributions where we treat each category of objects as a family and each point cloud for an object as the sample set. Top row shows unseen test point clouds and bottom row represents samples produced from TANs for these inputs. Presence of few artifacts in samples of unseen objects indicates a good fit.

anomaly/non-anomaly labels on instances. We used thresholded log-likelihoods on the test set to compute precision and recall. We use the average-precision metric and show our results in Fig. 7. TAN performs the best on all three datasets. Beyond providing another interesting use for our density estimates, seeing good performance in these outlier detection tasks further demonstrates that our models are learning semantically meaningful patterns.

4.5. Learning Parametric Family of Distributions

To further demonstrate flexibility of TANs, we consider a new task of learning parametric family of distributions together. Suppose we have a family of density $\mathcal{P}_\theta$. We assume in training data there are N sets $X_1, ..., X_N$, where the n-th set $X_n = \{x_{n,1}, ..., x_{n,m_n}\}$ consists of m_n i.i.d. samples from density $\mathcal{P}_{\theta_n}$, *i.e.* X_n is a set of sample points, and $x_{n,j} \sim \mathcal{P}_{\theta_n}, j = 1, ..., m_n$. We assume that we do not have access to underlying true parameters θ_n. We want to jointly learn the density estimate and parameterization of the sets to predict even for sets coming from unseen values of θ.

We achieve this with a novel approach that models each set X_i with $p(\cdot|\phi(X_i))$ where p is a shared TAN model for the family of distributions and $\phi(X_i)$ are a learned embedding (parameters) for the ith set with DeepSets (Zaheer et al., 2017). In particular, we use a permutation invariant network of DeepSets parameterized by W_1 to extract the embedding $\phi(X)$ for the given sample set X. The embedding is then fed along with sample set to TAN model parameterized by W_2. We then optimize the following modified objective:

$$\min_{W_1,W_2} -\frac{1}{N}\sum_i \frac{1}{m_i}\sum_j \log p_{W_2}\left(x_{ij}|\phi_{W_1}(X_{i\setminus j})\right). \quad (18)$$

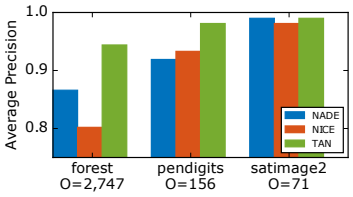

Figure 7. Average precision score on outlier detection datasets. For each dataset, the best performing TAN model picked using likelihood on a validation set, is shown.

We attempt to model point-cloud representation of objects from ShapeNet (Chang et al., 2015). We produce point-clouds with 1000 particles each $(x, y, z$-coordinates) from the mesh representation of objects using the point-cloud-library's sampling routine (Rusu & Cousins, 2011). We consider each category of objects (*e.g.* aeroplane, chair, car) as a family and each point cloud for each object in the category as a sample set. We train a TAN and only show samples in Fig. 6 produced for unseen test sets, as there are neither any baselines for this task nor ground truth for likelihood. From the samples, we see that our model is able to capture the structure of different kinds of unseen aeroplanes and chairs, with very few artifacts in samples, which reflects a good fit.

Note that this task is subtly different from conditional density estimation as we do not have access to class/parameter values during training. Also we want to caution users against using this method when the test sample set is very different from training or comes from a different family distribution.

5. Conclusion

In this work, we showed that we can significantly improve density estimation for real valued data by jointly leveraging transformations of variables with autoregressive models and proposed novel modules for both. We systematically characterized various modules and evaluated their contributions in a comprehensive ablation study. This exercise not only re-emphasized the benefits of joint modeling, but also revealed some straightforward modules and combinations thereof, which are empirically good, but were missed earlier, *e.g.* the untied linear conditionals. Finally we introduced a novel data driven framework for learning a family of distributions.

Acknowledgements

This research is partly funded by DOE grant DESC0011114, NSF IIS1563887, NIH R01GM114311, NSF IIS1447676, and the DARPA D3M program.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Al-Shedivat, M., Dubey, A., and Xing, E. P. Contextual explanation networks. *arXiv preprint arXiv:1705.10301*, 2017.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *ISMIR*, volume 2, pp. 10, 2011.

Bishop, C. M. Mixture density networks. *Technical Report*, 1994.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning*, 2012.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Dubey, K. A., Reddi, S. J., Williamson, S. A., Poczos, B., Smola, A. J., and Xing, E. P. Variance reduction in stochastic gradient langevin dynamics. In *Advances in neural information processing systems*, pp. 1154–1162, 2016.

Frey, B. J. *Graphical models for machine learning and digital communication*. MIT press, 1998.

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-15)*, pp. 881–889, 2015.

Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. Deep autoregressive networks. *ICML*, 2014.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pp. 2946–2954, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P., Salimans, T., and Welling, M. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

Liu, H., Lafferty, J., and Wasserman, L. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10(Oct):2295–2328, 2009.

Liu, H., Xu, M., Gu, H., Gupta, A., Lafferty, J., and Wasserman, L. Forest density estimation. *Journal of Machine Learning Research*, 12(Mar):907–951, 2011.

MAF Git Repository. The maf git repository. https://github.com/gpapamak/maf. Accessed: 2017-12-03.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pp. 416–423. IEEE, 2001.

Neal, R. M. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.

Neal, R. M. Probabilistic inference using markov chain monte carlo methods. 1993.

Neal, R. M. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

Rusu, R. B. and Cousins, S. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

Uria, B. Connectionist multivariate density-estimation and its application to speech synthesis., 2015.

Uria, B., Murray, I., and Larochelle, H. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pp. 2175–2183, 2013.

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *ICML*, pp. 467–475, 2014.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

Wasserman, L. All of nonparametric statistics, 2007.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems*, pp. 3394–3404, 2017.

Zen, H. and Senior, A. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 3844–3848. IEEE, 2014.