TraDE: Transformers for Density Estimation

Rasool Fakoor [1] **Pratik Chaudhari** [2] **Jonas Mueller** [1] **Alexander J. Smola** [1]

Abstract

We present TraDE, an attention-based architecture for auto-regressive density estimation. Our model is trained against both the standard maximum likelihood objective as well as a Maximum Mean Discrepancy loss to ensure that samples from the estimate resemble the training data distribution. The use of self-attention means that the model need not retain conditional sufficient statistics during the auto-regressive process beyond what is needed for each covariate. TraDE performs significantly better than existing approaches such as flow based estimators on standard tabular and image-based benchmarks in terms of the log-likelihood on held-out data. Furthermore, we present a suite of tasks such as regression using generated samples, out-of-distribution detection, and robustness to outliers in the training data and demonstrate that TraDE works well in these scenarios. Long version of paper is available here https://arxiv.org/abs/2004.02441.

1. Introduction

Density estimation involves estimating a probability density $p(x)$, given independent, identically distributed (iid) samples from it. This is a versatile and important problem as it allows one to generate synthetic data or perform novelty and outlier detection. It is also an important subroutine in applications of graphical models. Deep neural networks are a powerful function class and learning complex distributions with them is promising. This has resulted in a resurgence of interest in the classical problem of density estimation.

One of the more popular techniques for density estimation is to sample data from a simple reference distribution and then to learn a (sequence of) invertible transformations that allow us to adapt it to a target distribution. Flow-based methods (Durkan et al., 2019b) employ this with great success. A more classical approach is to decompose $p(x)$ in an iterative manner via conditional probabilities $p(x_{i+1}|x_{1\ldots i})$ and fit this distribution using the data (Murphy, 2013). One may even employ implicit generative models to sample from $p(x)$ directly, perhaps without the ability to compute the likelihood. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) that reign supreme for image synthesis (Karras et al., 2017) belong to this class.

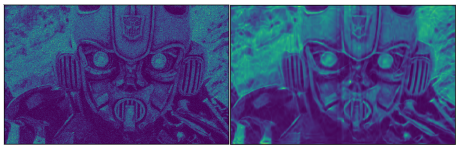


Figure 1. TraDE is well suited to density estimation of Transformers. Left: Bumblebee, Right: estimate.

Implementing above methods however requires special care, e.g., the normalizing transform requires the network to be invertible with an efficiently computable Jacobian. Auto-regressive models using recurrent networks are difficult to scale to high-dimensional data due to the need to store a potentially high-dimensional conditional sufficient statistic (and also due to vanishing gradients). Generative models can be difficult to train and GANs lack a closed density model. Much of the current work is devoted to mitigating these issues.

Contributions. We focus on auto-regressive models:

- We introduce TraDE, a simple but novel auto-regressive density estimator that uses self-attention to approximate arbitrary continuous and discrete conditional densities. An additional innovation that we propose is to use a recurrent neural network (RNN)-based input embedding for the self-attention network. Our model is simpler and more flexible than contemporary architectures such as (Durkan et al., 2019b;a; Kingma et al., 2016; De Cao et al., 2019) for this problem.
- The maximum likelihood objective in current density estimation methods does not directly impose constraints on sample fidelity which affects performance on downstream tasks such as classification using the sampled data. To fix this, we introduce a Maximum Mean Discrepancy (MMD)-based regularizer in the training objective of TraDE.
- Log-likelihood on held-out data is the prevalent metric

[1]Amazon Web Services [2]University of Pennsylvania. Correspondence to: Rasool Fakoor <fakoor@amazon.com.>.

2nd workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020)

to evaluate density estimators. However, this only provides a partial view of the performance in real-world applications. We propose a suite of experiments to systematically evaluate the performance of density estimators in downstream tasks such as classification and regression using generated samples, detection of out-of-distribution samples, and robustness to outliers in the training data.

- We provide extensive empirical evidence of strong performance of TraDE on standard benchmarks along with thorough ablation experiments.

2. Background and Related Work

Given a dataset $\{x^1, \ldots, x^n\}$ where each sample $x^k \in \mathbb{R}^d$ is drawn iid from some probability distribution $p(x)$, the maximum-likelihood formulation of density estimation finds a θ-parameterized distribution q_θ such that

$$\widehat{\theta} = \operatorname*{argmax}_\theta \frac{1}{n} \sum_{i=1}^{n} \log q(x^i; \theta). \tag{1}$$

The candidate distribution q can be parameterized in a variety of ways as we discuss next.

Normalizing flows write $x \sim q$ as a transformation of samples z from some base distribution p_z from which one can draw samples easily (Papamakarios et al., 2019). If this mapping is $f_\theta : z \to x$, two distributions can be related using the determinant of the Jacobian as $q(x; \theta) := p_z(z) \left| \frac{\mathrm{d}f_\theta}{\mathrm{d}z} \right|^{-1}$. A key property of flow-based models is that f_θ is a diffeomorphism, i.e., it is invertible and both f_θ and f_θ^{-1} are differentiable. This allows gradient-based minimization of (1) which involves a term of the form $\log |\mathrm{d}f_\theta / \mathrm{d}z|$. Good performance using normalizing flows requires that the mapping f_θ be powerful yet invertible with a Jacobian that can be computed efficiently. There are a number of techniques in the literature to achieve this, e.g., linear mappings, planar/radial flows (Rezende & Mohamed, 2015; Tabak & Turner, 2013), Sylvester flows (Berg et al., 2018), coupling (Dinh et al., 2014) and auto-regressive models (Larochelle & Murray, 2011). One may also compose the transformations, e.g., using monotonic mappings f_θ in each layer (Huang et al., 2018; De Cao et al., 2019).

Auto-regressive models have their roots in probabilistic graphical models (Koller & Friedman, 2009). These models factorize the distribution q_θ as a product of univariate conditional distributions $q(x; \theta) := \prod_i q_i(x_i | x_1, \ldots x_{i-1}; \theta)$. Note that in this case the Jacobian is a lower-triangular matrix with entries $\mathrm{d}x_i / \mathrm{d}x_j$ and the determinant is simply a product of the entries along the diagonal. Parameters of the conditionals in the product may be shared using RNNs (Oliva et al., 2018; Kingma et al., 2016).

For high-dimensional data, the challenge lies in handling the increasingly large state space $x_1, \ldots, x_{i-1}$ required to sample x_i. In a latent-variable autoregressive model past data is stored in some representation h_i which is updated via a function $h_{i+1} = g(h_i, x_i)$. This overcomes the problem of very high dimensional estimation, albeit at the expense of loss in fidelity. Techniques like masking the computational paths in a feed-forward network are popular to alleviate these problems further (Uria et al., 2016; Germain et al., 2015; Papamakarios et al., 2017). Choosing a good variable ordering for the factorization of q is paramount in auto-regressive models; several algorithms train ensembles over multiple orderings for good performance. While autoregressive models are commonly applied to natural language and time series data, this setting only involves variables that are already naturally ordered (Chelba et al., 2013). In contrast, we consider continuous (and discrete) density estimation of vector valued data. e.g. tabular data, where the underlying ordering and dependencies between variables is often unknown.

Generative models focus on drawing samples from the estimated distribution that look resemble the true distribution of data. There is a rich history of learning explicit models from variational inference (Jordan et al., 1999) that allow both drawing samples and estimating the log-likelihood or implicit models such as Generative Adversarial Networks (GANs, see (Goodfellow et al., 2014)) where one may only draw samples. These have been shown to work well for natural images (Kingma & Welling, 2013) but have not obtained similar performance for tabular data.

3. The Architecture of TraDE

TraDE is an auto-regressive density estimator that factorizes distribution q as

$$q(x; \theta) := \prod_{i=1}^{n} q_i(x_i | x_1, \ldots, x_{i-1}; \theta); \tag{2}$$

Here the i^{th} univariate conditional q_i conditions the feature x_i upon all the features preceding it, and may be easier to model than a high-dimensional joint distribution. The parameters θ of our model are shared amongst the conditionals. Our main observation is that the auto-regressive nature of the conditionals can be accurately modeled using the attention mechanism in a Transformer architecture (Vaswani et al., 2017).

Self-attention. The Transformer is a neural sequence transduction model and consists of a multi-layer encoder/decoder pair. We only need the encoder for building TraDE. The encoder takes the input sequence $(x_1, \ldots, x_d)$ and predicts the i^{th}-conditional $q_i(x_i | x_1, \ldots, x_{i-1}; \theta)$.

Each conditional is parametrized as a mixture of multi-variate Gaussians, each with mean that depends on $x_1, \ldots, x_{i-1}$ and a diagonal covariance:

$$q_i(x_i|x_1, \ldots, x_{i-1}; \theta) = \sum_{k=1}^{m} p_k \; N(x_i; \mu_k, \sigma_k^2 I). \quad (3)$$

with the mixture probabilities $\sum_{k=1}^{m} p_k = 1$. All the three quantities p_k, μ_k and σ_k are predicted by the model as outputs. They are parametrized by θ and depend on $x_1, \ldots, x_{i-1}$. The crucial property of the Transformer's encoder is the self-attention module outputs a representation that captures correlations in different parts of its input. In a nutshell, the self-attention map outputs $z_i = \sum_{j=1}^{d} \alpha_j \varphi(x_j)$ where $\varphi(x_j)$ is an embedding of the feature x_j and normalized weights $\alpha_j = \langle \varphi(x_i), \varphi(x_j) \rangle$ compute the similarity between the embedding of x_i and that of x_j. Self-attention therefore amounts to a linear combination of the embedding of each feature with features that are more similar to x_i getting a larger weight. We also use multi-headed attention like (Vaswani et al., 2017) which computes self-attention independently for different embeddings. Self-attention is the crucial property that allows TraDE to handle long-range and complex correlations in the input features; effectively this eliminates the vanishing gradient problem in RNNs by allowing direct connections between far away input neurons (Vaswani et al., 2017). Self-attention also enables permutation equivariance and naturally enables TraDE to be agnostic to the ordering of the features.

Masking is used to prevent $x_i, x_{i+1}, \ldots, x_d$ from taking part in the computation of the output q_i and thereby preserve the auto-regressive property of our density estimator. We keep residual connections, layer normalization and dropout in the encoder unchanged from the original architecture of (Vaswani et al., 2017). The final layer of the encoder, and our model, is a fully-connected layer where the i^{th} predicts the mixture probabilities p_k, the means μ_k and the corresponding standard deviations σ_k^2.

Positional encoding involves encoding the position k of feature x_k and is an integral part of the Transformer; the original authors append Fourier position features to each input. Picking hyper-parameters for the frequencies is however difficult and it does not work well either for density estimation (see Section 4). An alternative that we propose is to use a simple recurrent network at the input to embed the input values at each position. Here the time-steps of the RNN implicitly encode the positional information, and we use a Gated Recurrent Unit (GRU) model to better handle long-range dependencies (Cho et al., 2014). This parallels recent findings from language modeling where (Wang et al., 2019) also used an initial RNN embedding to generate inputs to the transformer. Observe that the GRU does not slow down TraDE at inference time since sampling is performed in auto-regressive fashion anyway and remains $\mathcal{O}(d)$. The complexity of training is marginally higher but this is more than made up for by the superior performance.

Remark 1 (Regularization in density estimation). The maximum-likelihood objective in (1) does not have a regularization term that would help with outliers. There exists a large number of classical techniques, such as maximum entropy and approximate moment matching techniques (Phillips et al., 2004; Altun & Smola, 2006) that can be used. They map to some extent to the parameter based capacity control in deep learning, such as Dropout or input permutation (or they're implicitly determined by the choice of architecture). Instead, we use MMD to penalize differences between training data and samples from the model directly. This allows us to use powerful models like the Transformer with less risk of overfitting.

Remark 2 (Comparison to other architectures). Our architecture for TraDE can be summarized as using the encoder of a Transformer with appropriate masking to achieve auto-regressive dependencies with an output layer consisting of a mixture of multi-variate Gaussians and an input embedding layer built using an RNN. It is more flexible than architectures for normalizing flows without restrictive constraints on the input-output map (e.g. invertible functions, Jacobian computational costs, etc.) As compared to other auto-regressive models, TraDE can handle long-range dependencies and does not need to permute input features during training/inference.

Unlike discrete language data, tabular datasets contain many numerical values and their categorical features do not share a common vocabulary. Thus existing applications of Transformers to such data remain limited. Beyond its insensitivity to feature order, Transformers are a good fit for tabular data because their lower layers only model lower-order feature interactions (starting with pairwise interactions in the first layer and building up in each additional layer). This relates them to ANOVA (Wahba et al., 1995) which only progressively blend features unlike in a fully-connected network.

Finally, the objective/architecture of TraDE is general enough to handle both continuous and discrete data, unlike many existing density estimators. As experiments in Section 4 shows these properties make TraDE very well-suited for auto-regressive density estimation.

3.1. Regularization using Maximum Mean Discrepancy (MMD)

An alternative to maximum likelihood estimation, and in some cases a dual to it, is to perform non-parametric moment matching (Altun & Smola, 2006). One can combine a log-likelihood loss and a two-sample discrepancy loss to

ensure high fidelity, i.e., the samples resemble those from the original dataset.

We can test whether two distributions p and q supported on a space $\mathcal{X}$ are different using samples drawn from each of them by finding a smooth function that is large on samples drawn from p and small on samples drawn from q. If $x \sim p$ and $y \sim q$, then $p = q$ if and only if $\mathbb{E}_x[f(x)] = \mathbb{E}_y[f(y)]$ for all bounded continuous functions f on $\mathcal{X}$ (Lemma 9.3.2 in (Dudley, 2018)). We can exploit this result computationally by restricting the test functions to some class $f \in \mathcal{F}$ and finding the worst test function. This leads to the Maximum Mean Discrepancy (MMD) metric defined next (Fortet & Mourier, 1953; Müller, 1997; Gretton et al., 2012; Sriperumbudur et al., 2016). For a class $\mathcal{F}$ of functions $f : \mathcal{X} \to \mathbb{R}$, the MMD between distributions p, q is

$$\text{MMD}[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)] \right). \quad (4)$$

It is cumbersome to find the supremum over a general class of functions $\mathcal{F}$ to compute the MMD. We can however restrict $\mathcal{F}$ to be the unit ball in a universal Reproducing Kernel Hilbert Space (RKHS) (Gretton et al., 2012)) with kernel k. The MMD is a metric in this case and is given by

$$\begin{aligned} &\text{MMD}^2[k, p, q] \quad (5) \\ &= \mathop{\mathbb{E}}_{x,x' \sim p}[k(x, x')] - 2 \mathop{\mathbb{E}}_{x \sim p, y \sim q}[k(x, y)] + \mathop{\mathbb{E}}_{y,y' \sim p}[k(y, y')] \end{aligned}$$

With a universal kernel (e.g. Gaussian, Laplace), MMD will capture *any* difference between distributions (Steinwart, 2001). We can easily obtain an empirical estimate of the MMD above using samples (Gretton et al., 2012).

3.2. The Loss Function of TraDE

The MLE objective in (1) only forces our network to consider how it is modeling each conditional, rather than how it is modeling/sampling the full joint distribution. To encourage both, we combine the MLE objective with a regularization term based on MMD in (5) to get the loss function of TraDE. The former ensures consistency of the estimate while the MMD term is effective in detecting obvious discrepancies when the samples drawn from the model do not resemble samples in the training dataset. In theory, MMD with a universal kernel would also produce *consistent* estimates, but maximizing likelihood ensures our estimate is statistically *efficient*. In practice a combination of both objectives yields superior results, and makes performance less dependent on the MMD kernel bandwidth. The objective minimized in TraDE is

$$-\frac{1}{nd} \sum_{i=1}^{n} \sum_{j=1}^{d} \log q_j(x_j^i | x_1^i, \ldots, x_{j-1}^i; \theta) + \lambda \, \text{MMD}^2[k, p, q(\theta)], \quad (6)$$

where hyper-parameter $\lambda \geq 0$ controls the degree of regularization. We have divided the first term by the dimensionality d because the MMD term does not scale with d. This objective is minimized using mini-batch gradient-based updates.

The gradient of the log-likelihood term can be computed using standard back-propagation. Computing the gradient of the MMD term involves differentiating the samples from q_θ with respect to the parameters θ. For continuous-valued data, this is easily done using the reparametrization trick (Kingma & Welling, 2013) since we model each conditional as a mixture of Gaussians. For categorical features, we calculate the gradient using the Gumbel softmax trick (Maddison et al., 2016). The objective/architecture of TraDE is thus general enough to handle both continuous and discrete data, unlike many existing density estimators.

4. Experiments

We first evaluate TraDE both qualitatively (Fig. 2 and Fig. 3 in the Supplementary Material) and quantitatively on standard benchmark datasets (Section 4.1). We then present additional ways to evaluate the performance of density estimators in downstream tasks (Appendix A.1) along with some ablation studies (Appendix A.2). Details for all the experiments in this section, including hyper-parameters, are provided in the Supplementary Material.

Figure 2. **Qualitative evaluation on 2-dimensional datasets.** We train TraDE on samples from six 2-dimensional densities and evaluate the model likelihood over the entire domain by sampling a fine grid; original densities are shown in rows 1 and 3 and estimated densities are shown in rows 2 and 4. This setup is similar to (Nash & Durkan, 2019). These distributions are highly multimodal with complex correlations. Nevertheless TraDE learns an accurate estimate of the true density across a large number of examples.

Table 1. **Average test log-likelihood in nats (higher is better) for benchmark datasets.** Entries marked with * evaluate standard deviation across 3 independent runs of the algorithm; all others are mean ± standard error. TraDE achieves significantly better log-likelihood than other algorithms on all datasets except MINIBOONE. This is in spite of the fact that some methods, e.g., MAF train an ensemble.

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
REAL NVP	0.17 ± 0.01	8.33 ± 0.14	-18.71 ± 0.02	-13.84 ± 0.52	153.28 ± 1.78
MADE MOG	0.4 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
MAF MOG	0.3 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
FFJORD	0.46	8.59	-14.92	-10.43	157.4
NAF	$0.62 \pm 0.01^*$	$11.96 \pm 0.33^*$	$-15.09 \pm 0.4^*$	$\mathbf{-8.86 \pm 0.15}^*$	$157.43 \pm 0.3^*$
TAN	0.6 ± 0.01	12.06 ± 0.02	-13.78 ± 0.02	-11.01 ± 0.48	159.8 ± 0.07
BNAF	$0.61 \pm 0.01^*$	$12.06 \pm 0.09^*$	$-14.71 \pm 0.38^*$	$-8.95 \pm 0.07^*$	$157.36 \pm 0.03^*$
NSF	$0.66 \pm 0.01^*$	$13.09 \pm 0.02^*$	$-14.01 \pm 0.03^*$	$-9.22 \pm 0.48^*$	$157.31 \pm 0.28^*$
AEM	0.70 ± 0.01	13.03 ± 0.01	-12.85 ± 0.01	-10.17 ± 0.26	158.71 ± 0.14
TRADE (OURS)	$\mathbf{0.73 \pm 0.00}^*$	$\mathbf{13.27 \pm 0.01}^*$	$\mathbf{-12.01 \pm 0.03}^*$	$-9.49 \pm 0.13^*$	$\mathbf{160.01 \pm 0.02}^*$

4.1. Results on Benchmark Datasets

We follow the experimental setup of (Papamakarios et al., 2017) to ensure the same training/validation/test dataset splits in our evaluation. In particular, the preprocessing of all the datasets is kept the same as that of (Papamakarios et al., 2017). The MNIST dataset (LeCun et al., 1990) is used to evaluate TraDE on high-dimensional image-based data. We follow the variational inference literature, e.g., (Oord et al., 2016), and use the binarized version of MNIST. The datasets for anomaly detection tasks are from the Outlier Detection DataSets (OODS) library (Rayana, 2016). We normalized the OODS data by subtracting the per-feature mean and dividing by the standard deviation. We show the results on benchmark datasets in Table 1. There is a wide diversity in the algorithms for density estimation but we make an effort to provide a complete comparison of known results irrespective of the specific methodology. Some methods like Neural Spline Flows (NSF) by (Durkan et al., 2019b) are quite complex to implement; others like Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017) use ensembles to estimate the density; some others like Autoregressive Energy Machines (AEM) of (Nash & Durkan, 2019) average the log-likelihood over a large number of importance samples. As the table shows, TraDE obtains performance improvements over all these methods in terms of the log-likelihood. This performance is persistent across all datasets except MINIBOONE where TraDE is competitive although not the best. The improvement is drastic for the POWER, HEPMASS and BSDS300.

We also evaluate TraDE on the MNIST dataset in terms of the log-likelihood on test data. As Table 2 shows TraDE obtains high log-likelihood even compared to sophisticated models such as Pixel-RNN (Oord et al., 2016). This is a difficult dataset for density estimation because of the high dimensionality. We also show the quality of the samples generated by the model in Fig. 3.

Table 2. **Negative average test log-likelihood in nats (smaller is better) on MNIST.**

	LOG-LIKELIHOOD
VAE	82.14 ± 0.07
PLANAR FLOWS	81.91 ± 0.22
SYLVESTER	80.22 ± 0.03
BLOCK NAF	80.71 ± 0.09
PIXELRNN	79.20
TRADE (OURS)	$\mathbf{78.92 \pm 0.00}$

5. Discussion

This paper demonstrates that self-attention is naturally suited to building auto-regressive models with strong performance in density estimation tasks. Our proposed method is more flexible than architectures for normalizing flows and can handle long-range dependencies as compared to other auto-regressive models. We contribute a suite of downstream tasks such as regression, out of distribution detection, and robustness to noisy data, which evaluate how useful the density estimates are in real-world applications. Our other contribution is a new regularized likelihood objective for density estimation with an MMD-penalty to ensure models produce high-fidelity samples during training. This is reminiscent of maximum entropy modeling which seeks a distribution matching moments between empirical averages and the expectations generated by the model. In the parlance of maximum entropy, we are effectively combining dual constraints (via log-likelihood) and primal ones (via MMD) to ensure a good fit. Additional objectives are appropriate since the models we use are not optimal within their function class and it is advantageous to enforce desirable constraints directly.

References

Altun, Y. and Smola, A. Unifying divergence minimization and statistical inference via convex duality. In *International Conference on Computational Learning Theory*, pp. 139–153. Springer, 2006.

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. One billion word benchmark for measuring progress in statistical language modeling. *arXiv:1312.3005*, 2013.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

De Cao, N., Titov, I., and Aziz, W. Block neural autoregressive flow. *arXiv preprint arXiv:1904.04676*, 2019.

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

Dudley, R. M. *Real analysis and probability*. Chapman and Hall/CRC, 2018.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Cubic-spline flows. *arXiv preprint arXiv:1906.02145*, 2019a.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. In *Advances in Neural Information Processing Systems*, pp. 7509–7520, 2019b.

Fortet, R. and Mourier, E. Convergence de la répartition empirique vers la répartition théorique. In *Annales scientifiques de l'École Normale Supérieure*, volume 70, pp. 267–285, 1953.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

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.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pp. 396–404, 1990.

Lopez-Paz, D. and Oquab, M. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Müller, A. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

Murphy, K. P. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

Nash, C. and Durkan, C. Autoregressive energy machines. *arXiv preprint arXiv:1904.05626*, 2019.

Oliva, J. B., Dubey, A., Zaheer, M., Poczos, B., Salakhutdinov, R., Xing, E. P., and Schneider, J. Transformation autoregressive networks. *arXiv preprint arXiv:1801.09819*, 2018.

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. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Phillips, S. J., Dudík, M., and Schapire, R. E. A maximum entropy approach to species distribution modeling. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 83, 2004.

Rayana, S. ODDS library, 2016. http://odds.cs.stonybrook.edu.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Sriperumbudur, B. et al. On the optimal estimation of probability measures in weak and strong topologies. *Bernoulli*, 22(3):1839–1893, 2016.

Steinwart, I. On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93, 2001.

Tabak, E. G. and Turner, C. V. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

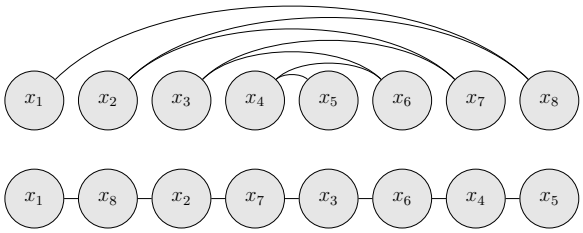
Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wahba, G., Wang, Y., Gu, C., Klein, R., Klein, B., et al. Smoothing spline anova for exponential families, with application to the wisconsin epidemiological study of diabetic retinopathy: the 1994 neyman memorial lecture. *The Annals of Statistics*, 23(6): 1865–1895, 1995.

Wang, C., Li, M., and Smola, A. J. Language models with transformers. *arXiv preprint arXiv:1904.09408*, 2019.

A. Tools of the TraDE



Consider the 8-dimensional Markov Random Field shown here, where the underlying graphical model is unknown in practice. Consider the following two orders in which to factorize the autoregressive model: $(1, 2, 3, 4, 5, 6, 7, 8)$ and $(1, 8, 2, 7, 3, 6, 4, 5)$. In the latter case the model becomes a simple sequence where e.g. $p(x_3|x_{1,8,2,7}) = p(x_3|x_7)$ due to conditional independence. A latent variable auto-regressive model only needs to preserve the most recently encountered state in this latter ordering. In the first ordering, $p(x_3|x_{1,2})$ can be be simplified further to $p(x_3|x_2)$, but we still need to carry the precise value of x_1 along until the end since $p(x_8|x_{1\ldots7}) = p(x_8|x_{1,2})$. This is a fundamental weakness in models employing RNNs such as (Oliva et al., 2018). In practice, we may be unable to select a favorable ordering for columns in a table (unlike for language where words are inherently ordered), especially as the underlying distribution is unknown.

The above problem is also seen in sequence modeling, and Transformers (Vaswani et al., 2017) were introduced to better model such long-range dependencies through self-attention. The utility of self-attention is its effectiveness at maintaining an accurate representation of $x_{<j}$ while predicting x_d, irrespective of the distance between them. A recurrent network can, in principle, absorb this information into its hidden state. In fact, Long-Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) units were engineered specifically to store long-range dependencies until needed. Nonetheless, storing information costs parameter space. For auto-regressive factorization where the true conditionals require one to store many variables' values for many time steps, the RNN/LSTM hidden state must be undesirably large. The following simple lemma formalizes this.

Lemma 3. *Denote by G the graph of an undirected graphical model over random variables $x_1, \ldots, x_d$. Depending on the order vertices are traversed in our factorization the largest number of latent variables a recurrent auto-regressive model needs to store is bounded from above and below by the minimum and the maximum number of variables with a cut edge of the graph G.*

The proof is as follows. Given a subset of known variables $S \subseteq \{1, \ldots d\}$ we want to estimate the conditional distribution of the variables on the complement $C := \{1, \ldots d\} \backslash S$. For this we need to decompose S into the Markov blanket M of C and its remainder. By definition M consists of the variables with a cut edge. Since $p(x_C|x_S) = p(x_C|x_M)$ we are done.

This problem with long-dependencies in auto-regressive models has been noted before. For instance, recent auto-regressive models employ masking to eliminate the sequential operations of recurrent models (Papamakarios et al., 2017). There are also models like Pixel RNN (Oord et al., 2016) which explicitly design a multi-scale masking mechanism suited for natural images. Note that while there is a natural ordering of random variables in text/image data, variables in tabular data do not follow any canonical ordering. An alternative to alleviating this state space problem is to use attention to attend only to parts of the data relevant for the conditioning.

A.1. Systematic Evaluation of Density Estimators

We propose evaluating density estimation in four canonical ways, regression using the generated samples, a two-sample test to check the quality of generated samples, out-of-distribution detection, and robustness of the density estimator to outliers in the training data. This section shows that TraDE performs well on these tasks which demonstrates that it not only obtains high log-likelihood on held-out data but can also be readily used for downstream tasks.

1. Regression using generated samples. First, we create a regression task where a random variable, say x_d is regressed using data from the others $x_{-d} = (x_1, \ldots, x_{d-1})$. The procedure is as follows: we use the training set of the HEPMASS dataset ($d = 21$) to fit the density estimator; create a synthetic dataset with both inputs x_{-d} and targets x_d sampled from the model. Two boosted decision forest-based regressors are fitted, one on the real data and another on this synthetic data. Both these regressors are tested on *real* test data from the HEPMASS dataset. If the model synthesizes good samples, one would expect that the test performance of the regressor fitted on synthetic data would be comparable to the regressor fitted on real data.

Table 3 shows the results of this experiment. Observe that the classifier trained on data synthetized by TraDE performs very similarly to the one trained on the original data. The MSE of a RNN-based auto-regressive density estimator, which is higher, is provided for comparison.

Table 3. **Mean squared error of regression on HEPMASS.**

Real data	Synthetic data (TraDE)	Synthetic data (RNN)
0.773	0.780	0.803

2. Two-sample test on the generated data. Next we train a boosted decision-forest-based classifier to differentiate between real data and synthesized data. The idea is similar to a two-sample test (Lopez-Paz & Oquab, 2016) in the discriminator of a GAN: if the samples generated by the auto-regressive model are good, the discriminator should have an accuracy of 50%. As Table 4 shows the samples generated by TraDE is much closer to the real data than those generated by the RNN model.

Table 4. **Accuracy of the discriminator trained to classify real data from synthesized data on HEPMASS.** These numbers are the average accuracy of multiple experiments, each of which uses a different subset of columns $x_1, (x_1, x_2), \ldots, (x_1, \ldots, x_d)$ as featuers for the discriminator.

Synthetic data (TraDE)	Synthetic data (RNN)
51 ± 1 %	55 ± 4 %

3. Out-of-distribution detection. This is a classical application of density estimation techniques where we seek to discover unlikely samples in a given dataset. We follow the setup of Oliva et al. (2018): we call a sample out-of-distribution if the likelihood of the sample under the model $q_\theta(x) \leq t$ for a chosen threshold $t \geq 0$. We compute the average precision of detecting out-of-distribution samples by sweeping across different values of t. The resuls are shown in Table 5. Observe that TraDE obtains extremely good performance, of more than 0.95 average precision, on the three datasets.

Table 5. **Average precision for out-of-distribution detection** The numbers for NADE, NICE and TAN were (precisely) eyeballed from the plots of Oliva et al. (2018).

	NADE	NICE	TAN	TraDE
Pendigits	0.91	0.92	0.97	0.98
ForestCover	0.87	0.80	0.94	0.95
Satimage-2	0.98	0.975	0.98	1.0

4. MMD regularization builds robustness to outliers in the data. Real data may contain a large number of outliers and in order to be useful on downstream tasks, density estimation must be insensitive to such outliers. The maximum-likelihood objective is sensitive to outliers in the training data. Methods such as NSF (Durkan et al., 2019b) or MAF (Papamakarios et al., 2017) indirectly mitigate this sensitivity using permutations of the input data or masking within hidden layers but these operations are not designed to be robust to noisy data. We study how TraDE deals with this scenario. In other words, this experiment directly demonstrates the effect of MMD regularization. We add noise to 10% of the entries in the training data; we then fit both TraDE and NSF on this noisy data; both models are evaluated on clean test data. As Table 6 shows, the degradation of both TraDE and NSF is about the same; the former obtains a higher log-likelihood as noted in Table 1.

Table 6. **Average test log-likelihood in nats for HEPMASS dataset with and without additive noise in the training data.**

	Clean Data	Noisy Data
NSF	-14.51	-14.98
TraDE	-11.98	-12.43

A.2. Ablation Experiments

To understand the effect of the design decisions in TraDE we disable (or replace) them one at a time. In particular, we aim to understand the effect of recurrent networks for auto-regressive models, using only multi-headed self-attention in the Transformer without the position encoding, the TraDE model without the MMD regularizer which uses a GRU for embedding the input, and the full TraDE algorithm.

Table 7. **Average test log-likelihood in nats (higher is better) on benchmark datasets** for four algorithms: an RNN for standard auto-regressive density estimation, a Transformer with multi-head attention without positional encoding, TraDE with $\lambda = 0$ in (6) and the full TraDE algorithm.

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
RNN	0.51	6.26	-15.87	-13.13	157.29
Transformer	0.71	12.95	-15.80	-22.29	134.71
TraDE w/o MMD	0.72	13.26	-12.22	**-9.44**	159.97
TraDE	**0.73**	**13.27**	**-12.01**	-9.49	**160.01**

As Table 7 shows, the performance of an RNN as an auto-regressive model is quite poor for all datasets. Using a Transformer network (without position encoding) improves this log-likelihood by a lot but this does not work for all datasets. The biggest improvement is obtained upon adding a GRU-based embedding to the Transformer. The MMD loss improves the log-likelihood by a large amount for HEPMASS and a small amount for POWER, the effect for other datasets is marginal. The other algorithms in this table use the same hyper-parameters (architecture and training) as those of TraDE.

Table 8 compares the performance of the GRU-based input encoding against a Transformer with position encoding. Compare the first row of this table with the second row of Table 7: the performance of the Transformer with position encoding is significantly better than the one without it. This suggests that incorporating the information about the position is critical for auto-regressive models. A recurrent network to incorporate the position information obtains significant performance boost as seen in the second row

of Table 8.

Table 8. **Log-likelihood in nats of using position encoding versus a recurrent network for input embedding.**

	HEPMASS	MINIBOONE	BSDS300
Transformer w/ position encoding	-13.89	-12.28	147.94
TraDE w/o MMD	**-12.22**	**-9.44**	**159.97**

B. Sampling MNIST images

To further investigate the quality samples generated by our model, we show more samples in Fig. 3.

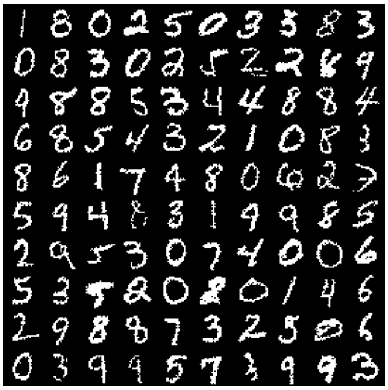


Figure 3. **Samples from TraDE fitted on binary MNIST.**

C. Hyper-parameters for benchmark datasets

All models are trained for 1000 epochs with the Adam optimizer. The MMD kernel is a mixture of up to 5 Gaussians for all datasets, i.e., $k(x, y) = \sum_{i=1}^{5} k_i(x, y)$ where each $k_i(x, y) = e^{-\|x-y\|_2^2/\sigma_i^2}$ of bandwidths $\sigma_i \in \{1, 2, 4, 8, 16\}$. Table 9 shows our model's hyper-parameters.

Table 9. **Hyper-parameters for benchmark datasets.**

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST
MMD coefficient λ	0.2	0.1	0.1	0.4	1.2	0.1
Gaussian mixture components	150	100	100	20	100	1
Number of layers	5	8	6	8	5	6
Multi-head attention head	8	16	8	8	2	4
Gradient clipping norm	5	5	5	5	5	5
Hidden neurons	512	400	128	64	128	256
Dropout	0.1	0.1	0.1	0.2	0.3	0.1
Learning rate	3E-4	3E-4	5E-4	5E-4	5E-4	5E-4
Mini-batch size	512	512	512	64	512	16
Weight decay	1E-6	1E-6	1E-6	0	1E-6	1E-6
Gumbel softmax temperature	n/a	n/a	n/a	n/a	n/a	1.5