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, $p(x_i | x_{i-1}, \ldots); \text{or } b)$ 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.

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 $p(z_i | z_{i-1}, \ldots) = f(z_i)$ (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, \( \text{MM}(\theta(x_{i-1}, \ldots, x_1)) \), with parameters \( \theta \) depending on \( x_{i-1}, \ldots, x_1 \):

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

\[
\theta(x_{i-1}, \ldots, x_1) = f(h_i) \tag{3}
\]

\[
h_i = g_i(x_{i-1}, \ldots, x_1), \tag{4}
\]

where \( f(\cdot) \) is a fully connected network that may use an 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):

\[
g_i(x_{i-1}, \ldots, x_1) = W(i)x_{ci} + b, \tag{5}
\]

where \( W(i) \in \mathbb{R}^{p \times (i-1)} \), \( b \in \mathbb{R}^p \), and \( x_{ci} = (x_{i-1}, \ldots, x_1)^T \).

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:

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

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:
where $\det \frac{d\mathbf{g}}{d\mathbf{z}}$ 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:

$$z = A\mathbf{x} + \mathbf{b},$$

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 an upper triangular matrix with arbitrary diagonals. As a function of $L$, $U$ we have that $\det \frac{d\mathbf{g}}{d\mathbf{z}} = \prod_{i=1}^d U_{ii}$; 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 straightforward. We consider the following form to an RNN transformation:

$$z_i = r_\alpha \left( \mathbf{y}_i + w^T \mathbf{s}_{i-1} + b \right), \quad s_i = r \left( \mathbf{u}_i + v^T \mathbf{s}_{i-1} + a \right),$$

where $r_\alpha$ is a leaky ReLU unit $r_\alpha(t) = \mathbb{I}(t < 0) a t + \mathbb{I}(t \geq 0) t$, $r$ is a standard ReLU unit, $s \in \mathbb{R}^d$ is the hidden state, $\mathbf{u}, \mathbf{w}, a$ are scalars, and $\mathbf{w}, \mathbf{v} \in \mathbb{R}^d$ 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):

$$x_i = \frac{1}{y} \left( r_\alpha^{-1} \left( z_i^{(r)} \right) - w^T \mathbf{s}_{i-1} - b \right),$$

$$s_i = r \left( \mathbf{u}_i + v^T \mathbf{s}_{i-1} + a \right).$$

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

$$\det \frac{d\mathbf{g}}{d\mathbf{x}} = y \prod_{i=1}^d r_\alpha' \left( \mathbf{y}_i + w^T \mathbf{s}_{i-1} + b \right),$$

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:

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

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

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

The Jacobian is again lower triangular, however due to the additive nature of (12), we have a unit diagonal. Thus, $\det \frac{d\mathbf{g}}{d\mathbf{z}} = 1$. 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 consider stacking (i.e., composing) several transformations $q = q^{(1)} \circ \ldots \circ q^{(T)}$ and renormalizing:

$$p(x_1, \ldots, x_d) = \prod_{t=1}^T \left[ \det \frac{d\mathbf{g}^{(t)}}{d\mathbf{g}^{(t-1)}} \right] \prod_{i=1}^d p(q_t(x) | q_{t-1}(x), \ldots, q_1(x)),$$

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 $q_t$s 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:

$$- \log p(x_1, \ldots, x_d) = - \sum_{i=1}^d \log \left[ \det \frac{d\mathbf{g}^{(i)}}{d\mathbf{g}^{(i-1)}} \right] - \sum_{i=1}^d \log p(q_i(x) | h_i),$$

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 ($d \leq 3$). 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 nonparametric 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:

\[
q_i(x_{i-1}, \ldots, x_1) = W_{ci}x_{ci} + b_i,
\]

where \(W_{ci} \in \mathbb{R}^{h \times 1}\) is the weight matrix composed 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:

\[
x_1 \sim N(0, 1), \quad x_2 \sim N(\text{sign}(x_1), \epsilon), \quad x_3 \sim N(\{ |x_1| < C_{0.5} \}, \epsilon),
\]

where \(C_{0.5} = 0.5\) is a 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 \(x = (x_{<d/2}, x_{d/2})\), and transforming the second half as an additive function of the first half:

\[
z = (x_{<d/2}, x_{d/2} + m(x_{<d/2}))
\]

where \(m(\cdot)\) is the output of a fully connected network. Inversion is simply a matter of subtraction \(x = (z_{<d/2}, z_{d/2} - m(z_{d/2}))\). 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, \(\det \frac{\partial z}{\partial x} = 1\).

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 instantiations of TANS using different combinations of conditional models \(p(q_i(x) | h_i)\) 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 \(p(q_i(x) | h_i)\) to be \(p(q_i(x) | \mathbf{MM}(\theta_i))\), that is we shall use \(d\) distinct independent mixtures to model the transformed covariates.
Similarly, \textit{SingleInd} takes \( p(q_i(x)|h_i) \) to be \( p(q_i(x)) \), the density of a standard single component. For transformations we considered: \text{None}, RNN, 2xRNN, 4xAdd+Re, 4xSRNN+Re, RNN+4xAdd+Re, and RNN+4xSRNN+Re. \textit{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 \cite{Dinh2014}, 4xAdd+Re performs four additive coupling transformations (17) with reversal permutations in between followed by a final element-wise rescaling: \( x \mapsto x \times \exp(s) \), 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.

\textbf{Implementation} Models were implemented in TensorFlow \cite{Abadi2016}. 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 4 Gaussians. We optimize all models using the AdamOptimizer \cite{Kingma2014} 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.

\textbf{4.1. Synthetic} To showcase the strengths of TANs and short-comings of only conditional models & only transformations, we carefully construct two synthetic datasets.

\textbf{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 \( \epsilon_i \sim \mathcal{N}(\epsilon_{i-1}, \sigma) \), \( f(i, y_1, y_2, x_3) = y_1 \sin(y_2 y_3 + y_5) \), 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 \( y_1, y_2, y_3, y_4 \), which determine the mean of the other covariates, \( y_i \sin(y_{2} y_i + y_5) \), stemming from function evaluations on a grid, and random noise \( \epsilon_i \) 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 \( \pi \) 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.

\textbf{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).

\textbf{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.

\textbf{Datasets} We carefully followed \cite{Papamakarios2017} and code (MAF Git Repository) to ensure that we operated over the same instances and covariates for each of the datasets considered in \cite{Papamakarios2017}. Specifically we performed unconditional density estimation on four datasets from UCI machine learning repository:\footnote{See https://github.com/lupalab/tan.}

\footnote{\url{http://archive.ics.uci.edu/ml/}}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Figure2.png}
\caption{RNN+4xSRNN+Re & RAM model samples. Each plot shows a single sample. We plot the sample values of unpermuted dimensions \( y_4, \ldots, y_{12}|y_1, y_2, y_3 \) 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.}
\end{figure}
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.

| Metric | Power | GAS | HEPMass | MINIBOONE | BSDS300 |
|--------|-------|-----|---------|-----------|---------|
| 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 | 12.06 ± 0.02 | -13.78 ± 0.02 | -11.01 ± 0.48 | 159.80 ± 0.07 |

- **Bits per pixel**

- **TAN**
- **MADE**
- **MAF MoG**
- **MAF (10)**
- **MAF (5)**
- **Real NVP (10)**
- **Real NVP (5)**
- **MADE MoG**

---

**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.

---

**Metric**

We use the average test log-likelihood 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).

---

**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\(^1\) and Stony Brook outlier detection datasets collection (ODDS)\(^2\) to evaluate log-likelihoods on test data. Broadly, the datasets can be divided into: **Particle acceleration**: higgs, hepams, 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

\(^1\)http://archive.ics.uci.edu/ml/
\(^2\)http://odds.cs.stonybrook.edu
Transformation Autoregressive Networks

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.

As noted in (Dinh et al., 2014), data degeneracies and other corner-cases may lead to arbitrarily low negative log-likelihoods on outliers in ODDS, we removed anomalies from test sets.

As noted in (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. 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.

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 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 $\hat{p}(x) < t$, where $t > 0$ 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

---

3 http://code.google.com/archive/p/word2vec/
We achieve this with a novel approach that models each set with DeepSets parameterized by $W$. To further demonstrate flexibility of TANs, we consider a likelihood on a validation set, is shown. For each dataset, the best performing TAN model picked using $\phi$ of DeepSets parameterized by $W$. In particular, we use a permutation invariant network $\phi$ family of distributions and $X$ to predict even for sets coming from unseen values of $\theta$. We want to jointly learn the density estimate and parameterization of the sets. We assume that we do not have access to underlying true parameters $\theta$. We want to jointly learn the density estimate and parameterization of the sets to predict even for sets coming from unseen values of $\theta$.

We achieve this with a novel approach that models each set $X_i$ with $p(\phi(X_i))$ where $\phi$ is a shared TAN model for the family of distributions and $\phi(X_i)$ are a learned embedding (parameters) for the $i$th 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:

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

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.
A. Appendix

Below we detail the results on several datasets using different combinations of transformations and autoregressive conditional models. Each additive coupling transformation uses a fully connected network with two hidden layers of 256 units. RNN transformations use a hidden state with 16 units. SingleInd conditional models each dimension’s conditional as independent mixtures with 40 components (each with mean, scale, and weight parameter). RAM, LAM, and Tied conditional models each had a hidden state with 120 units that was fed through two fully connected layers each with 120 units to produce the parameters of the mixtures with 40 components. The RAM hidden state was produced by a GRU with 256 units. LAM and Tied hidden states came through a linear map as discussed above.

Table 2. Held out test log-likelihoods for the Markovian dataset. The superscripts denote rankings of log-likelihoods on the validation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM     | RAM     | TIED    | MultiInd | SingleInd |
|---------------|---------|---------|---------|----------|-----------|
| None          | 14.319  | -29.950 | -0.612  | -41.472  |           |
| L None        | 15.486(9)| 14.538  | 10.906  | 5.252    | -9.426    |
| RNN           | 14.777  | -37.716 | 11.075  | -30.491  | -37.038   |
| L RNN         | 15.658(5)| 10.354  | 10.910  | 5.370    | 3.310     |
| 2xRNN         | 14.683  | 13.698  | 11.493  | -18.448  | -34.268   |
| L 2xRNN       | 15.474(8)| 15.752(3)| 12.316  | 5.385    | 3.739     |
| 4xAdd+Re      | 15.269  | 12.257  | 12.912  | 12.446   | 11.625    |
| L 4xAdd+Re    | 15.683(6)| 12.594  | 13.845  | 12.768   | 12.069    |
| 4xSRNN+Re     | 14.829  | 14.381  | 11.798  | 11.738   | 12.932    |
| L 4xSRNN+Re   | 15.289  | 16.202(1)| 12.748  | 15.415(10)| 13.908    |
| RNN+4xAdd+Re  | 15.171  | 12.991  | 14.455  | 11.467   | 10.382    |
| L RNN+4xAdd+Re| 15.078  | 12.655  | 14.415  | 12.886   | 12.315    |
| RNN+4xSRNN+Re | 14.968  | 16.216(2)| 12.590  | 15.589(4) | 14.231    |
| L RNN+4xSRNN+Re| 15.429  | 15.566(7)| 14.179  | 14.528   | 13.961    |

Table 3. Held out test log-likelihoods for star 32d dataset. The superscript denotes ranking of log-likelihood on cross validation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM     | RAM     | TIED    | MultiInd | SingleInd |
|---------------|---------|---------|---------|----------|-----------|
| None          | -2.041  | 2.554   | -10.454 | -29.485  |           |
| L None        | 5.454   | 8.247   | -7.858  | -26.988  | -38.952   |
| RNN           | -1.276  | 2.762   | -6.292  | -25.946  | -41.275   |
| L RNN         | 7.775   | 6.335   | -1.157  | -25.986  | -34.408   |
| 2xRNN         | 3.705   | 8.032   | -0.565  | -25.100  | -38.490   |
| L 2xRNN       | 14.878(3)| 9.946   | 0.901   | -23.772  | -33.075   |
| 4xAdd+Re      | 13.278(6)| 11.561(9)| 7.146   | -16.740  | -21.332   |
| L 4xAdd+Re    | 15.728(2)| 12.444(7)| 9.031   | -6.091   | -11.225   |
| 4xSRNN+Re     | 3.496   | 8.429   | -1.380  | -15.590  | -23.712   |
| L 4xSRNN+Re   | 16.042(1)| 9.939(10)| 5.598   | -12.530  | -16.889   |
| RNN+4xAdd+Re  | 14.071(5)| 14.123(4)| 6.868   | -14.773  | -20.483   |
| L RNN+4xAdd+Re| 11.819(8)| 9.253   | 2.638   | -7.662   | -14.530   |
| RNN+4xSRNN+Re | -0.679  | 3.320   | -6.172  | -12.879  | -19.204   |
| L RNN+4xSRNN+Re| 7.433   | 7.324   | 3.554   | -10.427  | -15.243   |
Table 4. Held out test log-likelihood for Star 128d dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM     | RAM     | TIED    | MultiInd | SingleInd |
|---------------|---------|---------|---------|----------|-----------|
| None          | 15.671  | 15.895  | -83.115 | -128.238 | -159.391  |
| L None        | 57.881  | -82.100 | -28.206 | -123.939 | -159.391  |
| RNN           | 18.766  | 48.295  | -22.485 | -113.181 | -178.641  |
| L RNN         | 66.076(9) | -49.084 | 31.136  | -107.083 | -155.324  |
| 2xRNN         | 27.295  | 45.834  | -11.930 | -113.210 | -178.331  |
| L 2xRNN       | 85.681(3) | -84.524 | 30.974  | -105.368 | -162.635  |
| 4xAdd+Re      | 77.195(6) | 61.947(10) | 16.062 | -75.206  | -111.542  |
| L 4xAdd+Re    | 88.837(1) | -21.882 | 20.234  | -65.694  | -96.071   |
| 4xSRNN+Re     | 33.577  | -98.796 | 3.256   | -88.912  | -98.936   |
| L 4xSRNN+Re   | 86.375(2) | 76.968(5) | 33.481  | -85.590  | -93.086   |
| RNN+4xAdd+Re  | 66.540(8) | -57.861 | -16.277 | -75.491  | -114.729  |
| L RNN+4xAdd+Re| 80.063(4) | 32.104  | 21.944  | -71.933  | -100.384  |
| RNN+4xSRNN+Re | 21.719  | -87.335 | -6.517  | -76.459  | -85.422   |
| L RNN+4xSRNN+Re| 72.463(7) | 56.201  | 26.269  | -71.843  | -91.695   |

Table 5. Average performance percentage score for each model across all datasets. Note that this measure is not over a logarithmic space.

| Transformation | LAM     | RAM     | TIED    | MultiInd | SingleInd | MAX   |
|---------------|---------|---------|---------|----------|-----------|-------|
| None          | 0.218   | 0.118   | 0.006   | 0.000    | 0.000     | 0.218 |
| L None        | 0.154   | 0.179   | 0.026   | 0.051    | 0.001     | 0.179 |
| RNN           | 0.086   | 0.158   | 0.014   | 0.001    | 0.000     | 0.158 |
| L RNN         | 0.173   | 0.540   | 0.014   | 0.040    | 0.013     | 0.540 |
| 2xRNN         | 0.151   | 0.101   | 0.045   | 0.001    | 0.000     | 0.151 |
| L 2xRNN       | 0.118   | 0.330   | 0.015   | 0.045    | 0.025     | 0.330 |
| 4xAdd+Re      | 0.036   | 0.047   | 0.015   | 0.010    | 0.006     | 0.047 |
| L 4xAdd+Re    | 0.153   | 0.096   | 0.025   | 0.014    | 0.009     | 0.153 |
| 4xSRNN+Re     | 0.086   | 0.051   | 0.031   | 0.010    | 0.008     | 0.086 |
| L 4xSRNN+Re   | 0.109   | 0.143   | 0.023   | 0.021    | 0.018     | 0.143 |
| RNN+4xAdd+Re  | 0.121   | 0.096   | 0.023   | 0.011    | 0.011     | 0.121 |
| L RNN+4xAdd+Re| 0.336   | 0.165   | 0.024   | 0.016    | 0.013     | 0.336 |
| RNN+4xSRNN+Re | 0.102   | 0.151   | 0.017   | 0.012    | 0.014     | 0.151 |
| L RNN+4xSRNN+Re| 0.211  | 0.288   | 0.024   | 0.018    | 0.016     | 0.288 |
| MAX           | 0.336   | 0.540   | 0.045   | 0.051    | 0.025     |       |
Transformation Autoregressive Networks

Table 6. Held out test log-likelihood for \textit{forest} dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM | RAM | TIED | MultiInd | SingleInd |
|----------------|-----|-----|------|----------|-----------|
| None           | 0.751 | -1.383 | -0.653 | -12.824 | - - - |
| L None         | 1.910 | 1.834 | -0.243 | -7.665 | -11.062 |
| RNN            | 1.395 | 0.053 | 0.221 | -5.130 | -15.983 |
| L RNN          | 2.189$^{(8)}$ | 1.747 | -0.087 | -4.001 | -5.807 |
| 2xRNN          | 1.832 | 1.830 | 0.448 | -6.162 | -9.095 |
| L 2xRNN        | 2.240$^{(6)}$ | 2.432$^{(3)}$ | 0.264 | -3.956 | -5.125 |
| 4xAdd+Re       | 1.106 | 1.430 | 0.420 | -0.021 | -0.492 |
| L 4xAdd+Re     | 2.043 | 1.979 | 0.909 | 0.365 | -0.088 |
| 4xSRNN+Re      | 1.178 | 1.428 | 0.187 | -0.029 | -0.212 |
| L 4xSRNN+Re    | 2.089$^{(9)}$ | 2.061$^{(10)}$ | 0.611 | 0.754 | 0.593 |
| RNN+4xAdd+Re   | 1.962 | 2.226$^{(7)}$ | 0.857 | 0.081 | 0.086 |
| L RNN+4xAdd+Re | 2.389$^{(4)}$ | 2.672$^{(1)}$ | 0.852 | 0.450 | 0.251 |
| RNN+4xSRNN+Re  | 1.599 | 1.545 | 0.510 | 0.182 | 0.369 |
| L RNN+4xSRNN+Re| 2.297$^{(5)}$ | 2.443$^{(2)}$ | 0.804 | 0.600 | 0.480 |

Table 7. Held out test log-likelihood for \textit{pendigits} dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM     | RAM    | TIED   | MultiInd | SingleInd |
|----------------|---------|--------|--------|----------|-----------|
| None           | 6.923$^{(1)}$ | 3.911$^{(8)}$ | 1.437  | -14.138  | - - - |
| L None         | 4.104$^{(9)}$ | 2.911  | -2.872 | -9.997   | -15.617 |
| RNN            | 5.464$^{(3)}$ | 3.273  | -1.676 | -10.144  | -19.719 |
| L RNN          | 4.072$^{(6)}$ | 1.398  | -2.299 | -10.840  | -13.103 |
| 2xRNN          | 6.376$^{(5)}$ | 3.896$^{(7)}$ | -4.002 | -12.132  | -16.576 |
| L 2xRNN        | 2.987   | 0.871  | -3.977 | -10.890  | -12.711 |
| 4xAdd+Re       | -1.924  | -3.087 | -3.172 | -5.010   | -6.498 |
| L 4xAdd+Re     | -1.796  | -1.438 | -2.288 | -4.951   | -7.834 |
| 4xSRNN+Re      | 5.854$^{(2)}$ | 2.146  | -2.827 | -5.970   | -7.084 |
| L 4xSRNN+Re    | 3.758   | -1.020 | -3.370 | -5.885   | -12.978 |
| RNN+4xAdd+Re   | -2.357  | -2.869 | -2.187 | -5.454   | -8.053 |
| L RNN+4xAdd+Re | -2.687  | -2.103 | -2.185 | -4.742   | -6.941 |
| RNN+4xSRNN+Re  | 5.207$^{(4)}$ | 2.425  | -2.126 | -5.147   | -8.859 |
| L RNN+4xSRNN+Re| 3.466$^{(10)}$ | 0.496  | -2.761 | -7.205   | -13.897 |
Table 8. Held out test log-likelihood for susy dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM | RAM | TIED | MultiInd | SingleInd |
|----------------|-----|-----|------|----------|-----------|
| None           | 9.736 | -14.821 | -5.721 | -21.369 | -        |
| L None         | 15.731 | **16.930**(8) | 6.410 | -8.846 | -17.130 |
| RNN            | 12.784 | 3.347 | 6.114 | -18.575 | -44.273 |
| L RNN          | 16.381 | **18.389**(2) | 6.772 | -5.744 | -11.489 |
| 2xRNN          | 11.052 | 14.362 | 3.595 | -16.478 | -33.126 |
| L 2xRNN        | 14.523 | **17.373**(7) | 10.687 | -6.884 | -10.420 |
| 4xAdd+Re       | 9.835 | 8.033 | 7.238 | 6.031 | 4.245 |
| L 4xAdd+Re     | **17.673**(3) | **16.500**(10) | 11.613 | 10.941 | 9.034 |
| 4xSRNN+Re      | 8.798 | 13.235 | 1.234 | 6.936 | 3.378 |
| L 4xSRNN+Re    | 14.242 | **17.870**(5) | 15.397 | 12.161 | 13.413 |
| RNN+4xAdd+Re   | 15.408 | 12.480 | 9.409 | 7.619 | 5.446 |
| L RNN+4xAdd+Re | **17.474**(6) | 16.376 | 13.765 | 10.951 | 8.269 |
| RNN+4xSRNN+Re  | 14.066 | **17.691**(4) | 9.136 | 10.088 | 7.656 |
| L RNN+4xSRNN+Re | **16.627**(9) | **18.941**(1) | 13.469 | 12.105 | 12.349 |

Table 9. Held out test log-likelihood for higgs dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM | RAM | TIED | MultiInd | SingleInd |
|----------------|-----|-----|------|----------|-----------|
| None           | -6.220 | -5.848 | -13.883 | -25.793 | -        |
| L None         | -3.798**(8) | -10.651 | -9.084 | -16.025 | -36.051 |
| RNN            | -5.800 | -2.600**(3) | -10.797 | -25.760 | -66.223 |
| L RNN          | -3.975**(9) | -0.340**(1) | -8.574 | -18.607 | -32.753 |
| 2xRNN          | -6.456 | -4.833 | -9.192 | -25.398 | -60.040 |
| L 2xRNN        | -5.866 | -3.222**(5) | -8.216 | -16.083 | -30.730 |
| 4xAdd+Re       | -6.502 | -10.491 | -9.356 | -13.678 | -15.138 |
| L 4xAdd+Re     | -5.377 | -5.611 | -8.006 | -12.106 | -14.129 |
| 4xSRNN+Re      | -7.422 | -6.863 | -11.033 | -11.878 | -12.182 |
| L 4xSRNN+Re    | -5.999 | -9.329 | -8.474 | -8.223 | -8.926 |
| RNN+4xAdd+Re   | -4.242**(10) | -4.804 | -9.187 | -12.321 | -15.261 |
| L RNN+4xAdd+Re | -3.396**(6) | -3.049**(4) | -8.052 | -12.246 | -13.765 |
| RNN+4xSRNN+Re  | -5.262 | -2.116**(2) | -10.105 | -12.307 | -9.388 |
| L RNN+4xSRNN+Re | -3.756**(7) | -4.773 | -8.097 | -9.378 | -7.721 |
Table 10. Held out test log-likelihood for hepmass dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM  | RAM   | TIED  | MultiInd | SingleInd |
|---------------|------|-------|-------|----------|-----------|
| None          | 2.328| **3.710**<sup>(6)</sup> | -4.948| -19.771 | - - -     |
| L None        | **3.570**<sup>(7)</sup> | 2.517 | -4.052| -9.266   | -35.042   |
| RNN           | 2.088| **4.935**<sup>(1)</sup> | -1.639| -19.851 | -47.686   |
| L RNN         | 2.869<sup>(10)</sup> | **5.047**<sup>(2)</sup> | -2.920| -16.032 | -30.210   |
| 2xRNN         | 1.774| 0.902 | -1.909| -15.440 | -36.754   |
| L 2xRNN       | 2.053| **3.680**<sup>(5)</sup> | -2.150| -15.457 | -24.079   |
| 4xAdd+Re      | 1.678| 1.873 | -4.046| -9.117   | -11.387   |
| L 4xAdd+Re    | 1.961| 2.543 | -2.259| -6.907   | -9.275    |
| 4xSRNN+Re     | 1.443| 2.156 | -2.904| -6.091   | -7.186    |
| L 4xSRNN+Re   | 2.072| 2.730 | -3.014| -5.747   | -6.245    |
| RNN+4xAdd+Re  | 2.817| 0.912 | -2.514| -6.003   | -9.284    |
| L RNN+4xAdd+Re| **3.906**<sup>(3)</sup> | -1.869| -3.847| -6.339   | -9.103    |
| RNN+4xSRNN+Re | 2.663| **3.586**<sup>(8)</sup> | -0.863| -7.146   | -3.939    |
| L RNN+4xSRNN+Re| **3.759**<sup>(4)</sup> | **3.487**<sup>(9)</sup> | -0.239| -7.522   | -6.102    |

Table 11. Held out test log-likelihood for satimage2 dataset. The superscript denotes ranking of log-likelihood on crossvalidation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM  | RAM   | TIED  | MultiInd | SingleInd |
|---------------|------|-------|-------|----------|-----------|
| None          | -1.716<sup>(9)</sup> | -1.257<sup>(3)</sup> | -9.296| -50.507  | - - -     |
| L None        | -20.164| -1.079<sup>(4)</sup> | -2.635| **-1.570**<sup>(5)</sup> | -5.972    |
| RNN           | -7.728| -4.949 | -5.466| -6.047   | -16.521   |
| L RNN         | -31.296| **-0.773**<sup>(2)</sup> | -3.944| **-1.824**<sup>(8)</sup> | -2.977    |
| 2xRNN         | -12.283| **-2.193**<sup>(7)</sup> | -2.137| -5.447   | -8.075    |
| L 2xRNN       | -20.968| -0.550<sup>(1)</sup> | -5.140| **-1.699**<sup>(6)</sup> | **-2.276**<sup>(10)</sup> |
| 4xAdd+Re      | -19.931| -7.539 | -11.826| -18.901  | -17.977   |
| L 4xAdd+Re    | -21.128| -9.944 | -12.336| **-21.677** | -24.070  |
| 4xSRNN+Re     | -7.519| -11.368| -2.549 | -7.730   | -7.232    |
| L 4xSRNN+Re   | -18.170| -7.709 | -5.533 | -17.085  | -15.347   |
| RNN+4xAdd+Re  | -19.278| -11.789| -12.837| -21.249  | -22.786   |
| L RNN+4xAdd+Re| -20.899| -12.949| -12.867| -26.164  | -28.302   |
| RNN+4xSRNN+Re | -13.476| -3.951 | -6.284 | -15.025  | -16.443   |
| L RNN+4xSRNN+Re| -20.179| -12.128| -7.258 | -18.065  | -18.125   |
Table 12. Held out test log-likelihood for music dataset. The superscript denotes ranking of log-likelihood on cross-validation dataset. Note that NADE is TIED conditional with None Transform and NICE is Add+Re Transformation with SingleInd Conditional. In parenthesis is the top-10 picks using validation set.

| Transformation | LAM   | RAM   | TIED  | MultiInd | SingleInd |
|---------------|-------|-------|-------|----------|-----------|
| None          | -57.873 | -97.925 | -98.047 | -113.099 | - - -     |
| L RNN None    | -52.954(4) | -74.220 | -72.441 | -82.866 | -104.287 |
| RNN           | -54.933(10) | -80.436 | -74.361 | -106.219 | -144.735 |
| L RNN 2xRNN   | -52.710(3) | -59.815 | -66.536 | -82.731 | -98.813  |
| 2xRNN         | -56.958 | -85.359 | -77.456 | -104.440 | -133.898 |
| L 2xRNN+Re    | -53.956(8) | -57.611 | -65.016 | -82.678 | -96.542  |
| 4xAdd+Re      | -56.349 | -69.302 | -67.064 | -73.886 | -83.524  |
| L 4xAdd+Re    | -53.169(5) | -59.282 | -59.093 | -69.887 | -79.330  |
| 4xSRNN+Re     | -53.879(7) | -55.665 | -63.894 | -77.564 | -81.188  |
| RNN+4xAdd+Re  | -53.177(6) | -67.377 | -63.372 | -73.882 | -84.032  |
| L RNN+4xAdd+Re| -51.572(1) | -56.190 | -58.885 | -69.484 | -79.555  |
| RNN+4xSRNN+Re | -54.065(9) | -61.204 | -76.437 | -71.814 | -81.087  |
| L RNN+4xSRNN+Re | -52.617(2) | -68.756 | -65.061 | -83.292 | -78.997  |

Table 13. Held out test log-likelihood for wordvecs dataset. The superscript denotes ranking of log-likelihood on validation dataset. Due to time constraints only models with linear transformations were trained.

| Transformation | LAM   | RAM   | TIED  | MultiInd | SingleInd |
|---------------|-------|-------|-------|----------|-----------|
| L None        | -252.659(6) | -279.788 | -278.789 | -332.474 | -387.341  |
| L RNN         | -252.894(7) | -278.795 | -278.663 | -332.689 | -386.700  |
| L 2xRNN       | -250.285(4) | -275.508 | -277.848 | -333.234 | -386.649  |
| L 4xAdd+Re    | -247.440(1) | -272.371(8) | -274.205 | -331.148 | -374.563  |
| L 4xSRNN+Re   | -248.393(2) | -300.666 | -273.372(9) | -308.735 | 0.000     |
| L RNN+4xAdd+Re| -249.980(3) | -280.938 | -273.976(10) | -331.316 | -380.031 |
| L RNN+4xSRNN+Re | -251.468(5) | -280.325 | -274.082 | -308.148 | -395.084 |