

Autoregressive Energy Machines

Charlie Nash^{* 1} Conor Durkan^{* 1}

Abstract

Neural density estimators are flexible families of parametric models which have seen widespread use in unsupervised machine learning in recent years. Maximum-likelihood training typically dictates that these models be constrained to specify an explicit density. However, this limitation can be overcome by instead using a neural network to specify an energy function, or unnormalized density, which can subsequently be normalized to obtain a valid distribution. The challenge with this approach lies in accurately estimating the normalizing constant of the high-dimensional energy function. We propose the Autoregressive Energy Machine, an energy-based model which simultaneously learns an unnormalized density and computes an importance-sampling estimate of the normalizing constant for each conditional in an autoregressive decomposition. The Autoregressive Energy Machine achieves state-of-the-art performance on a suite of density-estimation tasks.

1. Introduction

Modeling the joint distribution of high-dimensional random variables is a key task in unsupervised machine learning. In contrast to other unsupervised approaches such as variational autoencoders (Kingma & Welling, 2013; Rezende et al., 2014) or generative adversarial networks (Goodfellow et al., 2014), neural density estimators allow for exact density evaluation, and have enjoyed success in modeling natural images (van den Oord et al., 2016b; Dinh et al., 2017; Salimans et al., 2017; Kingma & Dhariwal, 2018), audio data (van den Oord et al., 2016a; Prenger et al., 2018; Kim et al., 2018), and also in variational inference (Rezende & Mohamed, 2015; Kingma et al., 2016). Neural density estimators are particularly useful where the focus is on accurate density estimation rather than sampling, and these

models have seen use as surrogate likelihoods (Papamakarios et al., 2019) and approximate posterior distributions (Papamakarios & Murray, 2016; Lueckmann et al., 2017) for likelihood-free inference.

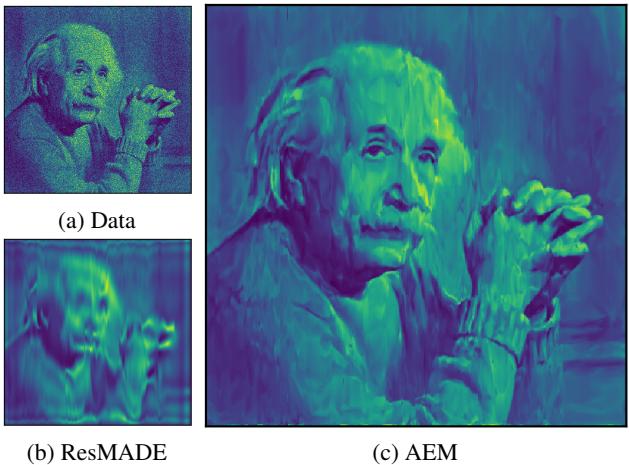


Figure 1: Accurately modeling a distribution with sharp transitions and high-frequency components, such as the distribution of light in a image (a), is a challenging task. We find that an autoregressive energy-based model (c) is able to preserve fine detail lost by an alternative model (b) with explicit conditionals.

Neural networks are flexible function approximators, and promising candidates to learn a probability density function. Typically, neural density models are normalized a priori, but this can hinder flexibility and expressiveness. For instance, many flow-based density estimators (Dinh et al., 2017; Papamakarios et al., 2017; Huang et al., 2018) rely on invertible transformations with tractable Jacobian which map data to a simple base density, so that the log probability of an input point can be evaluated using a change of variables. Autoregressive density estimators (Uria et al., 2013; Germain et al., 2015) often rely on mixtures of parametric distributions to model each conditional. Such families can make it difficult to model the low-density regions or sharp transitions characterized by multi-modal or discontinuous densities, respectively.

The contributions of this work are shaped by two main observations.

^{*}Equal contribution ¹School of Informatics, University of Edinburgh, United Kingdom. Correspondence to: Charlie Nash <charlie.nash@ed.ac.uk>, Conor Durkan <conor.durkan@ed.ac.uk>.

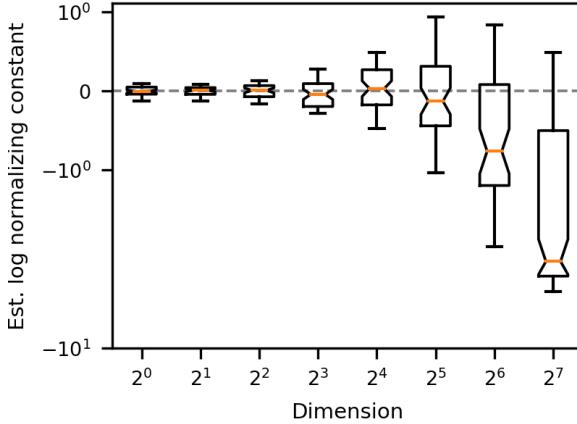


Figure 2: Importance sampling estimates of log normalizing constants deteriorate with increasing dimension. The target and proposal distributions are spherical Gaussians with $\sigma = 1$ and $\sigma = 1.25$, respectively. The true log normalizing constant is $\log Z = 0$. We plot the distribution of estimates over 50 trials, with each trial using 20 importance samples.

- An energy function, or unnormalized density, fully characterizes a probability distribution, and neural networks may be better suited to learning such an energy function rather than an explicit density.
- Decomposing the density estimation task in an autoregressive manner makes it possible to train such an energy-based model by maximum likelihood, since it is easier to obtain reliable estimates of normalizing constants in low dimensions.

Based on these observations, we present a scalable and efficient learning algorithm for an autoregressive energy-based model, which we term the *Autoregressive Energy Machine* (AEM). Figure 3 provides a condensed overview of how an AEM approximates the density of an input point.

2. Background

2.1. Autoregressive neural density estimation

A probability density function assigns a non-negative scalar value $p(\mathbf{x})$ to each vector-valued input \mathbf{x} , with the property that $\int p(\mathbf{x}) d\mathbf{x} = 1$ over its support. Given a dataset $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^N$ of N i.i.d. samples drawn from some unknown D -dimensional distribution $p^*(\mathbf{x})$, the density estimation task is to determine a model $p(\mathbf{x})$ such that $p(\mathbf{x}) \approx p^*(\mathbf{x})$. Neural density estimators are parametric models that make use of neural network components to increase their capacity to fit complex distributions, and autoregressive neural models are among the best performing of these.

The product rule of probability allows us to decompose

any joint distribution $p(\mathbf{x})$ into a product of conditional distributions:

$$p(\mathbf{x}) = \prod_{d=1}^D p(x_d | \mathbf{x}_{<d}). \quad (1)$$

Autoregressive density estimators model each conditional using parameters which are computed as a function of the preceding variables in a given ordering. In this paper, we use the term ARNN to describe any autoregressive neural network which computes an autoregressive function of a D -dimensional input \mathbf{x} , where the d^{th} output is denoted $\mathbf{f}(\mathbf{x}_{<d})$. The vector $\mathbf{f}(\mathbf{x}_{<d})$ is often interpreted as the parameters of a tractable parametric density for the d^{th} conditional, such as a mixture of location-scale distributions or the probabilities of a categorical distribution, but it is not restricted to this typical use-case.

Certain architectures, such as those found in recurrent models, perform the autoregressive computation sequentially, but more recent architectures exploit masking or causal convolution in order to output each conditional in a single pass of the network. Both types of architecture have found domain-specific (Sundermeyer et al., 2012; Theis & Bethge, 2015; Parmar et al., 2018), as well as general-purpose (Uria et al., 2013; Germain et al., 2015) use. In particular we highlight MADE (Germain et al., 2015), an architecture that masks weight matrices in fully connected layers to achieve causal structure. It is a building block in many models with autoregressive components (Kingma et al., 2016; Papamakarios et al., 2017; Huang et al., 2018), and we make use of a similar architecture in this work.

2.2. Energy-based models

In addition to an autoregressive decomposition, we may also write any density $p(\mathbf{x})$ as

$$p(\mathbf{x}) = \frac{e^{-\mathcal{E}(\mathbf{x})}}{Z}, \quad (2)$$

where $e^{-\mathcal{E}(\mathbf{x})}$ is the unnormalized density, $\mathcal{E}(\mathbf{x})$ is known as the energy function, and $Z = \int e^{-\mathcal{E}(\mathbf{x})} d\mathbf{x}$ is the normalizing constant. Assuming Z is finite, specifying an energy function is equivalent to specifying a probability distribution, since the normalizing constant is also defined in terms of $\mathcal{E}(\mathbf{x})$. Models described in this way are known as energy-based models. Classic examples include Boltzmann machines (Hinton, 2002; Salakhutdinov & Hinton, 2009; Hinton, 2012), products of experts (Hinton, 2002) and Markov random fields (Osindero & Hinton, 2007; Köster et al., 2009).

In order to do maximum-likelihood estimation of the parameters of an energy-based model, we must be able to evaluate or estimate the normalizing constant Z . This is

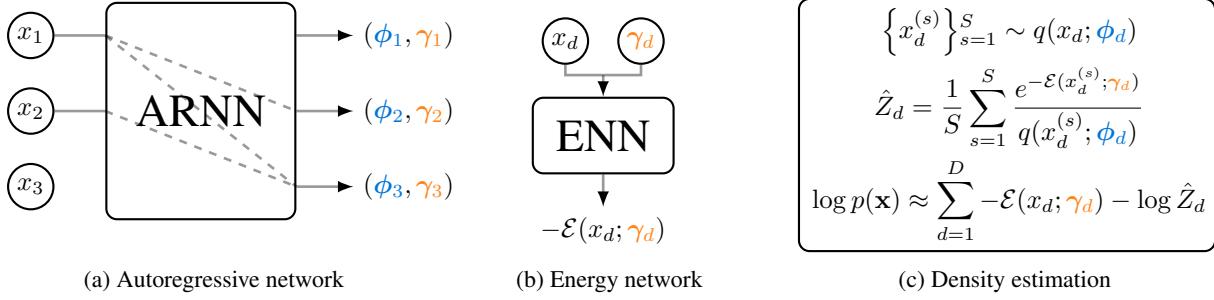


Figure 3: Overview of an AEM. (a) An autoregressive neural network computes an autoregressive function of an input \mathbf{x} , such that the d^{th} output depends only on $\mathbf{x}_{<d}$. The d^{th} output is a pair of vectors (ϕ_d, γ_d) which correspond to the proposal parameters and context vector, respectively, for the d^{th} conditional distribution. (b) The context vector γ_d and input x_d are passed through the energy network to compute an unnormalized log probability for the d^{th} conditional. (c) The parameters ϕ_d define a tractable proposal distribution, such as a mixture of location-scale family distributions, which can be used to compute an estimate of the normalizing constant \hat{Z}_d for the d^{th} conditional by importance sampling.

problematic, as it requires the estimation of a potentially high-dimensional integral. As such, a number of methods have been proposed to train energy-based models, which either use cheap approximations of the normalizing constant, or side-step the issue entirely. These include contrastive divergence (Hinton, 2002), noise-contrastive estimation (Gutmann & Hyvärinen, 2010; Ceylan & Gutmann, 2018), and score matching (Hyvärinen, 2005). In our case, phrasing the density estimation problem in an autoregressive manner allows us to make productive use of importance sampling, a stochastic approximation method for integrals.

Importance sampling. Given a proposal distribution $q(\mathbf{x})$ which is non-zero whenever the target $p(\mathbf{x}) \propto e^{-\mathcal{E}(\mathbf{x})}$ is non-zero, we can approximate the normalizing constant Z by

$$Z = \int e^{-\mathcal{E}(\mathbf{x})} d\mathbf{x} = \int \frac{e^{-\mathcal{E}(\mathbf{x})}}{q(\mathbf{x})} q(\mathbf{x}) d\mathbf{x} \quad (3)$$

$$\approx \frac{1}{S} \sum_{s=1}^S \frac{e^{-\mathcal{E}(\mathbf{x}^{(s)})}}{q(\mathbf{x}^{(s)})}, \quad \mathbf{x}^{(s)} \sim q(\mathbf{x}), \quad (4)$$

and this expression is an unbiased estimate of the normalizing constant for $p(\mathbf{x})$. The quotients in the summand are known as the importance weights. When $q(\mathbf{x})$ does not closely match $p(\mathbf{x})$, the importance weights will have high variance, and the importance sampling estimate will be dominated by those terms with largest weight. Additionally, when $q(\mathbf{x})$ does not adequately cover regions of high density under $p(\mathbf{x})$, importance sampling underestimates the normalizing constant (Salakhutdinov & Murray, 2008). Finding a suitable distribution $q(\mathbf{x})$ which closely matches $p(\mathbf{x})$ is problematic, since estimating the potentially complex distribution $p(\mathbf{x})$ is the original problem under consideration. This issue is exacerbated in higher dimensions, and importance sampling estimates may be unreliable in such cases. Figure 2 demonstrates how the accuracy of an importance

sampling estimate of the normalizing constant for a standard normal distribution deteriorates as dimensionality increases.

3. Autoregressive Energy Machines

The ability to specify a probability distribution using an energy function is enticing, since now a neural network can take on the more general role of an energy function in a neural density estimator. Further decomposing the task in an autoregressive manner means that importance sampling offers a viable method for maximum likelihood training, generally yielding reliable normalizing constant estimates in the one-dimensional case when the proposal distribution is reasonably well matched to the target (fig. 2). As such, the main contribution of this paper is to combine eq. (1) and eq. (2) in the context of a neural density estimator. We model a density function $p(\mathbf{x})$ as a product of D energy terms

$$p(\mathbf{x}) = \prod_{d=1}^D p(x_d | \mathbf{x}_{<d}) = \prod_{d=1}^D \frac{e^{-\mathcal{E}(x_d; \mathbf{x}_{<d})}}{Z_d}, \quad (5)$$

where $Z_d = \int e^{-\mathcal{E}(x_d; \mathbf{x}_{<d})} dx_d$ is the normalizing constant for the d^{th} conditional. If we also specify an autoregressive proposal distribution $q(\mathbf{x}) = \prod_d q(x_d | \mathbf{x}_{<d})$, we can estimate the normalizing constant for each of the D terms in the product by importance sampling:

$$Z_d = \int e^{-\mathcal{E}(x_d; \mathbf{x}_{<d})} dx_d \quad (6)$$

$$\approx \frac{1}{S} \sum_{s=1}^S \frac{e^{-\mathcal{E}(x_d^{(s)}; \mathbf{x}_{<d})}}{q(x_d^{(s)}; \mathbf{x}_{<d})}, \quad x_d^{(s)} \sim q(x_d; \mathbf{x}_{<d}). \quad (7)$$

This setup allows us to make use of arbitrarily complex energy functions, while relying on importance sampling to estimate normalizing constants in one dimension; a much more

tractable problem than estimation of the full D -dimensional integral.

3.1. A neural energy function

We implement the energy function as a neural network ENN that takes as input a scalar x_d as well as a context vector γ_d that summarizes the dependence of x_d on the preceding variables $\mathbf{x}_{<d}$. The ENN directly outputs the negative energy, so that $-\mathcal{E}(x_d; \mathbf{x}_{<d}) = -\mathcal{E}(x_d; \gamma_d) = \text{ENN}(x_d; \gamma_d)$. In our experiments, the ENN is a fully-connected network with residual connections (He et al., 2016a). To incorporate the context vector γ_d , we found concatenation to the input x_d to work well in practice. We share ENN parameters across dimensions, which reduces the total number of parameters, and allows us to learn features of densities which are common across dimensions. We also constrain the output of the ENN to be non-positive using a softplus non-linearity, so that the unnormalized density is bounded by one, since this improved training stability.

3.2. Learning in an AEM

We denote by ϕ_d the vector of parameters for the d^{th} proposal conditional, which, like the context vector γ_d , is computed as a function of $\mathbf{x}_{<d}$. In our case, this quantity consists of the mixture coefficients, locations, and scales of a tractable parametric distribution, and we find that a mixture of Gaussians works well across a range of tasks. The normalizing constant for the d^{th} conditional can thus be approximated by

$$\hat{Z}_d = \frac{1}{S} \sum_{s=1}^S \frac{e^{-\mathcal{E}(x_d^{(s)}; \gamma_d)}}{q(x_d^{(s)}; \phi_d)}, \quad x_d^{(s)} \sim q(x_d; \phi_d), \quad (8)$$

leading to an expression for the approximate log density of an input data point \mathbf{x}

$$\log p(\mathbf{x}) \approx \sum_{d=1}^D -\mathcal{E}(x_d; \gamma_d) - \log \hat{Z}_d. \quad (9)$$

Estimation of log densities therefore requires the energy network to be evaluated $S + 1$ times for each conditional; S times for the importance samples, and once for the data point x_d . In practice, we perform these evaluations in parallel, by passing large batches consisting of input data and importance samples for all conditionals to the energy net along with the relevant context vectors, and found $S = 20$ to be sufficient. Although the importance sampling estimates of the normalizing constants are unbiased, by taking the logarithm of \hat{Z}_d in eq. (9) we bias our estimates of $\log p(\mathbf{x})$. However, as we will show in Section 4.2, our estimates are well-calibrated, and can be made more accurate by increasing the number of importance samples. For the purposes of training, we did not find this bias to be an issue.

As illustrated in fig. 3a, we obtain both context vectors and proposal parameters in parallel using an autoregressive network ARNN. The ARNN outputs proposal parameters ϕ_d and context vectors γ_d for each of the D dimensions in a single forward pass. These quantities are used both to estimate $\log p(\mathbf{x})$ as in eq. (9), as well as to evaluate $\log q(\mathbf{x})$, and we form a maximum-likelihood training objective

$$\mathcal{L}(\theta; \mathbf{x}) = \log p(\mathbf{x}) + \log q(\mathbf{x}), \quad (10)$$

where θ refers collectively to the trainable parameters in both the ARNN and the ENN. We fit the AEM by maximizing eq. (10) across a training set using stochastic gradient ascent, and find that a warm-up period where the proposal is initially optimized without the energy model can improve stability, allowing the proposal to cover the data sufficiently before importance sampling begins.

It is important to note that we do not optimize the proposal distribution parameters ϕ with respect to the importance sampling estimate. This means that the proposal is trained independently of the energy-model, and estimates of $\log p(\mathbf{x})$ treat the proposal samples and proposal density evaluations as constant values for the purposes of optimization. In practice, this is implemented by stopping gradients on variables connected to the proposal distribution in the computational graph. We find maximum-likelihood training of q to be effective as a means to obtain a useful proposal distribution, but other objectives, such as minimization of the variance of the importance sampling estimate (Kuleshov & Ermon, 2017; Müller et al., 2018), might also be considered, although we do not investigate this avenue in our work.

3.3. Sampling

Although it is not possible to sample analytically from our model, we can obtain approximate samples by first drawing samples from the proposal distribution and then resampling from this collection using importance weights computed by the energy model (eq. (4)). This method is known as sampling importance resampling (Rubin, 1988), and is consistent in the limit of infinite proposal samples, but we found results to be satisfactory using just 100 proposal samples.

3.4. ResMADE

For general purpose density estimation of tabular data, we present a modified version of the MADE architecture (Germain et al., 2015) that incorporates residual connections (He et al., 2016a). In the standard MADE architecture, causal structure is maintained by masking certain weights in the network layers. We observe that in consecutive hidden layers with the same number of units and shared masks, the connectivity of the hidden units with respect to the inputs is preserved. As such, incorporating skip connections will

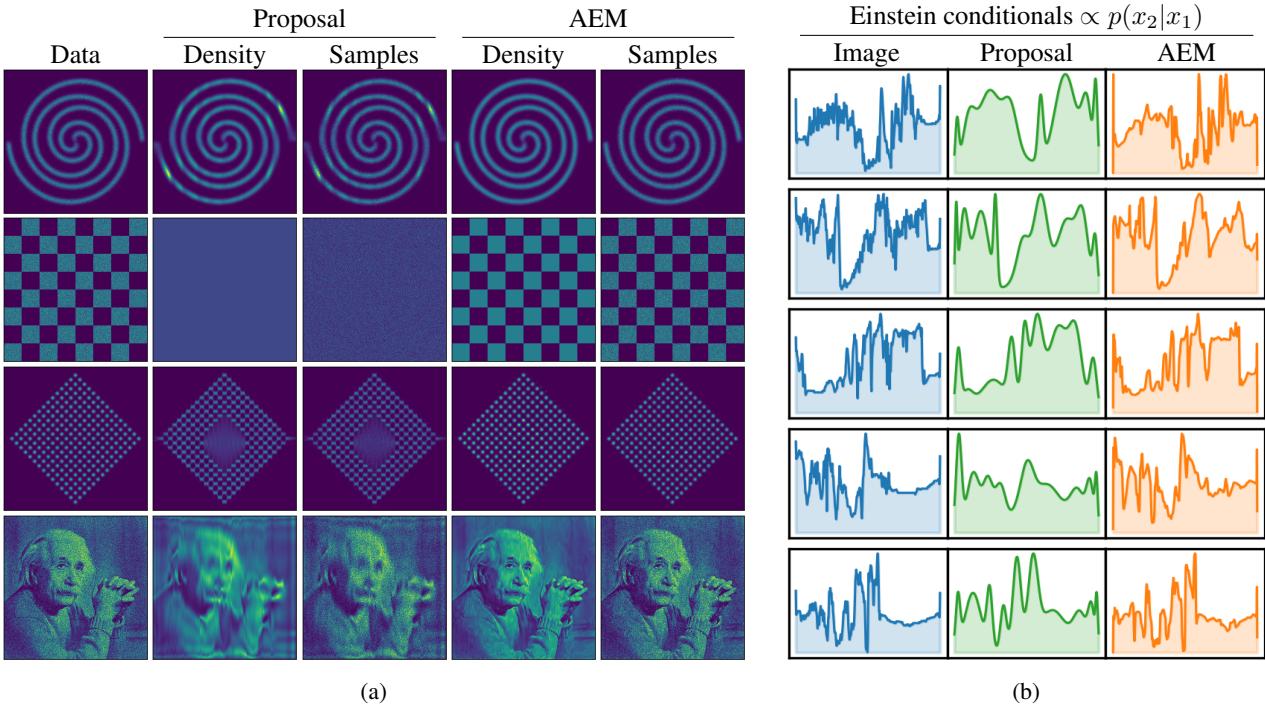


Figure 4: (a) Estimated densities and samples for both the proposal distribution and AEM on a range of synthetic two-dimensional densities. For the checkerboard grid (row 2), we used fixed uniform conditionals for the proposal distribution. AEM densities are evaluated with a normalizing constant estimated using 1000 importance samples. We present histograms of 10^6 samples from each trained model, for comparison with each training dataset of the same cardinality. (b) Unnormalized conditional distributions proportional to $p(x_2|x_1)$ for a selection of vertical slices on the Einstein task. An energy-based approach allows for a better fit to the true pixel intensities in the image, which feature high-frequency components ill-suited for a finite mixture of Gaussians.

maintain the architecture’s autoregressive structure.

Following the documented success of residual blocks (He et al., 2016a;b) as a component in deep network architectures, we implement a basic residual block for MADE-style models that can be used as a drop-in replacement for typical masked layers. Each block consists of two masked-dense layers per residual block, and uses pre-activations following He et al. (2016b). We use the term ResMADE to describe an autoregressive architecture featuring these blocks, and make use of the ResMADE as an ARNN component across our experiments. For a more detailed description see Appendix A.

4. Experiments

For our experiments, we use a ResMADE with four residual blocks for the ARNN, as well as a fully-connected residual architecture for the ENN, also with four residual blocks. The number of hidden units in the ResMADE is varied per task. We use the Adam optimizer (Kingma & Ba, 2014), and anneal the learning rate to zero over the course of training using a cosine schedule (Loshchilov & Hutter,

2016). For some tasks, we find regularization by dropout (Srivastava et al., 2014) to be beneficial. Full experimental details are available in Appendix B, and code is available at <https://github.com/conormdurkan/autoregressive-energy-machines>.

4.1. Synthetic datasets

We first demonstrate that an AEM is capable of fitting complex two-dimensional densities. For each task, we generate 10^6 data points for training. Results are displayed in fig. 4a. We plot each AEM density by estimating the normalizing constant with 1000 importance samples for each conditional. AEM samples are obtained by resampling 100 proposal samples as described in section 3.3.

Spirals The spirals dataset is adapted from Grathwohl et al. (2019). Though capable of representing the spiral density, the ResMADE proposal fails to achieve the same quality of fit as an AEM, with notable regions of non-uniform density.

Checkerboard The checkerboard dataset is also adapted from Grathwohl et al. (2019). This task illustrates that in some cases a learned proposal distribution is not required;

with a fixed, uniform proposal, an AEM is capable of accurately modeling the data, including the discontinuities at the boundary of each square.

Diamond The diamond task is adapted from the 100-mode square Gaussian grid of [Huang et al. \(2018\)](#), by expanding to 225 modes, and rotating 45 degrees. Although the ResMADE proposal does not have the capacity to represent all modes of the target data, an AEM is able to recover these lost modes.

Einstein We generate the Einstein data by sampling coordinates proportional to the pixel intensities in an image of Albert Einstein ([Müller et al., 2018](#)), before adding uniform noise, and re-scaling the resulting points to $[0, 1]^2$. This task in particular highlights the benefits of an energy-based approach. The distribution of light in an image features sharp transitions and edges, high-frequency components, and broad regions of near-constant density. Where a ResMADE proposal struggles with these challenges, an AEM is able to retain much more fine detail. In addition, samples generated by the AEM are difficult to distinguish from the original dataset.

Finally, fig. 4b presents an alternative visualization of the Einstein task. Each row corresponds to an unnormalized conditional proportional to $p(x_2|x_1)$ for a fixed value of x_1 . While the ResMADE proposal, consisting of a mixture of 10 Gaussians for each conditional, achieves a good overall fit to the true pixel intensities, it is ultimately constrained by the smoothness of its mixture components.

4.2. Normalizing constant estimation

Though importance sampling in one dimension is much less unwieldy than in high-dimensional space, it may still be the case that the proposal distributions do not adequately cover the support of the conditionals being modeled, leading to underestimates of the normalizing constants. Here we demonstrate that the normalizing constants learned by an AEM are well-calibrated by comparing to ‘true’ values computed with explicit numerical integration. In particular, we use a log-modified trapezoidal rule ([Pitkin, 2017](#)) to integrate the unnormalized log density output by the energy network over each dimension. This approach exploits the fast parallel computation of the energy net, allowing us to saturate the domain of integration, and compensate for the shortcomings of the trapezoidal rule compared to more advanced adaptive quadrature methods ([Gander & Gautschi, 2000](#)). We increase the number of integrand evaluations until the integral converges to seven significant figures.

For each trained model on the Einstein, Power, Hepmass, and BSDS300 tasks, we first randomly select 1000 dimensions with replacement according to the data dimensionality, disregarding the marginal $p(x_1)$ so as to not repeatedly com-

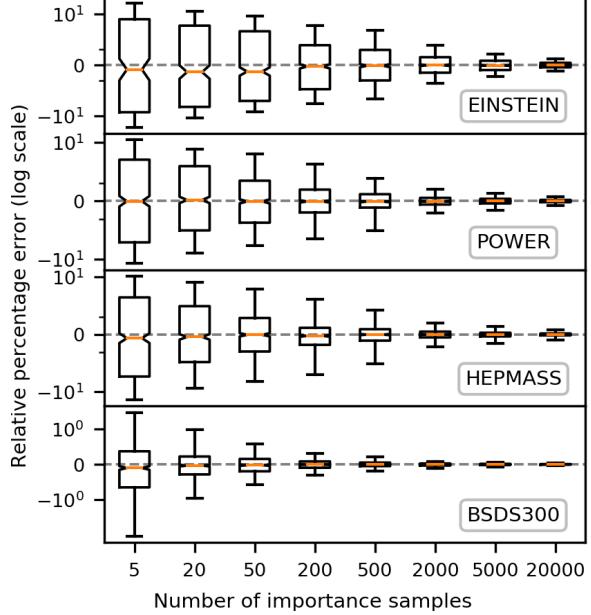


Figure 5: Accuracy of log normalizing constant estimates increases with number of importance samples for each task. Whiskers delineate the 5th and 95th percentiles of the relative error. The Einstein conditionals, as illustrated in fig. 4b, prove particularly difficult, while BSDS300 conditionals are in contrast much simpler to approximate.

pute the same value. Then, we compute the integral of the log unnormalized density corresponding to that one-dimensional conditional, using a log-trapezoidal rule and context vectors generated from a held out-validation set of 1000 samples. This procedure results in 1000 ‘true’ integrals for each task. We then test the AEM by comparing this true value with estimates generated using increasing numbers of importance samples. Note that in each task, the AEM has never estimated the log normalizing constant for the conditional densities under consideration, since these conditionals are specified using the validation set, and not the training set. Figure 5 visualizes the results of our calibration experiments.

4.3. Density estimation on tabular data

We follow the experimental setup of [Papamakarios et al. \(2017\)](#) in using a selection of pre-processed datasets from the UCI machine learning repository ([Dheeru & Karra Taniskidou, 2017](#)), and BSDS300 datasets of natural images ([Martin et al., 2001](#)). AEM log likelihoods are estimated using 20000 importance samples from the proposal distribution.

As a normalized approximation to the AEM, we use a kernel density estimate in which Gaussian kernels are centered at proposal samples and weighted by the importance weights.

Table 1: Test log likelihood (in nats) for UCI datasets and BSDS300, with error bars corresponding to two standard deviations. AEM* results are estimated with 20,000 importance samples. The best performing model for each dataset is shown in bold, as well the best performing model for which the exact log likelihood can be obtained. Results for non-AEM models taken from existing literature. MAF-DDSF[†] report error bars across five repeated runs rather than across the test set.

MODEL	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
MADE-MoG	0.40 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
MAF	0.30 ± 0.01	10.08 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
MAF-DDSF [†]	0.62 ± 0.01	11.96 ± 0.33	-15.09 ± 0.40	-8.86 ± 0.15	157.73 ± 0.04
TAN (VARIOUS)	0.60 ± 0.01	12.06 ± 0.02	-13.78 ± 0.02	-11.01 ± 0.48	159.80 ± 0.07
RESMADE-MoG (PROPOSAL)	0.61 ± 0.01	12.80 ± 0.01	-13.42 ± 0.01	-11.01 ± 0.23	157.41 ± 0.14
AEM-KDE	0.65 ± 0.01	12.89 ± 0.01	-12.87 ± 0.01	-10.33 ± 0.22	158.44 ± 0.14
AEM*	0.70 ± 0.01	13.03 ± 0.01	-12.85 ± 0.01	-10.17 ± 0.26	158.71 ± 0.14

Table 2: Latent variable modeling results with AEM priors. AEM-VAE* results obtained with 1000 importance samples and $\log p(\mathbf{x})$ lower-bounded using the method of Burda et al. (2016) with 50 samples. IAF-DSF[†] report error bars across five repeated runs rather than across the test set.

Model	ELBO	$\log p(\mathbf{x})$
IAF-DSF [†]	-81.92 ± 0.04	-79.86 ± 0.01
VAE	-84.43 ± 0.23	-81.23 ± 0.21
ResMADE-VAE	-82.96 ± 0.23	-79.89 ± 0.21
AEM-KDE-VAE	-82.95 ± 0.23	-79.88 ± 0.21
AEM-VAE*	-82.92 ± 0.23	-79.87 ± 0.21

We include the proposal itself as a mixture component in order to provide probability density in regions not well-covered by the samples. The KDE bandwidth and proposal distribution mixture weighting are optimized on the validation set. We call the model under this evaluation scheme AEM-KDE. KDE estimates also use 20000 samples from the proposal distribution for each conditional.

Table 1 shows the test-set log likelihoods obtained by our models and by other state-of-the-art models. We first note that the AEM proposal distribution (ResMADE-MoG) provides a strong baseline relative to previous work. In particular, it improves substantially on the the MADE-MoG results reported by Papamakarios et al. (2017), and improves on the state-of-the-art results reported by NAF (Huang et al., 2018) and TAN (Oliva et al., 2018). This demonstrates the benefits of adding residual connections to the MADE architecture. As such, practitioners may find ResMADE a useful component in many applications which require autoregressive computation.

AEM and AEM-KDE outperform both the proposal distribution and existing state-of-the-art methods on the Power, Gas and Hepmass datasets, demonstrating the potential benefit of flexible energy-based conditionals. Despite regularization,

overfitting was an issue for Miniboone due to the size of the training set ($n = 29,556$) relative to the data dimension ($D = 43$). This highlights the challenges associated with using very expressive models on domains with limited data, and the need for stronger regularization methods in these cases. On BSDS300, our models achieve the second highest scores relative to previous work. On this dataset we found that the validation scores (~ 174) were substantially higher than test-set scores (~ 158), indicating differences between their empirical distributions. Overall, the AEM-KDE obtains improved scores relative to the proposal distribution, and these scores are close to those of the AEM.

4.4. Latent variable modeling

We evaluate the AEM in the context of deep latent-variable models, where it can be used as an expressive prior. We train a convolutional variational autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014), making use of residual blocks in the encoder and decoder in a similar architecture to previous work (Huang et al., 2018; Kingma et al., 2016). We initially train the encoder and decoder with a standard Gaussian prior, and then train an AEM post-hoc to maximize the likelihood of samples from the aggregate approximate posterior

$$q_{\text{agg}}(\mathbf{z}) = \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})}[q(\mathbf{z}|\mathbf{x})]. \quad (11)$$

This training method avoids an issue where maximum-likelihood training of the proposal distribution interferes with the lower bound objective. During pre-training, we set $\beta = 0.9$ in the modified variational objective

$$\mathcal{L}_\beta = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})). \quad (12)$$

This weighting of the KL-divergence term has the effect of boosting the reconstruction log probability at the expense of an aggregate posterior that is less well-matched to the standard Gaussian prior. This provides an opportunity for the AEM to improve on the original prior as a model of

the aggregate posterior. Weighting of the KL-divergence has been used in previous work to reduce the occurrence of unused latent variables (Sønderby et al., 2016), and to control the types of representations encoded by the latent variables (Higgins et al., 2017).



Figure 6: Binarized MNIST data examples and unconditional samples for AEM-VAE obtained by resampling from 100 proposal samples.

Table 2 shows that the AEM-VAE improves substantially on the standard Gaussian prior, and that the results are competitive with existing approaches on dynamically-binarized MNIST (Burda et al., 2016). The AEM does not improve on the proposal distribution scores, possibly because the aggregate latent posterior is a mixture of Gaussians, which is well-modeled by the ResMADE-MoG. Figure 6 shows data examples and samples from the trained model.

5. Related Work

Flow-based neural density estimation Together with autoregressive approaches, flow-based models have also seen widespread use in neural density estimation. Flow-based models consist of invertible transformations for which Jacobian determinants can be efficiently computed, allowing exact density evaluation through the change-of-variables formula. Multiple transformations are often stacked to enable more complex models.

Efficiently invertible flows Flows exploiting a particular type of transformation, known as a coupling layer, not only allow for one-pass density evaluation, but also one-pass sampling. Examples of such models include NICE (Dinh et al., 2014) and RealNVP (Dinh et al., 2017), and the approach has recently been extended to image (Kingma & Dhariwal, 2018) and audio (Prenger et al., 2018; Kim et al., 2018) data. The case of continuous flows based on ordinary differential equations has also recently been explored by FFJORD (Grathwohl et al., 2019). However, efficient sampling for this class of models comes at the cost of density estimation performance, with autoregressive models generally achieving better log likelihood scores.

Autoregressive flows Originally proposed by Kingma et al. (2016) for variational inference, autoregressive flows were adapted for efficient density estimation by Papamakarios et al. (2017) with MAF. Subsequent models such as NAF (Huang et al., 2018) and TAN (Oliva et al., 2018) have developed on this idea, reporting state-of-the-art results for density estimation. Sampling in these models is expensive, since autoregressive flow inversion is inherently sequential. In some cases, such as Huang et al. (2018), the flows do not have an analytic inverse, and must be inverted numerically for sampling. Despite these caveats, autoregressive density estimators remain the best performing neural density estimators for general density estimation tasks.

Energy-based models In this work we describe energy-based models as unnormalized densities that define a probability distribution over random variables. However, there exist multiple notions of energy-based learning in the machine learning literature, including non-probabilistic interpretations (LeCun et al., 2006; Zhao et al., 2017). We focus here on recent work which includes applications to density estimation with neural energy functions. Deep energy estimator networks (Saremi et al., 2018) use an energy function implemented as a neural network, and train using the score-matching framework. This objective avoids the need to estimate the normalizing constant, but also makes it challenging to compare log-likelihood scores with other density estimators. Bauer & Mnih (2018) propose an energy-based approach for increasing the flexibility of VAE priors, in which a neural network energy function is used to mask a pre-specified proposal function. As in our work, training is performed using importance sampling, but due to the larger dimensionality of the problem, 2^{10} samples were used in the estimates during training.

Other related work Müller et al. (2018) propose neural importance sampling, in which a flow-based neural sampler is optimized in order to perform low-variance Monte Carlo integration of a given target function. This is similar to the goal of the proposal distribution in the AEM, but in our case the proposal is trained jointly with an energy model, and we do not assume that the target function is known a priori.

6. Conclusion

We proposed the Autoregressive Energy Machine, a neural density estimator that addresses the challenges of energy-based modeling in high dimensions through a scalable and efficient autoregressive estimate of the normalizing constant. While exact density evaluation is intractable for an AEM, we have demonstrated that the flexibility of an energy-based model enables us to model challenging synthetic data, as well as achieve state-of-the-art results on a suite of benchmark datasets.

Acknowledgements

The authors thank George Papamakarios, Iain Murray, and Chris Williams for helpful discussion. This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

- Bauer, M. and Mnih, A. Resampled priors for variational autoencoders. *arXiv preprint arXiv:1810.11428*, 2018.
- Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *International conference on Learning Representations*, 2016.
- Ceylan, C. and Gutmann, M. U. Conditional noise-contrastive estimation of unnormalised models. *International Conference on Machine Learning*, 2018.
- Dheeru, D. and Karra Taniskidou, E. UCI Machine Learning Repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *International conference on Learning Representations workshops*, 2014.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations*, 2017.
- Gander, W. and Gautschi, W. Adaptive Quadrature Revisited. *BIT Numerical Mathematics*, 40(1):84–101, 2000.
- Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked Autoencoder for Distribution Estimation. *International Conference on Machine Learning*, 2015.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. pp. 2672–2680, 2014.
- Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.
- Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016a.
- He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. *ECCV (4)*, 2016b.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. β -VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017.
- Hinton, G. E. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 2002.
- Hinton, G. E. A practical guide to training restricted Boltzmann machines. In *Neural networks: Tricks of the trade*, pp. 599–619. Springer, 2012.
- Huang, C., Krueger, D., Lacoste, A., and Courville, A. C. Neural autoregressive flows. 2018.
- Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 2005.
- Kim, S., Lee, S.-g., Song, J., and Yoon, S. FloWaveNet: A Generative Flow for Raw Audio. *arXiv preprint arXiv:1811.02155*, 2018.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.
- Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1×1 convolutions. *Advances in Neural Information Processing Systems*, 2018.
- Kingma, D. P. and Welling, M. Auto-encoding Variational Bayes. *International Conference on Learning Representations*, 2013.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 2016.
- Köster, U., Lindgren, J. T., and Hyvärinen, A. Estimating Markov Random Field Potentials for Natural Images. *ICA*, 2009.
- Kuleshov, V. and Ermon, S. Neural variational inference and learning in undirected graphical models. *Advances in Neural Information Processing Systems*, 2017.
- LeCun, Y., Chopra, S., and Hadsell, R. A tutorial on energy-based learning. 2006.
- Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

- Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems*, 2017.
- 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. 2:416–423, July 2001.
- Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural Importance Sampling. *arXiv preprint arXiv:1808.03856*, 2018.
- Oliva, J. B., Dubey, A., Póczos, B., Schneider, J., and Xing, E. P. Transformation Autoregressive Networks. *International Conference on Machine Learning*, 2018.
- Osindero, S. and Hinton, G. E. Modeling image patches with a directed hierarchy of Markov Random Fields. *Advances in Neural Information Processing Systems*, 2007.
- Papamakarios, G. and Murray, I. Fast ε -free inference of simulation models with Bayesian conditional density estimation. *Advances in Neural Information Processing Systems*, 2016.
- Papamakarios, G., Murray, I., and Pavlakou, T. Masked Autoregressive Flow for Density Estimation. *Advances in Neural Information Processing Systems*, 2017.
- Papamakarios, G., Sterratt, D. C., and Murray, I. Sequential Neural Likelihood: Fast likelihood-free inference with autoregressive flows. *AISTATS*, 2019.
- Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image Transformer. *International Conference on Machine Learning*, 2018.
- Pitkin, M. `lintegrate`. <https://github.com/mattptkin/lintegrate>, 2017.
- Prenger, R., Valle, R., and Catanzaro, B. Waveglow: A flow-based generative network for speech synthesis. *arXiv preprint arXiv:1811.00002*, 2018.
- Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *International Conference on Machine Learning*, 2015.
- Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, 2014.
- Rubin, D. B. Using the SIR algorithm to simulate posterior distributions. *Bayesian statistics*, 3:395–402, 1988.
- Salakhutdinov, R. and Hinton, G. E. Deep Boltzmann Machines. *AISTATS*, 2009.
- Salakhutdinov, R. and Murray, I. On the quantitative analysis of deep belief networks. *International Conference on Machine Learning*, 2008.
- Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. *International Conference on Learning Representations*, 2017.
- Saremi, S., Mehrjou, A., Schölkopf, B., and Hyvärinen, A. Deep Energy Estimator Networks. *arXiv preprint arXiv:1805.08306*, 2018.
- Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder Variational Autoencoders. *Advances in Neural Information Processing Systems*, 2016.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- Sundermeyer, M., Schlüter, R., and Ney, H. LSTM neural networks for language modeling. *Conference of the International Speech Communication Association*, 2012.
- Theis, L. and Bethge, M. Generative image modeling using spatial LSTMs. *Advances in Neural Information Processing Systems*, 2015.
- Uria, B., Murray, I., and Larochelle, H. RNADE: The real-valued neural autoregressive density-estimator. *Advances in Neural Information Processing Systems*, 2013.
- van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *ISCA Speech Synthesis Workshop*, 2016a.
- van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel Recurrent Neural Networks. *International Conference on Machine Learning*, 2016b.
- Zhao, J., Mathieu, M., and LeCun, Y. Energy-based Generative Adversarial Network. *International Conference on Learning Representations*, 2017.

A. ResMADE

Residual connections (He et al., 2016a) are widely used in deep neural networks, and have demonstrated favourable performance relative to standard networks. Residual networks typically consist of many stacked transformations of the form

$$\mathbf{h}_{l+1} = \mathbf{h}_l + \mathbf{f}(\mathbf{h}_l), \quad (13)$$

where \mathbf{f} is a residual block. In this work, we observe that it is possible to equip a MADE (Germain et al., 2015) with residual connections when certain conditions on its masking structure are met.

At initialization, each unit in each hidden layer of a MADE is assigned a positive integer, termed its degree. The degree specifies the number of input dimensions to which that particular unit is connected. For example, writing d_k^l for unit k of layer l , a value $d_k^l = m$ means that the unit depends only on the first m dimensions of the input, when the input is prescribed a particular ordering (we always assume the ordering given by the data). In successive layers, this means that a unit may only be connected to units in the previous layer whose degrees strictly do not exceed its own. In other words, the degree assignment defines a binary mask matrix which multiplies the weight matrix of each layer in a MADE elementwise, maintaining autoregressive structure. Indeed, the mask M for layer l is given in terms of the degrees for layer $l - 1$ and layer l :

$$M_{ij}^l = \begin{cases} 1 & \text{if } d_i^l \geq d_j^{l-1} \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

Through this masking process, it can be guaranteed that output units associated with data dimension d only depend on the previous inputs $\mathbf{x}_{<d}$. Though the original MADE paper considered a number of ways in which to assign degrees, we focus here on fixed sequential degree assignment, used also by MAF (Papamakarios et al., 2017). For an input of dimension D , we define the degree

$$d_k^l = (k - 1) \pmod{(D - 1)} + 1. \quad (15)$$

We also assume the dimension H of each hidden layer is at least D , so that no input information is lost. The result of sequential degree assignment for $D = 3$ and $H = 4$ is illustrated in fig. 7.

If all hidden layers in the MADE are of the same dimensionality H , sequential degree assignment means that each layer will also have the same degree structure. In this way, two vectors of hidden units in the MADE may be combined using any binary element-wise operation, also shown in Figure fig. 7. In particular, a vector computed from a fully-connected block with masked layers can be added

to the input to that block while maintaining the same autoregressive structure, allowing for the traditional residual connection to be added to the MADE architecture.

Not only do residual connections enhance a MADE in its own right, but the resulting ResMADE architecture can be used as a drop-in replacement wherever a MADE is used as a building block, such as IAF, MAF, or NAF (Kingma et al., 2016; Papamakarios et al., 2017; Huang et al., 2018).

B. Experimental settings

In all experiments, we use the same ResMADE and ENN architectures, each with 4 pre-activation residual blocks (He et al., 2016b). The number of hidden units and the dimensionality of the context vector for the ENN are fixed across all tasks at 128 and 64 respectively. The number of hidden units in the ResMADE is tuned per experiment. For the exact experimental settings used in each experiment, see Tables 3 and 4.

For the proposal distributions, we use a mixture of Gaussians in all cases except for the checkerboard experiment, where we use a fixed uniform distribution. We use 10 mixture components for the synthetic experiments, and 20 components for all other experiments. We use a minimum scale of 10^{-3} for the Gaussian distributions in order to prevent numerical issues.

For optimization we use Adam (Kingma & Ba, 2014) with a cosine annealing schedule (Loshchilov & Hutter, 2016) and an initial learning rate of 5×10^{-4} . The number of training steps is adjusted per task. For Miniboone, we use only 6000 training updates, as we found overfitting to be a significant problem for this dataset. Early-stopping is used to select models, although in most cases the best models are obtained at the end of training. Normalizing constants are estimated using 20 importance samples, and dropout is applied to counter overfitting, with a rate that is tuned per task. Dropout is applied between the two layers of each residual block in both the ENN and ResMADE.

For the VAE, we use an architecture similar to those used in IAF and NAF (Kingma et al., 2016; Huang et al., 2018). For the encoder, we alternate between residual blocks that preserve spatial resolution, and downsampling residual blocks with strided convolutions. The input is projected to 16 channels using a 1×1 convolution, and the number of channels is doubled at each downsampling layer. After three downsampling layers we flatten the feature maps and use a linear layer to regress the means and log-variances of the approximate posterior with 32 latent units. The decoder mirrors the encoder, with the input latents being linearly projected and reshaped to a $[4, 4, 128]$ spatial block, which is then upsampled using transpose-convolutions in order to output pixel-wise Bernoulli logits.

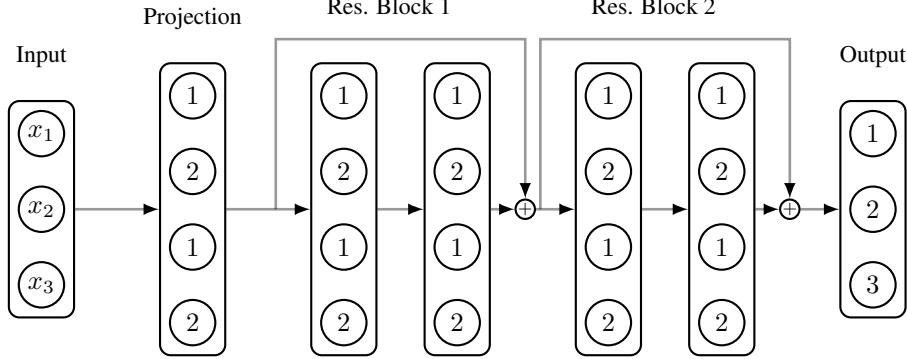


Figure 7: ResMADE architecture with $D = 3$ input data dimensions and $H = 4$ hidden units. The degree of each hidden unit and output is indicated with an integer label. Sequential degree assignment results in each hidden layer having the same masking structure, here alternating between dependence on the first input, or the first two inputs. These layers can be combined using any binary elementwise operation, while preserving autoregressive structure. In particular, residual connections can be added in a straightforward manner. The ResMADE architecture consists of an initial masked projection to the target hidden dimensionality, a sequence of masked residual blocks, and finally a masked linear layer to the output units.

Table 3: Experimental setting for synthetic data and VAE experiments.

HYPERPARAMETER	SPIRALS	CHECKERBOARD	DIAMOND	EINSTEIN	VAE
BATCH SIZE	256	256	256	256	256
RESMADE HIDDEN DIM.	256	256	256	256	512
RESMADE ACTIVATION	RELU	RELU	RELU	RELU	RELU
RESMADE DROPOUT	0	0	0	0	0.5
CONTEXT DIM.	64	64	64	64	64
ENN HIDDEN DIM.	128	128	128	128	128
ENN ACTIVATION	RELU	RELU	RELU	RELU	RELU
ENN DROPOUT	0	0	0	0	0.5
MIXTURE COMPS.	10	-	10	10	20
MIXTURE COMP. SCALE MIN.	1E-3	-	1E-3	1E-3	1E-3
LEARNING RATE	5E-4	5E-4	5E-4	5E-4	5E-4
TOTAL STEPS	400000	400000	400000	3000000	100000
WARM-UP STEPS	5000	0	0	0	0

Table 4: Experimental settings for UCI and BSDS300 datasets.

HYPERPARAMETER	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
BATCH SIZE	512	512	512	512	512
RESMADE HIDDEN DIM.	512	512	512	512	1024
RESMADE ACTIVATION	RELU	RELU	RELU	RELU	RELU
RESMADE DROPOUT	0.1	0	0.2	0.5	0.2
CONTEXT DIM.	64	64	64	64	64
ENN HIDDEN DIM.	128	128	128	128	128
ENN ACTIVATION	RELU	TANH	RELU	RELU	RELU
ENN DROPOUT	0.1	0	0.2	0.5	0.2
MIXTURE COMPS.	20	20	20	20	20
MIXTURE COMP. SCALE MIN.	1E-3	1E-3	1E-3	1E-3	1E-3
LEARNING RATE	5E-4	5E-4	5E-4	5E-4	5E-4
TOTAL STEPS	800000	400000	400000	6000	400000
WARM-UP STEPS	5000	5000	5000	0	5000