Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models

Sam Bond-Taylor, Adam Leach, Yang Long, Chris G. Willcocks

Abstract—Deep generative modelling is a class of techniques that train deep neural networks to model the distribution of training samples. Research has fragmented into various interconnected approaches, each of which making trade-offs including run-time, diversity, and architectural restrictions. In particular, this compendium covers energy-based models, variational autoencoders, generative adversarial networks, autoregressive models, normalizing flows, in addition to numerous hybrid approaches. These techniques are drawn under a single cohesive framework, comparing and contrasting to explain the premises behind each, while reviewing current state-of-the-art advances and implementations.

Index Terms—Deep Learning, Generative Models, Energy-Based Models, Variational Autoencoders, Generative Adversarial Networks, Autoregressive Models, Normalizing Flows

1 Introduction

Generative modelling using neural networks has its origins in the 1980s with aims to learn about data with no supervision, potentially providing benefits for standard classification tasks; collecting training data for unsupervised learning is naturally much lower effort and cheaper than collecting labelled data but there is considerable information still available making it clear that generative models can be beneficial for a wide variety of applications.

Beyond this, generative modelling has numerous direct applications including image synthesis: super-resolution, text-to-image and image-to-image conversion, inpainting, attribute manipulation, pose estimation; video: synthesis and retargeting; audio: speech and music synthesis; text: summarisation and translation; reinforcement learning; computer graphics: rendering, texture generation, character movement, liquid simulation; medical: drug synthesis, modality conversion; and out-of-distribution detection.

The central idea of generative modelling stems around training a generative model whose samples $\tilde{\boldsymbol{x}} \sim p_\theta(\tilde{\boldsymbol{x}})$ come from the same distribution as the training data distribution, $\boldsymbol{x} \sim p_d(\boldsymbol{x})$. Early neural generative models, energy-based models achieved this by defining an energy function on data points proportional to likelihood, however, these struggled to scale to complex high dimensional data such as natural images, and require Markov Chain Monte Carlo (MCMC) sampling during both training and inference, a slow iterative process. In recent years there has been renewed interest in generative models driven by the advent of large freely available datasets as well as advances in both general deep learning architectures and generative models, breaking new ground in terms of visual fidelity and sampling speed. In many cases, this has been achieved using latent variables $\boldsymbol{z}$ which are easy to sample from and/or calculate the density of, instead learning $p(\boldsymbol{x}, \boldsymbol{z})$; this requires marginalisation over the unobserved latent variables, however in general, this is intractable. Generative models therefore typically make trade-offs in execution time, architecture, or optimise proxy functions. Choosing what to optimise for has implications for sample quality, with direct likelihood optimisation often leading to worse sample quality than alternatives.

There exists a variety of survey papers focusing on particular generative models such as normalizing flows [113], [164], generative adversarial networks [64], [230], and energy-based models [189], however, naturally these dive into the intricacies of their respective method rather than comparing with other methods; additionally, some focus on applications rather than theory. While there exists a recent survey on generative models as a whole [162], it is less broad, diving deeply into a few specific implementations.

This survey provides a comprehensive overview of generative modelling trends, introducing new readers to the field by bringing methods under a single cohesive statistical framework, comparing and contrasting so as to explain the modelling decisions behind each respective technique. Additionally, advances old and new are discussed in order to bring the reader up to date with current research. In particular, this survey covers energy-based models (Section 2), typically single unnormalised density models, variational autoencoders (Section 3), variational approximation of a latent-based model's posterior, generative adversarial networks (Section 4), two models set in a mini-max game, autoregressive models (Section 5), model data decomposed as a product of conditional probabilities, and normalizing flows (Section 6), exact likelihood models using invertible transformations. This breakdown is defined to closely match the typical divisions within research, however, numerous hybrid approaches exist that blur these lines, these are dis-

• *The authors are with the Department of Computer Science, Durham University, Durham, DH1 3LE, United Kingdom.*

TABLE 1: Comparison between deep generative models in terms of training and test speed, parameter efficiency, sample quality, sample diversity, and ability to scale to high resolution data. Quantitative evaluation is reported on the CIFAR-10 dataset [114] in terms of Fréchet Inception Distance (FID) and negative log-likelihood (NLL) in bits-per-dimension (BPD).

Method	Train Speed	Sample Speed	Param. Effic.	Sample Quality	Relative Divers.	Resolution Scaling	FID	NLL (in BPD)
Generative Adversarial Networks								
DCGAN [169]	★★★★★	★★★★★	★★★★☆	★★★☆☆	★☆☆☆☆	★★☆☆☆	17.70	-
ProGAN [102]	★★☆☆☆	★★★★★	★★★★☆	★★★☆☆	★★☆☆☆	★★★★★	15.52	-
BigGAN [17]	★★★★☆	★★★★★	★★★★☆	★★★★☆	★☆☆☆☆	★★★★☆	14.73	-
StyleGAN2 + ADA [103]	★★★☆☆	★★★★★	★★★★☆	★★★★☆	★★☆☆☆	★★★★★	2.42	-
Energy Based Models								
IGEBM [42]	★★★☆☆	★★☆☆☆	★★★★★	★★★☆☆	★★★☆☆	★★★☆☆	37.9	-
Denoising Diffusion [80]	★★☆☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★★★★☆	★★★★☆	3.17	≤ 3.75
DDPM++ Continuous [191]	★★☆☆☆	★★★☆☆	★★★☆☆	★★★★☆	★★★★☆	★★★★☆	2.92	2.99
Flow Contrastive [51]	★☆☆☆☆	★★★☆☆	★☆☆☆☆	★★☆☆☆	★★★☆☆	★★☆☆☆	37.30	≈ 3.27
VAEBM [226]	★★☆☆☆	★★★☆☆	★★★☆☆	★★★☆☆	★★★☆☆	★★★★☆	12.19	-
Variational Autoencoders								
Convolutional VAE [110]	★★★★★	★★★★★	★★★★☆	★★☆☆☆	★★★★☆	★★☆☆☆	106.37	≤ 4.54
Variational Lossy AE [27]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★☆	★☆☆☆☆	-	≤ 2.95
VQ-VAE [171], [215]	★★☆☆☆	★★★☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★★★★★	-	≤ 4.67
VD-VAE [29]	★★★☆☆	★★★★☆	★★★☆☆	★★★☆☆	★★★★☆	★★★★★	-	≤ 2.87
Autoregressive Models								
PixelRNN [214]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★★★★	★★★☆☆	-	3.00
Gated PixelCNN [213]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★★	★★★☆☆	65.93	3.03
PixelIQN [161]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★★	★★★☆☆	49.46	-
Sparse Trans. + DistAug [30], [99]	★★☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★☆☆	14.74	2.66
Normalizing Flows								
RealNVP [39]	★☆☆☆☆	★★★★☆	★☆☆☆☆	★☆☆☆☆	★★★★★	★★★☆☆	-	3.49
Masked Autoregressive Flow [165]	★★☆☆☆	★☆☆☆☆	★★☆☆☆	★☆☆☆☆	★★★☆☆	★★☆☆☆	-	4.30
GLOW [111]	★☆☆☆☆	★★★★☆	★☆☆☆☆	★★☆☆☆	★★★★★	★★★★☆	45.99	3.35
FFJORD [56]	★☆☆☆☆	★★★☆☆	★★★★★	★★☆☆☆	★★★★★	★★☆☆☆	-	3.40
Residual Flow [24]	★★☆☆☆	★★★★☆	★★★★☆	★★☆☆☆	★★★★★	★★★★☆	46.37	3.28

cussed in the most relevant section or both where suitable.

For a brief insight into the differences between different architectures, we provide Table 1 which contrasts a diverse array of techniques through easily comparable star ratings. Specifically, training speed is assessed based on reported total training times thus taking into account a variety of factors including architecture, number of function evaluations per step, ease of optimisation, and stochasticity involved; sample speed is based on network speed and number of evaluations required; parameter efficiency is determined by the total number of parameters required respective to the dataset trained on, while more powerful models will often have more parameters, the correlation with quality is not strong across model types; sample quality is determined using the following rules - 1 star: some structure/texture is captured, 2 stars: a scene is recognisable but global structure/detail is lacking, 3 stars: significant structure is captured but scenes look 'weird', 4 stars: difference to real images is identifiable, and 5 stars: difference is completely imperceptible; diversity is based on mode coverage/likelihood estimates of similar models; and resolution scaling is determined by the maximum reported resolutions.

2 Energy-Based Models

Energy-based models (EBMs) [119] are based on the observation that any probability density function $p(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^D$ can be expressed as

$$p(\boldsymbol{x}) = \frac{e^{-E(\boldsymbol{x})}}{\int_{\tilde{\boldsymbol{x}} \in \mathcal{X}} e^{-E(\tilde{\boldsymbol{x}})}}, \tag{1}$$

where $E(\boldsymbol{x})\colon \mathbb{R}^D \to \mathbb{R}$ is an energy function which associates realistic points with low values and unrealistic points with high values. Modelling data in such a way offers a number of perks, namely the simplicity and stability associating with training a single model; utilising a shared set of features thereby minimising required parameters; and the lack of any prior assumptions eliminates related bottlenecks [42]. Despite these benefits, EBM adoption fell in favour of implicit generative approaches due in part to poor scaling to high dimensional data, however, interest is increasing thanks to a number of tricks aiding better scaling.

A key issue with EBMs is how to optimise them; since the denominator in Equation 1 is intractable for most models, a popular proxy objective is contrastive divergence where energy values of data samples are 'pushed' down, while samples from the energy distribution are 'pushed' up. More formally, the gradient of the negative log-likelihood loss $\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x} \sim p_d}[-\ln p_\theta(\boldsymbol{x})]$ has been shown to approximately demonstrate the following property [21], [193],

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\boldsymbol{x}^+ \sim p_d}[\nabla_\theta E_\theta(\boldsymbol{x}^+)] - \mathbb{E}_{\boldsymbol{x}^- \sim p_\theta}[\nabla_\theta E_\theta(\boldsymbol{x}^-)], \tag{2}$$

where $\boldsymbol{x}^- \sim p_\theta$ is a sample from the EBM found through a Markov Chain Monte Carlo (MCMC) generating procedure.

2.1 Boltzmann Machines

A Boltzmann machine [76] is a fully connected undirected network of binary neurons that are turned on with probability determined by a weighted sum of their inputs i.e. for some state s_i, $p(s_i = 1) = \sigma(\sum_j w_{i,j} s_j)$. The neurons can be

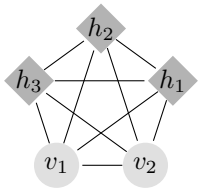


Fig. 1: A Boltzmann machine with 2 visible units and 3 hidden units.

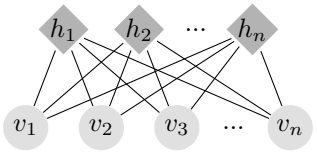


Fig. 2: A restricted Boltzmann machine has its architecture restricted so that there are no connections between units with each group.

divided into visible $\boldsymbol{v} \in \{0,1\}^D$ units, those which are set by inputs to the model, and hidden $\boldsymbol{h} \in \{0,1\}^P$ units, all other neurons. The energy of the state $\{\boldsymbol{v}, \boldsymbol{h}\}$ is defined (without biases for succinctness but can be generalised trivially) as

$$E_\theta(\boldsymbol{v}, \boldsymbol{h}) = -\frac{1}{2}\boldsymbol{v}^T \boldsymbol{L}\boldsymbol{v} - \frac{1}{2}\boldsymbol{h}^T \boldsymbol{J}\boldsymbol{h} - \frac{1}{2}\boldsymbol{v}^T \boldsymbol{W}\boldsymbol{h}, \tag{3}$$

where $\boldsymbol{W}$, $\boldsymbol{L}$, and $\boldsymbol{J}$ are symmetrical learned weight matrices. Due to the binary nature of the neurons, the probability assigned to a visible vector $\boldsymbol{v}$ as defined in Equation 1 can be reduced to finite summations

$$p_\theta(\boldsymbol{v}) = \frac{\sum_{\boldsymbol{h}} e^{-\beta E_\theta(\boldsymbol{v}, \boldsymbol{h})}}{\sum_{\tilde{\boldsymbol{v}}} \sum_{\boldsymbol{h}} e^{-\beta E_\theta(\tilde{\boldsymbol{v}}, \boldsymbol{h})}}. \tag{4}$$

Boltzmann machines are typically trained via negative log-likelihood through contrastive divergence, the architecture simplifying Equation 2 to vector products, which for $\boldsymbol{W}$ is

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \frac{\partial \ln p(\boldsymbol{x})}{\partial w_{i,j}} = \mathbb{E}_{p_d}[\boldsymbol{v}\boldsymbol{h}^T] - \mathbb{E}_{p_\theta}[\boldsymbol{v}\boldsymbol{h}^T]. \tag{5}$$

Exact computation of the expectations takes an exponential amount of time in the number of hidden units, making exact maximum likelihood intractable. Instead, training iterates between a positive training phase where visible units are fixed to a data vector and Gibbs sampling is used to approximate $\boldsymbol{h}$, and a negative training phase where no units are fixed and Gibbs sampling is used to approximate all units. The time to approach equilibrium increases significantly in Boltzmann machines with many hidden units, making Boltzmann machines only practical for low dimensional data. Additionally, for contrastive divergence to perform well, exact samples from $p(\boldsymbol{h}|\boldsymbol{v}; \theta)$ are required, further making Boltzmann machines impractical [77], [223].

2.2 Restricted Boltzmann Machines

Many of the issues associated with Boltzmann machines can be overcome by restricting their connectivity. One approach, known as the restricted Boltzmann machine (RBM) [77] is to set both $\boldsymbol{J} = \boldsymbol{0}$ and $\boldsymbol{L} = \boldsymbol{0}$. Because there are no direct connections between hidden units, it is possible to perform exact inference and thereby easily obtain an unbiased sample of $\mathbb{E}_{p_d}[\boldsymbol{v}\boldsymbol{h}^T]$; for a training sample $\boldsymbol{v} \in \mathcal{X}$ or hidden vector $\boldsymbol{h}$, the binary states h_j are activated with probability

$$p(h_j = 1|\boldsymbol{v}) = \sigma(\sum_i v_i w_{ij}), \tag{6a}$$

$$p(v_i = 1|\boldsymbol{h}) = \sigma(\sum_j h_j w_{ij}). \tag{6b}$$

Obtaining an unbiased sample of $\mathbb{E}_{p_{model}}[\boldsymbol{v}\boldsymbol{h}^T]$, however, is more difficult. Gibbs sampling starting with an initial visible state $\boldsymbol{v}$ is simplified through the iterative updating of $\boldsymbol{h}$ and $\boldsymbol{v}$ according to Equations 6a and 6b

$$h_j^{(n+1)} \sim \sigma(\sum_i v_i^{(n)} w_{ij}), \tag{7a}$$

$$v_i^{(n+1)} \sim \sigma(\sum_j h_j^{(n+1)} w_{ij}). \tag{7b}$$

Since both can be computed in parallel, this allows for considerable speedup. In practice, if $\boldsymbol{v}$ is initialised to be a sample from the dataset, a single step is sufficient [77]. Taking p_{recon} to be the distribution of visible units after a single step of Gibbs sampling, derivatives can be approximated as

$$\sum_{\boldsymbol{x} \in p_d} \frac{\partial \ln p(\boldsymbol{x})}{\partial w_{i,j}} \approx \mathbb{E}_{\mathcal{X}}[\boldsymbol{v}\boldsymbol{h}^T] - \mathbb{E}_{p_{recon}}[\boldsymbol{v}\boldsymbol{h}^T]. \tag{8}$$

2.3 Deep Belief Networks

An efficient way to learn powerful functions is to combine simpler functions; it is natural therefore to attempt to stack RBMs, using features from lower down as inputs for the next layer. Training an entire model of this form concurrently, however, is impossible since NLL calculations are intractable. Instead, deep belief networks train RBMs layer by layer in a greedy fashion, composing probability densities [78]. A RBM finds it easy to learn weights that convert the posterior distribution over the hidden units into the data distribution, $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{W})$, however, modelling the posterior distribution over the hidden units $p(\boldsymbol{h}|\boldsymbol{W})$ is more difficult. Nevertheless, training a second RBM to model the posterior of the first is then an easier task because the distribution is closer to one that it can model perfectly. Thus by improving $p(\boldsymbol{h})$, $p(\boldsymbol{v})$ is also improved; each time another layer of features is added, the variational lower bound on the log probability is improved. Sampling data from a deep belief network is much slower than from a RBM; first, an equilibrium sample from the top level RBM is found by performing Gibbs sampling for a long time, then a top-down pass is performed to obtain states for all the other layers.

2.4 Deep EBMs via Contrastive Divergence

To train more powerful architectures through contrastive divergence, one must be able to efficiently sample from p_θ. Specifically, we would like to model an energy function with a deep neural network, taking advantage of recent advances in discriminative models [232] applied to high dimensional data. MCMC methods such as random walk and Gibbs sampling [78], when applied to high dimensional data, have long mixing times, making them impractical.

A number of recent approaches [42], [58], [152], [153], [228] have advocated the use of stochastic gradient

Langevin dynamics [174], [224] which permits sampling through the following iterative process,

$$\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}) \qquad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \frac{\alpha}{2}\frac{\partial E_\theta(\boldsymbol{x}_i)}{\partial \boldsymbol{x}_i} + \boldsymbol{\epsilon}, \tag{9}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha \boldsymbol{I})$, $p_0(\boldsymbol{x})$ is typically a uniform distribution over the input domain and α is the step size. As the number of updates $N \to \infty$ and $\alpha \to 0$, the distribution of samples converges to p_θ [224]; however, α and $\boldsymbol{\epsilon}$ are often tweaked independently to speed up training.

While it is possible to train with noise initialisation as above using as few as 100 update steps, known as short-run MCMC, by training a non-convergent EBM [152], this number of steps is still prohibitively slow for large scale tasks [58]. Additionally, because the number of update steps is so small, samples are not truly from the correct probability density; there are a number of advantages however, such as allowing image interpolation and reconstruction (since short-run MCMC does not mix) [153]. One solution is to select samples from a large replay buffer which stores previously generated samples, and with some probability resets a sample to noise [42], [228]. This allows samples to be continually refined with a relatively small number of Langevin update steps while maintaining diversity. Faster training comes at the expense of slower inference times, however, as well as worse diversity [153]. Other approaches include initialising MCMC sampling chains with data points [228] and samples from an implicit generative model [227], as well as adversarially training an implicit generative model, mitigating mode collapse somewhat by maximising its entropy [59], [108], [117]. Improved/augmented MCMC samplers with neural networks can also improve the efficiency of MCMC sampling [82], [122], [186], [200].

One application of EBMs of this form comes by using standard classifier architectures, $f_\theta \colon \mathbb{R}^D \to \mathbb{R}^K$, which map data points to logits used by a softmax function,

$$p_\theta(y|\boldsymbol{x}) = \frac{\exp(f_\theta(\boldsymbol{x})[y])}{\sum_{\tilde{y}} \exp(f_\theta(\boldsymbol{x})[\tilde{y}])}, \tag{10}$$

which assigns the probability of $\boldsymbol{x}$ belonging to class y. By marginalising out y, these logits can be used to define an unnormalised density model and thus an energy model [58],

$$p_\theta(\boldsymbol{x}) = \sum_y p_\theta(\boldsymbol{x}, y) = \frac{\sum_y \exp(f_\theta(\boldsymbol{x}[y]))}{Z(\theta)}, \tag{11a}$$

$$E_\theta(\boldsymbol{x}) = -\ln \sum_y \exp(f_\theta(\boldsymbol{x}[y])). \tag{11b}$$

As such, it is possible to train a classifier as both a generative a competitive classification model.

2.5 Deep EBMs via Score Matching

Although Langevin sampling has allowed EBMs to scale to high dimensional data, training times are still slow due to the need to sample from the model distribution when using contrastive divergence. This is made worse by the finite nature of the sampling process, meaning that samples can be arbitrarily far away from the model's distribution [57]. An alternative approach is score matching [93] which is based on the idea of minimising the difference between the derivatives of the data and model's log-density functions; the score function is defined as $s(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \ln p(\boldsymbol{x})$ which does not depend on the intractable denominator and can therefore be applied to build an energy model [121], [194] by minimising the Fisher divergence between p_d and p_θ,

$$\mathcal{L} = \frac{1}{2}\mathbb{E}_{p_d(\boldsymbol{x})}[\, \|s_\theta(\boldsymbol{x}) - s_d(\boldsymbol{x})\|_2^2], \tag{12}$$

however, the score function of data is usually not available. There exist various methods to estimate the score function including spectral approximation [181], sliced score matching [188], finite difference score matching [163], and notably denoising score matching [219] which allows the score to be approximated using corrupted data samples $q(\tilde{\boldsymbol{x}}|\boldsymbol{x})$. In particular, when $q = \mathcal{N}(\tilde{\boldsymbol{x}}|\boldsymbol{x}, \sigma^2 \boldsymbol{I})$, Equation 12 simplifies to

$$\mathcal{L} = \frac{1}{2}\mathbb{E}_{p_d(\boldsymbol{x})}\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathcal{N}(\boldsymbol{x},\sigma^2\boldsymbol{I})}\left[\left\| s_\theta(\tilde{\boldsymbol{x}}) + \frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\sigma^2}\right\|_2^2\right]. \tag{13}$$

That is, s_θ learns to estimate the noise thereby allowing it to be used as a generative model [178], [187]. Notice in Equation 9 that the Langevin update step uses $\nabla_{\boldsymbol{x}} \ln p(\boldsymbol{x})$, or s_θ, thus it is possible to sample from a score matching model using Langevin dynamics [210]. This is only possible, however, when trained over a large variety of noise levels so that $\tilde{\boldsymbol{x}}$ covers the whole space.

This process is very similar to diffusion models [1], [10], [80], [184] which undergo a similar procedure using a predetermined variance schedule over a fixed number of steps, allowing a bound on $p(\boldsymbol{x})$ to be derived. Diffusion models applied to high dimensional data have demonstrated the ability to generate state-of-the-art samples [80], [148], [191].

Similar to implicit energy models, sampling from score matching models requires a large number of steps. By modelling the Langevin sampling procedure as a stochastic differential equation (SDE), pre-existing SDE solvers can be used, reducing the number of steps required [191], [209]. Another proposed approach is to model noisy data points as $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$, allowing the generative process to skip some steps using its approximation of end samples $\boldsymbol{x}_0$ [185].

2.6 Correcting Implicit Generative Models

While EBMs offer powerful representation ability due to unnormalized likelihoods, they can suffer from high variance training, long training and sampling times, and struggle to support the entire data space. In this section, a number of hybrid approaches are discussed which address these issues.

2.6.1 Exponential Tilting

To eliminate the need for an EBM to support the entire space, an EBM can instead be used to correct samples from an implicit generative network, simplifying the function to learn and allowing easier sampling. This procedure, referred to as exponentially tilting an implicit model, is defined as

$$p_{\theta,\phi}(\boldsymbol{x}) = \frac{1}{Z_{\theta,\phi}} q_\phi(\boldsymbol{x}) e^{-E_\theta(\boldsymbol{x})}. \tag{14}$$

By parameterising $q_\phi(\boldsymbol{x})$ as a latent variable model such as a normalizing flow [2], [151] or VAE generator [226], MCMC sampling can be performed in the latent space rather than the data space. Since the latent space is much simpler, and

often uni-modal, MCMC mixes much more effectively. This limits the freedom of the model, however, leading some to jointly sample in latent and data space [2], [226].

2.6.2 Noise Contrastive Estimation

Noise contrastive estimation [48], [68] transforms EBM training into a classification problem using a noise distribution $q_\phi(\boldsymbol{x})$ by optimising the loss function,

$$\mathbb{E}_{p_d}\left[\ln \frac{p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{x})+q_\phi(\boldsymbol{x})}\right]+\mathbb{E}_{q_\phi}\left[\ln \frac{q_\phi(\boldsymbol{x})}{p_\theta(\boldsymbol{x})+q_\phi(\boldsymbol{x})}\right], \quad (15)$$

where $p_\theta(\boldsymbol{x})=e^{E_\theta(\boldsymbol{x})-c}$. This approach can be used to train a correction via exponential tilting [151], but can also be used to directly train an EBM and normalizing flow [51]. Equation 15 is equivalent to GAN Equation 23, however, training formulations differ, with noise contrastive estimation explicitly modelling likelihood ratios.

2.7 Alternative Training Objectives

As aforementioned, energy models trained with contrastive divergence approximately maximises the likelihood of the data; likelihood however does not correlate directly with sample quality [199]. Training EBMs with arbitrary f-divergences is possible, yielding improved FID scores [231].

Since score estimates have high variance, the Stein discrepancy has been proposed as an alternative objective, requiring no sampling and more closely correlating with likelihood [57]. A middle ground between denoising score matching and contrastive divergence is diffusion recovery likelihood [11] which can be optimised via a sequence of denoising EBMs conditioned on increasingly noisy samples of the data, the conditional distributions being much easy to MCMC sample from than typical EBMs [52].

3 Variational Autoencoders

One of the key problems associated with energy-based models is that sampling is not straightforward and mixing can require a significant amount of time. To circumvent this issue, it would be beneficial to explicitly sample from the data distribution with a single network pass.

To this end, suppose we have a latent based model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with prior $p_\theta(\boldsymbol{z})$ and posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$; unfortunately optimising this model through max likelihood is intractable due to the integral in $p_\theta(\boldsymbol{x})=\int_{\boldsymbol{z}} p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z})d\boldsymbol{z}$. Instead, [110] propose learning a second function $q_\phi(\boldsymbol{z}|\boldsymbol{x})=\arg\min_q D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x}))$ which approximates the true intractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. From the definition of KL divergence we get

$$\begin{aligned} D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln q_\phi(\boldsymbol{z}|\boldsymbol{x})]-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{z},\boldsymbol{x})] \\ &\quad +\ln p_\theta(\boldsymbol{x}), \end{aligned} \quad (16)$$

which can be rearranged to find an alternative definition for $p_\theta(\boldsymbol{x})$ that does not require the knowledge of $p_\theta(\boldsymbol{z}|\boldsymbol{x})$

$$\begin{aligned} \ln p_\theta(\boldsymbol{x}) &= D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x}))-\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln q_\phi(\boldsymbol{z}|\boldsymbol{x})] \\ &\quad +\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{z},\boldsymbol{x})] \\ &\geq -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln q_\phi(\boldsymbol{z}|\boldsymbol{x})]+\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{z},\boldsymbol{x})] \\ &= -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln q_\phi(\boldsymbol{z}|\boldsymbol{x})]+\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{z})] \\ &\quad +\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] \\ &= -D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}))+\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] \\ &\equiv \mathcal{L}(\theta,\phi;\boldsymbol{x}), \end{aligned} \quad (17)$$

where $\mathcal{L}$ is known as the evidence lower bound (ELBO) [98]. To optimise this bound with respect to θ and ϕ, gradients must be backpropagated through the stochastic process of generating samples from $\tilde{\boldsymbol{z}}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$. This is permitted by reparameterizing $\tilde{\boldsymbol{z}}$ using a differentiable function $g_\phi(\boldsymbol{\epsilon},\boldsymbol{x})$ of a noise variable $\boldsymbol{\epsilon}$: $\tilde{\boldsymbol{z}}=g_\phi(\boldsymbol{\epsilon},\boldsymbol{x})$ with $\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})$ [110].

Monte Carlo gradient estimators can be used to approximate the expectations, however this yields very high variance making it impractical. However, if $D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}))$ can be integrated analytically then the variance is manageable. A prior with such a property needs to be simple enough to sample from but also sufficiently flexible to match the true posterior; a common choice is a normally distributed prior with diagonal covariance, $\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})=\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I})$ with $\tilde{\boldsymbol{z}}=\boldsymbol{\mu}+\boldsymbol{\sigma}\odot\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. In this case, the loss simplifies to

$$\begin{aligned} \tilde{\mathcal{L}}_{VAE}(\theta,\phi;\boldsymbol{x}) &\simeq \frac{1}{2}\sum_{j=1}^{J}\left(1+\ln((\sigma^{(j)})^2)-(\mu^{(j)})^2-(\sigma^{(j)})^2\right) \\ &\quad +\frac{1}{L}\sum_{l=1}^{L}\ln p_\theta(\boldsymbol{x}|\tilde{\boldsymbol{z}}_l). \end{aligned} \quad (18)$$

Despite success on small scale datasets, when applied to more complex datasets such as natural images, samples tend to be unrealistic and blurry [41]. This blurriness has been attributed to the max likelihood objective itself and MSE reconstruction loss, however, there is evidence that limited approximation of the true posterior is the root cause [239]; with MSE causing highly non-gaussian posteriors. As such, the gaussian posterior implies an overly simple model which, when unable to perfectly fit, maps multiple data points to the same encoding leading to averaging.

There are a number of other issues associated with limited posterior approximation, namely under-estimation of the variance of the posterior, resulting in poor predictions, and biases in the MAP estimates of model parameters [207]. Quality can be improved using higher capacity functions, however these can be harder to balance, leading to posterior collapse [73], [239], as well as by combining with adversarial training [89], [118]. There have also been attempts to improve the variational bound itself [19], as well as use different regularisation techniques such as Wasserstein distance [201] and GANs [136].

3.1 Beyond Simple Priors

One approach to improve variational bounds and increase sample quality is to improve the priors used for instance by careful selection to the task or by increasing its complexity [83]. Complex priors can be learned by warping simple distributions and inducing variational dependencies between the latent variables: variational Gaussian processes permit this by forming an infinite ensemble of mean-field distributions [203]; an EBM can be used to model a flexible prior [163]; normalizing flows (see Section 6) transform

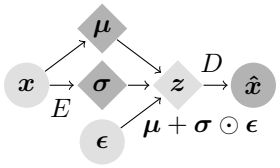


Fig. 3: Variational autoencoder with a normally distributed prior. $\boldsymbol{\epsilon}$ is sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

distributions through a series of invertible parameterised functions [12], [56], [88], [112], [173], [177].

Alternatively, by rewriting the VAE training objective to have two regularisation terms [136],

$$\begin{aligned}\mathcal{L}(\theta, \phi; \boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})]] \\ + \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}[\mathbb{H}[q_\phi(\boldsymbol{z}|\boldsymbol{x})]] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[p_\theta(\boldsymbol{z})],\end{aligned} \tag{19}$$

the latter of which is the cross entropy between the aggregate posterior and the prior, the prior can be defined as the aggregate posterior, thus obtaining a rich multi-modal latent representation that combats inactive latent variables. Since the true aggregate posterior is intractable, VampPrior [202] approximates it for a set of pseudo-inputs, tensors with the same shape as data points learned during training. Exemplar VAEs [155] scale this approach up, using the full training set to approximate the aggregate posterior, by approximating the prior using k-nearest-neighbours. Alternatively, the aggregate posterior can be approximated with a learned prior; this has been achieved with a learned rejection sampling procedure that transforms a base distribution [6].

3.1.1 2-Stage VAEs

By interpreting VAEs as regularised autoencoders, it is natural to apply less strict regularisation to the latent space during training then subsequently train a density estimator on this space, thus obtaining a more complex prior [53]. Vector Quantized-Variational Autoencoders (VQ-VAE) [170], [215] achieve this by training an autoencoder with a discrete latent space, then training a powerful autoregressive model (see Section 5) on the latent space. This is achieved by defining a codebook of latent vectors; the encoder's outputs are compared to these codes and set to the code they are closest to. Since there is no gradient defined for this discretisation process, gradients are passed 'straight through' from the input to the decoder to the output of the encoder [9], allowing useful information to be passed through. Meanwhile, latent vectors in the codebook are moved closer to the encoder's outputs. This approach has been extended to use a hierarchy of codes, allowing much larger images to be modelled [171]. When data of dimension d lies on a sub-manifold of dimension r and $r < d$ then global VAE optimums exist that do not recover the data distribution, however, when $r = d$, global optimums do recover the data distribution; as such, 2 stage VAEs that first map data to latents of dimension r then use a second VAE to correct the learned density can better capture the data [35].

3.1.2 Hierarchical VAEs

Hierarchical VAEs build complex priors with multiple levels of latent variables, each conditionally dependent on the last,

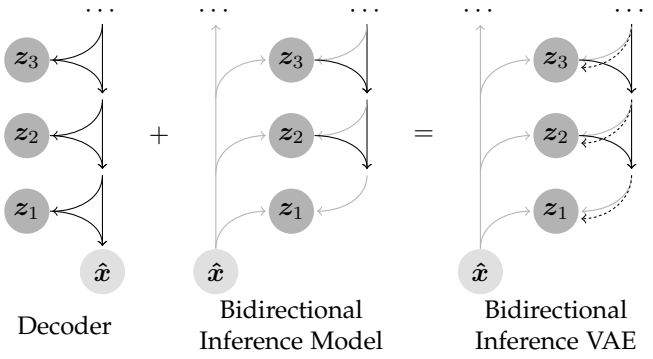


Fig. 4: A hierarchical VAE with bidirectional inference [112].

forming dependencies depthwise though the network,

$$p_\theta(\boldsymbol{z}) = p_\theta(\boldsymbol{z}_0)p_\theta(\boldsymbol{z}_1|\boldsymbol{z}_0)\cdots p_\theta(\boldsymbol{z}_N|\boldsymbol{z}_{<N}), \tag{20a}$$

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}_0|\boldsymbol{x})q_\phi(\boldsymbol{z}_1|\boldsymbol{z}_0, \boldsymbol{x})\cdots q_\phi(\boldsymbol{z}_N|\boldsymbol{z}_{<N}, \boldsymbol{x}). \tag{20b}$$

Ladder VAEs [196] achieve this conditioning structure using a bidirectional inference network where a deterministic "bottom-up" pass generates features at various resolutions, then the latent variables are processed from top to bottom with the features shared (Fig. 4). Specifically, they model latents as normal distributions conditioned on the last latent,

$$p_\theta(\boldsymbol{z}_i|\boldsymbol{z}_{i-1}) = \mathcal{N}(\boldsymbol{z}_i|\mu_{p,i}(\boldsymbol{z}_{i-1}), \sigma^2_{p,i}(\boldsymbol{z}_{i-1})). \tag{21}$$

By introducing skip connections around the stochastic sampling process, latents can be conditioned on all previously sampled latents [112], [134], [211]. Such an architecture is a generalisation of autoregressive models; inferring latents in parallel allows for significantly fewer steps compared to typical autoregressive models since many latents are statistically independent and allows different latent levels to correspond to global/local details depending on their depth. It has been argued that a single level of latents is sufficient since Gibbs sampling performed on that level can recover the data distribution [238]. Despite that, Gibbs sampling can take a long time to converge, making hierarchical representations more efficient; in support of this, deeper hierarchical VAEs have been shown to improve log-likelihood, independent of capacity [29].

3.2 Data Modelling Distributions

Unlike energy-based models, VAEs must model an explicit density $p(\boldsymbol{x}|\boldsymbol{z})$. For efficient sampling, typically this distribution is decomposed as a product of independent simple distributions, allowing unrestricted architectures to be used to parameterise the chosen distributions. Common instances include modelling variables as Bernoulli [128], Gaussian [110], multinomial distributions, or as mixtures [176].

3.2.1 Autoregressive Decoders

To introduce dependencies between the output variables, numerous works have used powerful autoregressive networks [66]. While these approaches allow complex distributions to be learned, they increase the runtime and can suffer from posterior collapse since powerful autoregressive models are capable of modelling the data distribution independent of any conditioning [16]. Various methods to prevent

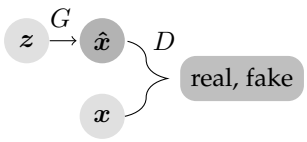


Fig. 5: Generative adversarial networks set two networks in a game: D detects real from fake samples while G tricks D.

posterior collapse have been proposed: by restricting the autoregressive network's receptive field to a small window, it is forced to use latents to capture global structure [27]; a mutual information term can be added to the loss to encourage high correlation between $\boldsymbol{x}$ and $\boldsymbol{z}$ [240]; encouraging the posterior to be diverse by controlling its geometry to evenly covering the data space, redundancy is reduced and latents are encouraged to learn global structure [133].

3.3 Bridging Variational and MCMC Inference

While variational approaches offer substantial speedup over MCMC sampling, there is an inherent discrepancy between the true posterior and approximate posterior despite improvements in this field. To this end, a number of approaches have been proposed to find a middle ground, yielding improvements over variational methods with lower costs than MCMC. Semi-amortised VAEs [109] use an encoder network followed by stochastic gradient descent on latents to improve the ELBO, however, this still relies on an inference network. The inference network can be removed by assigning latent vectors to data points, then optimising them with Langevin dynamics or gradient descent, during training; although this allows fast training, convergence for unseen samples is not guaranteed and there is still a large discrepancy between the true posterior and latent approximations due to lag in optimisation [14], [71]. Short-run MCMC has also been applied however it has poor mixing properties [154]. Gradient Origin Networks [15] replace the encoder with an empirical Bayes approximation of the posterior that only requires a single gradient step.

VAEBMs offer a different perspective, rather than performing latent MCMC sampling based on the ELBO, they use an auxiliary energy-based model to correct blurry VAE samples, with MCMC sampling performed in both the data space and latent space. This setup is defined by $h_{\phi,\theta}(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{Z_{\phi,\theta}} p_\theta(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z}) e^{-E_\phi(\boldsymbol{x})}$, where $p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the VAE, and $E_\phi(\boldsymbol{x})$ is the energy model. This, however, requires 2 stages of training to avoid calculating the gradient of the normalising constant $Z_{\phi,\theta}$, training only the VAE and fixing the VAE and training the EBM respectively.

4 Generative Adversarial Networks

Another approach at eliminating the Markov chains used in energy models is the generative adversarial network (GAN) [54]. GANs consist of two networks, a discriminator $D\colon \mathbb{R}^n \to [0,1]$ which estimates the probability that a sample comes from the data distribution $\boldsymbol{x} \sim p_d(\boldsymbol{x})$, and a generator $G\colon \mathbb{R}^m \to \mathbb{R}^n$ which given a latent variable $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$, captures p_d by tricking the discriminator into thinking its samples are real. This is achieved through adversarial training of the networks: D is trained to correctly label training samples as real and samples from G as fake, while G is trained to minimise the probability that D classes its samples as fake. This can be interpreted as D and G playing a mini-max game, as with prior work [179], [180], optimising the value function $V(G,D)$,

$$\begin{aligned}\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[\ln D(\boldsymbol{x})] \\ + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\ln(1 - D(G(\boldsymbol{z})))].\end{aligned} \tag{22}$$

For a fixed generator G, the training objective for D can be reformulated as

$$\begin{aligned}&\max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_d}[\ln D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\ln(1 - D(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_d}\left[\ln \frac{p_d(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \\ &\quad + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\ln \frac{p_g(\boldsymbol{x})}{p_d(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \\ &= D_{KL}(p_d||\tfrac{1}{2}(p_d + p_g)) + D_{KL}(p_g||\tfrac{1}{2}(p_d + p_g)) + C.\end{aligned} \tag{23}$$

Therefore the loss is equivalent to the Jensen-Shannon divergence between the generative distribution p_g and the data distribution p_d and thus with sufficient capacity, the generator can recover the data distribution. The use of symmetric JS-divergence is well behaved when both distributions are small unlike the asymmetric KL-divergence used in max-likelihood models. Additionally, it has been suggested that reverse KL-divergence, $D_{KL}(p_g||p_d)$, is a better measure for training generative models than normal KL-divergence, $D_{KL}(p_d||p_g)$, since it minimises $\mathbb{E}_{\boldsymbol{x}\sim p_g}[\ln p_d(\boldsymbol{x})]$ [92]; while reverse KL-divergence is not a viable objective function, JS-divergence is and behaves more like reverse KL-divergence than KL-divergence alone. With that said, JS-divergence is not perfect; if 0 mass is associated with a data sample in a max-likelihood model, KL-divergence is driven to infinity, whereas this can happen with no consequence in a GAN.

4.1 Stabilising Training

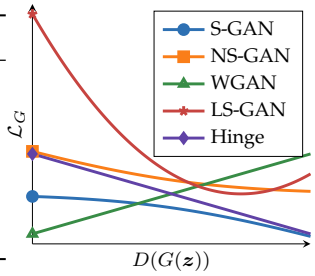
The adversarial nature of GANs makes them notoriously difficult to train [3]; Nash equilibrium is hard to achieve [175] since non-cooperation cannot guarantee convergence, thus training often results in oscillations of increasing amplitude. As the discriminator improves, gradients passed to the generator vanish, accelerating this problem; on the other hand, if the discriminator remains poor, the generator does not receive useful gradients. Another problem is mode collapse, where one network gets stuck in a bad local minima and only a small subset of the data distribution is learned. The discriminator can also jump between modes resulting in catastrophic forgetting, where previously learned knowledge is forgotten when learning something new [198]. This section explores proposed solutions to these problems.

4.1.1 Loss Functions

Since the cause of many of these issues can be linked with the use of JS-divergence, other loss functions have been proposed that minimise other statistical distances; in general, any f-divergence can be used to train GANs [156]. One notable example of such is the Wasserstein distance which intuitively indicates how much "mass" must be

Name	Discriminator Loss	Generator Loss
NSGAN [54]	$-\mathbb{E}[\ln(\sigma(D(\boldsymbol{x})))] - \mathbb{E}[\ln(1-\sigma(D(G(\boldsymbol{z}))))]$	$-\mathbb{E}[\ln(\sigma(D(G(\boldsymbol{z}))))]$
WGAN [4]	$\mathbb{E}[D(\boldsymbol{x})] - \mathbb{E}[D(G(\boldsymbol{z}))]$	$\mathbb{E}[D(G(\boldsymbol{z}))]$
LSGAN [137]	$\mathbb{E}[(D(\boldsymbol{x})-1)^2] + \mathbb{E}[D(G(\boldsymbol{z}))^2]$	$\mathbb{E}[(D(G(\boldsymbol{z}))-1)^2]$
Hinge [123]	$\mathbb{E}[\min(0, D(\boldsymbol{x})-1)] - \mathbb{E}[\max(0, 1+D(G(\boldsymbol{z})))]$	$-\mathbb{E}[D(G(\boldsymbol{z}))]$
EBGAN [237]	$D(\boldsymbol{x}) + \max(0, m - D(G(\boldsymbol{z})))$	$D(G(\boldsymbol{z}))$
RSGAN [97]	$\mathbb{E}[\ln(\sigma(D(\boldsymbol{x}) - D(G(\boldsymbol{z}))))]$	$\mathbb{E}[\ln(\sigma(-D(G(\boldsymbol{z})) - D(\boldsymbol{x})))]$

(a) GAN losses.



(b) Generator loss functions.

Fig. 6: A comparison of popular losses used to train GANs. (a) Respective losses for discriminator/generator. (b) Plots of generator losses with respect to discriminator output. Notably, NS-GAN's gradient disappears as discriminator gets better.

moved to transform one distribution into another. Wasserstein distance is defined formally in Equation 24a, which by the Kantorovich-Rubinstein duality can be shown to be equivalent to Equation 24b [218]:

$$W(p_d, p_g) = \inf_{\gamma \in \prod(p_d, p_g)} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}[\|\boldsymbol{x}-\boldsymbol{y}\|], \tag{24a}$$

$$W(p_d, p_g) = \sup_{\|D\|_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim p_d}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_g}[D(\boldsymbol{x})], \tag{24b}$$

where the supremum is taken over all 1-Lipschitz functions, that is, f such that for all x_1 and x_2, $\|f(x_1) - f(x_2)\|_2 \leq \|x_1 - x_2\|_2$. Optimising Wasserstein distance, as described in Table 6a, offers linear gradients thus eliminating the vanishing gradients problem (see Fig. 6b). Moreover, Wasserstein distance is also equivalent to minimising reverse KL-divergence [143], offers improved stability, and allows training to optimality. Numerous approaches to enforce 1-Lipschitz continuity have been proposed: weight clipping [4] invalidates gradients making optimisation difficult; applying a gradient penalty within the loss is heavily dependent on the support of the generative distribution and computation with finite samples makes application to the entire space intractable [65]; spectral normalisation (discussed below) applies global regularisation by estimating the singular values of parameters. Other popular loss functions include least squares GAN, hinge loss, energy-based GAN, and relativistic GAN (detailed in Table 6a).

The catastrophic forgetting problem can be mitigated by conditioning the GAN on class information, encouraging more stable representations [17], [142], [234]. Nevertheless, labelled data, if available, only covers limited abstractions. Self-supervision achieves the same goal by training the discriminator on an auxiliary classification task based solely on the unsupervised data. Proposed approaches are based on randomly rotating inputs to the discriminator, which learns to identify the angle rotated separately to the standard real/fake classification [26]. Extensions include training the discriminator to jointly determine rotation and real/fake to provide better feedback [206], and training the generator to trick the discriminator at both the real/fake and classification tasks [206]. A more explicit approach is to model the generator with a normalizing flow, avoiding collapse by jointly optimising the GAN and likelihood objectives [63].

4.1.2 Spectral Normalisation

Spectral normalisation [143] is a technique to make a function globally 1-Lipschitz utilising the observation that the Lipschitz constant of a linear function is its largest singular value (spectral norm). The spectral norm of a matrix $\boldsymbol{A}$ is

$$SN(\boldsymbol{A}) := \max_{\boldsymbol{h}:\boldsymbol{h}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2 \leq 1} \|\boldsymbol{A}\boldsymbol{h}\|_2\,, \tag{25}$$

thus a weight matrix $\boldsymbol{W}$ is normalised to be 1-Lipschitz by replacing the weights with $\boldsymbol{W}_{SN} := \frac{\boldsymbol{W}}{SN(\boldsymbol{W})}$. Rather than using singular value decomposition to compute the norm, the power iteration method is used; for randomly initialised vectors $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$, the procedure is

$$\boldsymbol{u}_{t+1} = \boldsymbol{W}\boldsymbol{v}_t,\ \boldsymbol{v}_{t+1} = \boldsymbol{W}^T\boldsymbol{u}_{t+1},\ SN(\boldsymbol{W}) \approx \boldsymbol{u}^T\boldsymbol{W}\boldsymbol{v}. \tag{26}$$

Since weights change only marginally with each optimisation step, a single power iteration step per global optimisation step is sufficient to keep $\boldsymbol{v}$ and $\boldsymbol{u}$ close to their targets.

As aforementioned, enforcing the discriminator to be 1-Lipschitz is essential for WGANs, however, spectral normalisation has been found to dramatically improve sample quality and allow scaling to datasets with thousands of classes across a variety of loss functions [17], [143]. Spectral collapse, has been linked to discriminator overfitting when spectral norms of layers explode [17] as well as mode collapse when spectral norms fall in value significantly [126]. Additionally, regularising the discriminator in this manner helps balance the two networks, reducing the number of discriminator update steps required [17], [234].

4.1.3 Data Augmentation

Augmenting training data to increase the quantity of training data is often common practice; when training GANs the types of augmentations permitted are limited to more simple augmentations such as cropping and flipping to prevent the generator from creating undesired artefacts. Several approaches independently proposed applying augmentations to all discriminator inputs, allowing more substantial augmentations to be used [103], [205], [241], [242]; the training procedure for a WGAN with augmentations is

$$\mathcal{L}_D = \mathbb{E}_{\boldsymbol{x}\sim p_d(\boldsymbol{x})}[D(T(\boldsymbol{x}))] - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[D(T(G(\boldsymbol{z})))], \tag{27a}$$

$$\mathcal{L}_G = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[D(T(G(\boldsymbol{z})))], \tag{27b}$$

where T is a random augmentation. These approaches have been shown to improve sample quality on equivalent architectures and stabilise training. Each work offers a different perspective on why augmentation is so effective: the increased quantity of training data in conjunction with

the more difficult discrimination task prevents overfitting and in turn collapse [17], notably this applies even on very small datasets (100 samples); the nature of GAN training leads to the generated and data distributions having non-overlapping supports, complicating training [195], strong augmentations may cause these distributions to overlap further. If an augmentation is differentiable and represents an invertible transformation of the data space's distribution, then the JS-divergence is invariant, and the generator is guaranteed to not create augmented samples [103], [205].

4.1.4 Discriminator Driven Sampling

In order to improve sample quality and address overpowered discriminators, numerous works have taken inspiration from the connection between GANs and energy models [237]. Interpreting the discriminator of a Wasserstein GAN [4] as an energy-based model means samples from the generator can be used to initialise an MCMC sampling chain which converges to the density learned by the discriminator, correcting errors learned by the generator [146], [208]. This is similar to pure EBM approaches, however, training the two networks adversarially changes the dynamics. The slow convergence rates of high dimensional MCMC sampling has led others to instead sample in the latent space [22], [192].

4.1.5 GANs without Competition

Originally proposed as a proxy to measure GAN convergence [62], the duality gap is an upper bound on the JS-divergence that can be directly optimised [61], defined as

$$DG(D,G) = \max_{D'} V(G,D') - \min_{G'} V(G',D). \tag{28}$$

Cooperative training simplifies the optimisation procedure, avoiding oscillations. Each training step, however, requires optimising for D' and G' which slows down training and could suffer from vanishing gradients.

4.2 Architectures

Careful network design is a key component for stable GAN training. Scaling any deep neural network to high-resolution data is non-trivial due to vanishing gradients and high memory usage, but since the discriminator can classify high-resolution data more easily, GANs notably struggle [157].

Early approaches designed hierarchical architectures, dividing the learning procedure into more easily learnable chunks. LapGAN [37] builds a Laplacian pyramid such that at each layer, a GAN conditioned on the previous image resolution predicts a residual adding detail. Stacked GANs [90], [235] use two GANs trained successively: the first generates low-resolution samples, then the second upsamples and corrects the first, thus fewer GANs need to be trained. A related approach, progressive growing [102], [104], iteratively trains a single GAN at higher resolutions by adding layers to both the generator and discriminator upscaling the previous output, after the previous resolution converges. Training in this manner, however, not only takes a long time but leads to high frequency components being learned in the lower layers, resulting in shift artefacts [105].

Accordingly, a number of works have targeted a single GAN that can be trained end-to-end. DCGAN [169] introduced a fully convolutional architecture with batch normalisation [94] and ReLU/LeakyReLU activations. BigGAN [17] employ a number of tricks to scale to high resolutions including using very large mini-batches to reduce variation, spectral normalisation to discourage spectral collapse, and using large datasets to prevent overfitting Despite this, training collapse still occurs thus requiring early stopping. Another approach is to include skip connections between the generator and discriminator at each resolution, allowing gradients to flow through shorter paths to each layer, providing extra information to the generator [101], [105], [212]. By treating subsets of the generator's parameters as smaller generators, Anycost GANs extend this approach, allowing samples to be generated at multiple resolutions and speeds [124]. To learn long-range dependencies, GANs can be built with self-attention components [96], [216], [234], however, full quadratic attention does not scale well to high dimensional data. Alternatively, a powerful learned discrete prior can be used to model very high resolution data [49].

4.3 Training Speed

The mini-max nature of GAN training leads to slow convergence, if achieved at all. This problem has been exacerbated by numerous works as a byproduct of improving stability or sample quality. One such example is that by using very large mini-batches, reducing variance and covering more modes, sample quality can be improved significantly, however, this comes at the cost of slower training [17]. Small-GAN [182] combats this by replacing large batches with small batches that approximate the shape of the larger batch using core set sampling [182], significantly improving the mode coverage and sample quality of GANs trained with small batches.

While strong discriminator regularisation stabilises training, it allows the generator to make small changes and trick the discriminator, making convergence very slow. RobGAN [127], include an adversarial attack step [135] that perturbs real images to trick the discriminator without altering the content inordinately, adapting the GAN objective into a min-max-min problem. This provides a weaker regularisation, enforcing small Lipschitz values locally rather than globally. This approach has been connected with the follow-the-ridge algorithm [221], [243], an optimisation approach for solving mini-max problems that reduces the optimisation path and converges to local mini-max points.

Another approach to improve training speed is to design more efficient architectures. Depthwise convolutions [31] apply separate convolutions to each channel of a tensor reducing the number of operations and hence also the runtime, have been found to have comparable quality to standard convolutions [147]. Lightweight GANs [125] achieve fast training using a number of tricks including small batch sizes, skip-layer excitation modules which provide efficient shortcut gradient flow, as well as using a self-supervised discriminator forcing good features to be learned.

5 Autoregressive Likelihood Models

Autoregressive generative models [8] are based on the chain rule of probability, where the probability of a variable that can be decomposed as $\boldsymbol{x} = x_1, \ldots, x_n$ is expressed as

$$p(\boldsymbol{x}) = p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}). \tag{29}$$

As such, unlike GANs and energy models, it is possible to directly maximise the likelihood of the data by training a recurrent neural network to model $p(x_i|\boldsymbol{x}_{1:i-1})$ by minimising the negative log-likelihood,

$$-\ln p(\boldsymbol{x}) = -\sum_{i}^{n} \ln p(x_i|x_1, \ldots, x_{i-1}). \tag{30}$$

While autoregressive models are extremely powerful density estimators, sampling is inherently a sequential process and can be exceedingly slow on high dimensional data. Additionally, data must be decomposed into a fixed ordering; while the choice of ordering can be clear for some modalities (e.g. text and audio), it is not obvious for others such as images and can affect performance depending on the network architecture used.

5.1 Architectures

The majority of research is focused on improving network architectures to increase their receptive fields and memory, ensuring the network has access to all parts of the input to encourage consistency, as well as increasing the network capacity, allowing more complex distributions to be modelled.

5.1.1 Recurrent Neural Networks

A natural architecture to apply is that of standard recurrent neural networks (RNNs) such as LSTMs [81], [214] and GRUs [33], [138] which model sequential data by tracking information in a hidden state. However, RNNs are known to forget information, limiting their receptive field thus preventing modelling of long range relationships. This can be improved by stacking RNNs that run at different frequencies allowing long data such as multiple seconds of audio to be modelled [33]. Nevertheless, their sequential nature means that training can be too slow for many applications.

5.1.2 Causal Convolutions

An alternative approach is that of causal convolutions, which apply masked or shifted convolutions over a sequence [28], [176], [213]. When stacked, this only provides a receptive field linear with depth, however, by dilating the convolutions to skip values with some step the receptive field can be orders of magnitude higher.

5.1.3 Self-Attention

Neural attention is an approach which at each successive time step is able to select where it wishes to 'look' at previous time steps. This concept has been used to autoregressively 'draw' images onto a blank 'canvas' [60] in a manner similar to human drawing. More recently self-attention (known as Transformers when used in an encoder-decoder setup) [216] have made significant strides improving not only autoregressive models, but also other generative models due to their parallel nature, infinite receptive field, and stable training. They achieve this using an attention scheme that can reference any previous input but using an entirely independent process per time step so that there are no dependencies. Specifically, inputs are encoded as key-value pairs, where the values $\boldsymbol{V}$ represent the inputs, and the keys $\boldsymbol{K}$ act as an indexing method. At each time step a query q is made; taking the dot product of the queries and keys, a similarity vector is formed that describes which value vectors to access. This process can be expressed as

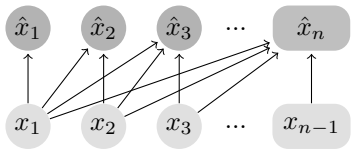


Fig. 7: Autoregressive models decompose data points using the chain rule and learn conditional probabilities.

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{31}$$

where d_k is the key/query dimension and is used to normalise gradient magnitudes. Since the self-attention process contains no recurrence, positional information must be passed into the function. A simple effective method to achieve this is to add sinusoidal positional encodings which combine sine and cosine functions of different frequencies to encode positional information [216]; alternatively others use trainable positional embeddings [30].

The infinite receptive fields of attention provides a powerful tool for representing data, however, the attention matrix $\boldsymbol{Q}\boldsymbol{K}^T$ grows quadratically with data dimension, making scaling difficult. Approaches include scaling across large quantities of GPUs [18], interleaving attention between causal convolutions [28], attending over local regions [166], and using sparse attention patterns that provide global attention when multiple layers are stacked [30]. More recently, a number of linear transformers have been proposed whose memory and time footprints grow linearly with data dimension [32], [106], [220]. By approximating the softmax operation with a kernel function with feature representation $\phi(\boldsymbol{x})$, the order of multiplications can be rearranged to

$$\Big(\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^T\Big)\boldsymbol{V} = \phi(\boldsymbol{Q})\Big(\phi(\boldsymbol{K})^T\boldsymbol{V}\Big), \tag{32}$$

allowing $\phi(\boldsymbol{K})^T\boldsymbol{V}$ to be cached and used for each query.

5.1.4 Multiscale Architectures

Even with a linear autoregressive model, scaling to high-resolution images grows quadratically with resolution. By using a multi-scale architecture that makes local independence assumptions, it is possible to reduce the complexity to $O(\ln N)$ for N pixels, allowing scaling to high resolutions [172]. By imposing a partitioning on the spatial dimensions, it is possible to build a multi-scale architecture without making independence assumptions; while this reduces the memory required, sampling times are still slow [141].

5.2 Data Modelling Decisions

An obvious method of modelling individual variables x_i is to discretize and use a softmax layer on outputs. However, this can cause complications or be infeasible in some cases such as 16-bit audio modelling in which the outputs would need 65,536 neurons. A number of solutions have been proposed including:

- Applying μ-law, a logarithmic companding algorithm which takes advantage of human perception of sound, then quantizing to 8-bit values [160].
- First predicting the first 8-bits, then predicting the second 8-bits conditioned on the first.
- Modelling output probabilities using a mixture of logistic distributions has the benefits of providing more useful gradients and allowing intensities never seen to still be sampled [176].

When modelling images, many works use "raster scan" ordering [176], [213], [214] where pixels are estimated row by row. Alternatives have been proposed such as "zig-zag" ordering [28] which allows pixels to depend on previously sampled pixels to the left and above, providing more relevant context. Another factor when modelling images is how to factorise sub-pixels. While it is possible to treat them as independent variables, this adds additional complexity. Alternatively, it is possible to instead condition on whole pixels, and output joint distributions in a single step [175].

5.3 Variants

As aforementioned, directly maximising the likelihood of data points may not be appropriate, with likelihood not always correlating with greater perceptual quality [199]. It requires the choice between spreading over models or collapsing to a single mode, meaning approximate solutions are not incentivised; as such, it can be viewed as lacking an underlying metric, focusing entirely on probability.

5.3.1 Quantile Regression

One alternative approach is quantile regression, which is equivalent to minimising the Wasserstein distance between distributions. In this case, the inverse cumulative distribution function can be learned by minimising the Huber quantile loss [91],

$$\rho_\tau^\kappa(u) = \begin{cases} \frac{|\tau - \mathbb{I}\{u \leq 0\}|}{2\kappa} u^2, & \text{if} |u| \leq \kappa \\ |\tau - \mathbb{I}\{u \leq 0\}|(|u| - \frac{1}{2}\kappa), & \text{otherwise.} \end{cases} \tag{33}$$

Quantile regression has been applied to autoregressive models [161], requiring only minor architectural changes, and also having the benefit of not requiring quantization and improving perceptual quality scores.

5.3.2 Unnormalized Densities

Explicitly modelling the density of data points using a complementary distribution restricts the expressiveness of the network, for instance, finite mixtures of Gaussians struggle to model high frequency signals. A simple approach is to reduce the Lipschitz constant of the data distribution, for instance by adding Gaussian noise [139]. A more general approach is to learn an autoregressive energy model, removing this restriction at the expense of less efficient sampling. Nevertheless, MCMC sampling a low-dimensional space converges much faster than the high dimensional sampling involved for other energy models [42]. One approach to train an autoregressive energy model is to estimate the normalising constant for each time step using importance sampling, which due to the relatively low dimensionality is manageable [145]. Another proposed approach is to output the score at each time step by optimising what the authors term the composite score matching divergence, removing the need to calculate normalising constants [140].

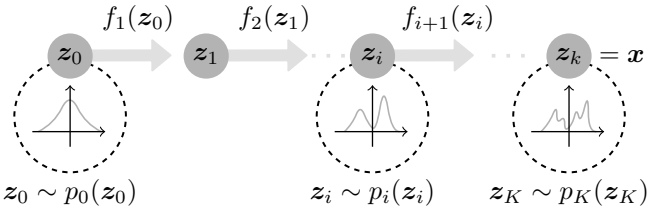


Fig. 8: Normalizing flows build complex distributions by mapping a simple distribution through invertible functions.

6 Normalizing Flows

While training autoregressive models with max-likelihood offers plenty of benefits including stable training, density estimation, and a useful validation metric, the slow sampling speed and poor scaling properties handicaps them significantly. Normalizing flows are a technique that also allows exact likelihood calculation while being efficiently parallelisable as well as offering a useful latent space for downstream tasks.

Consider an invertible, smooth function $f\colon \mathbb{R}^d \to \mathbb{R}^d$; by applying this transformation to a random variable $\boldsymbol{z} \sim q(\boldsymbol{z})$, then the distribution of the resulting random variable $\boldsymbol{z}' = f(\boldsymbol{z})$ can be determined through the change of variables rule (and application of the chain rule),

$$q(\boldsymbol{z}') = q(\boldsymbol{z}) \left| \det \frac{\partial f^{-1}}{\partial \boldsymbol{z}'} \right| = q(\boldsymbol{z}) \left| \det \frac{\partial f}{\partial \boldsymbol{z}} \right|^{-1}. \tag{34}$$

Consequently, arbitrarily complex densities can be constructed by composing simple maps and applying Equation 34 [217]. This chain is known as a normalizing flow [173] (see Fig. 8). The density $q_K(\boldsymbol{z}_K)$ obtained by successively transforming a random variable $\boldsymbol{z}_0$ with distribution q_0 through a chain of K transformations f_k can be defined as

$$\boldsymbol{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\boldsymbol{z}_0), \tag{35a}$$

$$\ln q_K(\boldsymbol{z}_K) = \ln q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}} \right|. \tag{35b}$$

Each transformation therefore must be sufficiently expressive while being easily invertible and have an efficient to compute Jacobian determinant. While restrictive, there have been a number of works which have introduced more powerful invertible functions (see Table 2). Nevertheless, normalizing flow models are typically less parameter efficient than other generative models.

One disadvantage of requiring transformations to be invertible is that the input dimension must be equal to the output dimension which makes deep models inefficient and difficult to train. A popular solution to this is to use a multi-scale architecture [39], [111] (see Fig. 9) which divides the process into a number of stages, at the end of each factoring out half of the remaining units which are treated immediately as outputs. This allows latent variables to sequentially represent course to fine features and permits deeper architectures.

TABLE 2: Normalizing Flow Layers: $\odot$ represents elementwise multiplication, $\star_l$ represents a cross-correlation layer

Description	Function	Inverse Function	Log-Determinant
Low Rank			
Planar [173]	$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{z} + b)$ With $\boldsymbol{w} \in \mathbb{R}^D$, $\boldsymbol{u} \in \mathbb{R}^D$, $\boldsymbol{b} \in \mathbb{R}$	No closed form inverse	$\ln \lvert 1 + \boldsymbol{u}^T h'(\boldsymbol{w}^T\boldsymbol{z} + b)\boldsymbol{w} \rvert$
Sylvester [12], [72]	$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{U}h(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})$	No closed form inverse	$\ln \det(\boldsymbol{I}_M + \text{diag}(h'(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}))\boldsymbol{W}\boldsymbol{U}^T)$
Coupling/Autoregressive			
General Coupling	$\boldsymbol{y}^{(1:d)} = \boldsymbol{x}^{(1:d)}$ $\boldsymbol{y}^{(d+1:D)} = h(\boldsymbol{x}^{(d+1:D)}; f_\theta(\boldsymbol{x}^{(1:d)}))$	$\boldsymbol{x}^{(1:d)} = \boldsymbol{y}^{(1:d)}$ $\boldsymbol{x}^{(d+1:D)} = h^{-1}(\boldsymbol{y}^{(d+1:D)}; f_\theta(\boldsymbol{y}^{(1:d)}))$	$\ln \lvert \det \nabla_{\boldsymbol{x}^{(d+1:D)}} h \rvert$
MAF [165]	$y^{(t)} = h(x^{(t)}; f_\theta(\boldsymbol{x}^{(1:t-1)}))$	$x^{(t)} = h^{-1}(y^{(t)}; f_\theta(\boldsymbol{x}^{(1:t-1)}))$	$-\sum_{t=1}^D \ln \lvert \frac{\partial y^{(t)}}{\partial x^{(t)}} \rvert$
IAF [112]	$y^t = h(x^{(t)}; f_\theta(\boldsymbol{y}^{(1:t-1)}))$	$x^t = h^{-1}(y^{(t)}; f_\theta(\boldsymbol{y}^{(1:t-1)}))$	$\sum_{t=1}^D \ln \lvert \frac{\partial y^{(t)}}{\partial x^{(t)}} \rvert$
Affine Coupling [39]	$h(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{x} \odot \exp(\boldsymbol{\theta}_1) + \boldsymbol{\theta}_2$	$h^{-1}(\boldsymbol{y}; \boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{\theta}_2) \odot \exp(-\boldsymbol{\theta}_1)$	$\sum_{i=1}^d \theta_1^{(i)}$
Flow++ [79]	$h(\boldsymbol{x}; \boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_1) \odot F(\boldsymbol{x}, \boldsymbol{\theta}_3) + \boldsymbol{\theta}_2$ where F is a monotone function.	Calculated through bisection search	$\sum_{i=1}^d \theta_1^{(i)} + \ln \frac{\partial F(\boldsymbol{x}, \boldsymbol{\theta}_3)_i}{\partial x_i}$
Spline Flows [144] [46] [47]	$h(\boldsymbol{x}; \boldsymbol{\theta}) = \text{Spline}(\boldsymbol{x}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are the spline's knots.	$h^{-1}(\boldsymbol{y}; \boldsymbol{\theta}) = \text{Spline}^{-1}(\boldsymbol{y}; \boldsymbol{\theta})$	Computed in closed-form as a product of quotient derivatives
B-NAF [36]	$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}^T$ for blocked weights: $\boldsymbol{W} = \exp(\tilde{\boldsymbol{W}}) \odot \boldsymbol{M}_d + \tilde{\boldsymbol{W}} \odot \boldsymbol{M}_o$ where $\boldsymbol{M}_d$ selects diagonal blocks and $\boldsymbol{M}_o$ selects off-diagonal blocks.	No closed form inverse	$\ln \sum_{i=1}^d \exp(\tilde{W}_{ii})$
Convolutions			
1x1 Convolution [111]	$h \times w \times c$ tensor $\boldsymbol{x}$ & $c \times c$ tensor $\boldsymbol{W}$ $\forall i, j : \boldsymbol{y}_{i,j} = \boldsymbol{W}\boldsymbol{x}_{i,j}$	$\forall i, j : \boldsymbol{x}_{i,j} = \boldsymbol{W}^{-1}\boldsymbol{y}_{i,j}$	$h \cdot w \cdot \ln \lvert \det \boldsymbol{W} \rvert$
Emerging Convolutions [84]	$\boldsymbol{k} = \boldsymbol{w}_1 \odot \boldsymbol{m}_1, \quad \boldsymbol{g} = \boldsymbol{w}_2 \odot \boldsymbol{m}_2$ $\boldsymbol{y} = \boldsymbol{k} \star_l (\boldsymbol{g} \star_l \boldsymbol{x})$	$\boldsymbol{z}_t = (\boldsymbol{y}_t - \sum_{i=t+1} G_{t,i}\boldsymbol{z}_i)/G_{t,t}$ $\boldsymbol{x}_t = (\boldsymbol{z}_t - \sum_{i=1}^{t-1} K_{t,i}\boldsymbol{x}_i)/K_{t,t}$	$\sum_c \ln \lvert \boldsymbol{k}_{c,c,m_y,m_x} \boldsymbol{g}_{c,c,m_y,m_x} \rvert$
Lipshitz Residual			
i-ResNet [7]	$\boldsymbol{y} = \boldsymbol{x} + f(\boldsymbol{x})$ where $\lVert f \rVert_L < 1$	$\boldsymbol{x}_1 = \boldsymbol{y}. \quad \boldsymbol{x}_{n+1} = \boldsymbol{y} - f(\boldsymbol{x}_n)$ converging at an exponential rate	$\text{tr}(\ln(\boldsymbol{I} + \nabla_{\boldsymbol{x}} f)) =$ $\sum_{k=1}^\infty (-1)^{k+1} \frac{\text{tr}((\nabla_{\boldsymbol{x}} f)^k)}{k}$

6.1 Coupling and Autoregressive Layers

A simple way of building an expressive invertible function is the coupling flow [38], which divide inputs into two sections and applies a bijection h on one half parameterised by the other,

$$\boldsymbol{y}^{(1:d)} = \boldsymbol{x}^{(1:d)}, \tag{36a}$$

$$\boldsymbol{y}^{(d+1:D)} = h(\boldsymbol{x}^{(d+1:D)}; f_\theta(\boldsymbol{x}^{(1:d)})), \tag{36b}$$

here f can be arbitrarily complex i.e. a neural network. h tends to be selected as an elementwise function making the Jacobian triangular allowing efficient computation of the determinant, i.e. the product of elements on the diagonal.

6.1.1 Affine Coupling

A simple example of this is the affine coupling layer [39],

$$\boldsymbol{y}^{(d+1:D)} = \boldsymbol{x}^{(d+1:D)} \odot \exp(f_\sigma(\boldsymbol{x}^{(1:d)})) + f_\mu(\boldsymbol{x}^{(1:d)}), \tag{37}$$

which has a simple Jacobian determinant and can be trivially rearranged to obtain a definition of $\boldsymbol{x}^{(d+1:D)}$ in terms of $\boldsymbol{y}$, provided that the scaling coefficients are not 0. This simplicity, however, comes at the cost of expressivity; stacking numerous such flows significantly increases their expressivity, allowing them to learn representations of complex high dimensional data such as images [111], but it is unknown whether multiple layers of affine flows are universal approximators or not [164].

6.1.2 Monotone Functions

Another method of creating invertible functions that can be applied element-wise is to enforce monotonicity. One possibility to achieve this is to define h as an integral over a positive but otherwise unconstrained function g [222],

$$h(x_i; \boldsymbol{\theta}) = \int_0^{x_i} g_\phi(x; \boldsymbol{\theta}_1) dx + \theta_2, \tag{38}$$

however, this integration requires numerical approximation. Alternatively, by choosing g to be a function with a known integral solution, h can be efficiently evaluated. This has been accomplished using positive polynomials [95] and the CDF of a mixture of logits [79]. Both cases, however, don't have analytical inverses and have to be approximated iteratively with bisection search. Another option is to represent g as a monotonic spline: a piecewise function where each piece is easy to invert. As such, the inverse is as fast to evaluate as the forward pass. Linear and quadratic splines [144], cubic splines [46], and rational-quadratic splines [47] have been applied so far.

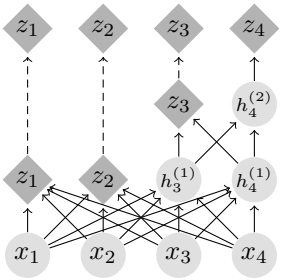


Fig. 9: Factoring out variables at different scales allows normalizing flows to scale to high dimensional data.

6.1.3 Autoregressive Flows

For a single coupling layer, a significant proportional of inputs remain unchanged. A more flexible generalisation of coupling layers is the autoregressive flow, or MAF [165],

$$y^{(t)} = h(x^{(t)}; f_\theta(\boldsymbol{x}^{(1:t-1)})). \tag{39}$$

Here f_θ can be arbitrarily complex, allowing the use of advances in autoregressive modelling (Section. 5), and h is a bijection as used for coupling layers. Some monotonic bijectors have been created specifically for autoregressive flows, namely Neural Autoregressive Flows (NAF) [87] and Block NAF [36]. Unlike coupling layers, a single autoregressive flow is a universal approximator.

Alternatively, an autoregressive flow can be conditioned on $\boldsymbol{y}^{(1:t-1)}$ rather than $\boldsymbol{x}^{(1:t-1)}$, this is known as an Inverse Autoregressive Flow, or IAF [112]. While coupling layers can be evaluated efficiently in both directions, MAF permits parallel density estimation but sequential sampling, and IAF permits parallel sampling but sequential density estimation.

6.1.4 Probability Density Distillation

Inverse autoregressive flows [112] offer the ability to sample from an autoregressive model in parallel, however, training via max-likelihood is inherently sequential making this infeasible for high dimensional data. Probability density distillation [159] has been proposed as a solution to this where a second pre-trained autoregressive network which can perform density estimation in parallel is used as a 'teacher' network while an IAF network is used as a 'student' and mimics the teacher's distribution by minimising the KL divergence between the two distributions:

$$D_{KL}(p_S||p_T) = H(p_S, p_T) - H(p_S), \tag{40}$$

where p_S and p_T are the student's and teacher's distributions respectively, $H(p_S, p_T)$ is the cross-entropy between p_S and p_T, and $H(p_S)$ is the entropy of p_S. Crucially, this never requires the student's inverse function to be used allowing it can computed entirely in parallel.

6.2 Convolutional

A considerable problem with coupling and autoregressive flows is the restricted triangular Jacobian, meaning that all inputs cannot interact with each other. Simple solutions involve fixed permutations on the output space such as reversing the order [38], [39]. A more general approach is to use a 1×1 convolution which is equivalent to a linear transformation applied across channels [111].

A number of works have proposed other convolution-based flows that permit larger kernel sizes. A number of these apply variations on causal convolutions [160], including emerging convolutions [84] whose inverse is sequential, MaCow [132] which uses smaller conditional fields allowing more efficient sampling, and MintNet [190] which approximates the inverse using fixed-point iteration. Alternative approaches to causal masking involve imposing repeated (periodic) structure [100], however in general this is not a good assumption for image modelling, as well as representing convolutions as exponential matrix-vector products, $\exp(\boldsymbol{M})\boldsymbol{x}$, approximated implicitly with a power series, allowing otherwise unconstrained kernels [85].

6.3 Residual Flows

Residual networks [74] are a popular technique to build deep neural networks, avoiding vanishing gradients by stacking blocks of the form

$$\boldsymbol{y} = \boldsymbol{x} + f_\theta(\boldsymbol{x}). \tag{41}$$

By restricting f_θ it is possible to enforce invertibility.

6.3.1 Matrix Determinant Lemma

If a function has a certain residual form, then its Jacobian determinant can be computed with the matrix determinant lemma [173]. A simple example is planar flow [173] which is equivalent to a 3 layer MLP with a single neuron bottleneck:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{x} + b), \tag{42}$$

where $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$, and h is a differentiable non-linearity function. Planar flows are invertible provided some simple conditions are satisfied, however its inverse is difficult to compute making it only practical for density estimation tasks. A higher rank generalisation of the matrix determinant lemma has been applied to planar flows, known as Sylvester flows, removing the severe bottleneck thus allowing greater representation ability [12], [72].

6.3.2 Lipschitz Constrained

By restricting the Lipschitz constant of $f_\theta, \|f_\theta\|_L < 1$, then this block is invertible [7]. The inverse, however, has no closed form definition but can be found through fixed-point iteration which by the Banach fixed-point theorem converges to a fixed unique solution at an exponential rate dependant on $\|f_\theta\|_L$. The authors originally proposed a biased approximation of the log determinant of the Jacobian as a power series where the Jacobian trace is approximated using Hutchkinson's trace estimator (see Table 2), but an unbiased approximator known as a Russian roulette estimator has also been proposed [24]. Unlike coupling layers, residual flows have dense Jacobians, allowing interaction. Enforcing Lipschitz constraints has been achieved with convolutional networks [55], [126], [143] as well as self-attention [107].

Making strong Lipschitz assumptions severely restricts the class of functions learned; an N layer residual flow network is at most 2^N-Lipshitz. Implicit flows [129] bypass this by solving implicit equations of the form

$$F(\boldsymbol{x}, \boldsymbol{y}) = f_\theta(\boldsymbol{x}) - f_\phi(\boldsymbol{y}) + \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0}, \tag{43}$$

where both f_θ and f_ϕ both have Lipschitz constants less than 1. Both the forwards (solve for $\boldsymbol{y}$ given $\boldsymbol{x}$) and backwards (solve for $\boldsymbol{x}$ given $\boldsymbol{y}$) directions require solving a root finding problem similar to the inverse process of residual flows; indeed, an implicit flow is equivalent to the composition of a residual flow and the inverse of a residual flow. This allows them to model arbitrary Lipschitz transformations.

6.4 Surjective and Stochastic Layers

Restricting the class of functions available to those that are invertible introduces a number of practical problems related to the topology-preserving property of diffeomorphisms. For example, mapping a uni-modal distribution to a multi-modal distribution is extremely challenging, requiring a highly varying Jacobian [40]. By composing bijections with surjective or stochastic layers these topological constraints can be bypassed [150]. While the log-likelihood of stochastic layers can only be bounded by their ELBO, functions surjective in the inference direction permit exact likelihood evaluation even with altered dimensionality. Surjective transformations have the following likelihood contributions:

$$\mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{x})}\left[\ln \frac{p(\boldsymbol{x}|\boldsymbol{y})}{q(\boldsymbol{y}|\boldsymbol{x})}\right], \tag{44}$$

where $p(\boldsymbol{x}|\boldsymbol{y})$ is deterministic for generative surjections, and $q(\boldsymbol{y}|\boldsymbol{x})$ is deterministic for inference surjections.

One approach to build a surjective layer is to augment the input space with additional dimensions allowing smoother transformation to be learned [23], [44], [86]; the inverse process, where some dimensions are factored out, is equivalent to a multi-scale architecture [39]. Another approach known as RAD [40] learns a partitioning of the data space into disjoint subsets $\{\mathcal{Y}_i\}_{i=1}^K$, and applies piecewise bijections to each region $g_i \colon \mathcal{X} \to \mathcal{Y}_i, \forall i \in \{1, \dots, K\}$. The generative direction learns a classifier on $\mathcal{X}$, $i \sim p(i|\boldsymbol{x})$, allowing the inverse to be calculated as $\boldsymbol{y} = g_i(\boldsymbol{x})$. Similar to both of these approaches are CIFs [34] which consider a continuous partitioning of the data space via augmentation equivalent to an infinite mixture of normalizing flows. Other approaches include modelling finite mixtures of flows [43].

Some powerful stochastic layers have already been discussed in this survey, namely VAEs [110] and DDPMs [80]. Stochastic layers have been incorporated into normalizing flows by interleaving small energy models, sampled with MCMC, between bijectors [225].

6.5 Continuous Time Flows

It is possible to consider a normalizing flow with an infinite number of steps that is defined instead by an ordinary differential equation specified by a Lipschitz continuous neural network f with parameters θ, that describes the transformation of a hidden state $\boldsymbol{z}(t) \in \mathbb{R}^D$ [25],

$$\frac{\partial \boldsymbol{z}(t)}{\partial t} = f(\boldsymbol{z}(t), t, \theta). \tag{45}$$

Starting from input noise $\boldsymbol{z}(0)$, an ODE solver can solve an initial value problem for some time T, at which data is defined, $\boldsymbol{z}(T)$. Modelling a transformation in this form has a number of advantages such as inherent invertibility by running the ODE solver backwards, parameter efficiency, and adaptive computation. However, it is not immediately clear how to train such a model through backpropagation. While it is possible to backpropagate directly through an ODE solver, this limits the choice of solvers to differentiable ones as well as requiring large amounts of memory. Instead, the authors apply the adjoint sensitivity method which instead solves a second, augmented ODE backwards in time and allows the use of a black box ODE solver. That is, to optimise a loss dependent on an ODE solver:

$$\begin{aligned}\mathcal{L}(\boldsymbol{z}(t_1)) &= \mathcal{L}\left(\boldsymbol{z}(t_0) + \int_{t_0}^{t_1} f(\boldsymbol{z}(t), t, \theta) dt\right), \\ &= \mathcal{L}(\text{ODESolve}(\boldsymbol{z}(t_0), f, t_0, t_1, \theta)),\end{aligned} \tag{46}$$

the adjoint $\boldsymbol{a}(t) = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}(t)}$ can be used to calculate the derivative of loss with respect to the parameters in the form of another initial value problem [167],

$$\frac{\partial \mathcal{L}}{\partial \theta} = \int_{t_1}^{t_0} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}(t)}\right)^T \frac{\partial f(\boldsymbol{z}(t), t, \theta)}{\partial \theta} dt, \tag{47}$$

which can be efficiently evaluated by automatic differentiation at a time cost similar to evaluating f itself.

Despite the complexity of this transformation, the continuous change of variables rule is remarkably simple:

$$\frac{\partial \ln p(\boldsymbol{z}(t))}{\partial t} = -tr\left(\frac{\partial}{\partial \boldsymbol{z}(t)} f(\boldsymbol{z}(t), t, \theta)\right), \tag{48}$$

and can be computed using an ODE solver as well. The resulting continuous-time flow is known as FFJORD [56]. Since the length of the flow tends to infinity (an infinitesimal flow), the true posterior distribution can be recovered [173].

As previously mentioned, invertible functions suffer from topological problems; this is especially true for Neural ODEs since their continuous nature prevents trajectories from crossing. Similar to augmented normalizing flows [86], this can be solved by providing additional dimensions for the flow to traverse [44]. Specifically, a p-dimensional Euclidean space can be approximated by a Neural ODE in a $(2p+1)$-dimensional space [233].

6.5.1 Regularising Trajectories

ODE solvers can require large numbers of network evaluations, notably when the ODE is stiff or the dynamics change quickly in time. By introducing regularisation, a simpler ODE can be learned, reducing the number of evaluations required. Specifically, all works here are inspired by optimal transport theory to encourage straight trajectories. Monge-Ampère Flow [236] and Potential Flow Generators [229] parameterise a potential function satisfying the Monge-Ampère equation [20], [217] with a neural network. RNODE [50] applies transport costs to FFJORD as well as regularising the Frobenius norm of the Jacobian, encouraging straight trajectories. OT-Flow [158] combines these approaches, parameterising a potential as well as applying transport costs, additionally, utilising the optimal transport derivation they derive an exact trace definition with similar complexity to using Hutchkinson's trace estimator.

7 Connections between Generative Models

While deep generative models all maximise the likelihood of the training data, they achieve this in various ways, making different trade offs such as diversity for quality, sampling for training speed, and capacity for complexity. There are also a plethora of hybrid approaches, attempting to get the best of both worlds, or at least find saddle points, balancing trade-offs. The varied connections between these systems mean that advances in one field inevitably benefit others, for instance, improved variational bounds [19], [131] are beneficial for not only VAEs [110] but also diffusion models [79] and surjective flows [150].

An area of particular overlap is the augmentation of training data, acting as regularisation while increasing the quantity of available training data; augmentation can be applied almost freely to GANs (see Section 4.1.3), however, direct application to other architectures would lead to augmentations leaking into samples. By conditioning these models on the augmentation type, effective training is not only possible but has been found to substantially improve sample quality, even on small models [99].

7.1 Assessing Quality

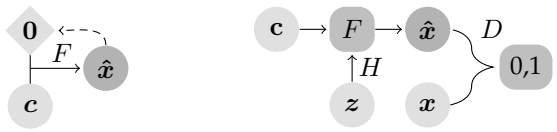
A huge problem when developing generative models is how to effectively evaluate and compare them. Qualitative comparison of random samples plays a large role in the majority of state-of-the-art works, however, it is subjective and time-consuming to compare many works. Calculating the log-likelihood on a separate validation set is popular for tractable likelihood models but comparison with implicit likelihood models is difficult and while it is a good measure of diversity, it does not correlate well with quality [199].

One approach to quantify sample quality is Inception Score (IS) [175] which takes a trained classifier and determines whether a sample has low label entropy, indicating that a meaningful class is likely, and whether the distribution of classes over a large number of samples has high entropy, indicating that a diverse range of images can be sampled. A perfect IS can be scored by a model that creates only one image per class [130] leading to the creation of Fréchet Inception Distance (FID) [75] which models the activations of a particular layer of a classifier as multivariate Gaussians for real and generated data, measuring the Fréchet distance between the two.

These approaches are trivially solved by memorising the dataset and are less applicable to non-natural image-related data. Kernel Inception Distance (KID) [13] mitigates this somewhat, instead calculating the squared maximum mean discrepancy in feature space, however, pretrained features may not be sufficient to detect overfitting. Another approach is to train a neural network to distinguish between real and generated samples similar to the discriminator from a GAN; while this detects overfitting well, it increases the complexity and time required to evaluate a model and is biased towards adversarially trained models [67].

7.2 Applications

In general, the definition of a generative model means that any technique can be used on any modality/task, however, some models are more suited for certain tasks. Standard autoregressive networks are popular for text/audio generation [18], [30], [160]; VAEs have been applied but posterior collapse is difficult to mitigate [5], [16]; GANs are more parameter efficient but struggle to model discrete data [149] and suffer from mode collapse [115]; some normalizing flows offer parallel synthesis, providing substantial speedup [168], [204], [244]. Video synthesis is more challenging due its exceptionally high dimensionality, typically approaches combine a latent-based implicit generative model to generate individual frames, with an autoregressive network used to predict future latents [5], [116], [120]. This approach is similar to the one used in reinforcement learning to construct world models [69], [70].



(a) Implicit GON [15]. (b) Implicit GAN [45].

Fig. 10: Implicit networks model data continuously permitting arbitrarily high resolutions. Dashed lines represent gradients, F is an implicit network, and H is a hypernetwork.

7.3 Implicit Representation

Typically deep architectures discussed in this survey are built with data represented as discrete arrays thus using discrete components such as convolutions and self-attention. Implicit representation on the other hand treats data as continuous signals, mapping coordinates to data values [183], [197]. Implicit Gradient Origin Networks (GONs; Fig. 10a) [15] form a latent variable model by concatenating latent vectors with coordinates which are passed through an implicit network; here latent vectors are calculated as the gradient of a reconstruction loss with respect to the origin. By sampling using a finer grid of coordinates, super-resolution beyond resolutions seen during training is possible. Another approach to learn an implicit generative model as a GAN is to map latents to the weights of an implicit function using a hyper-network, the output of which is classified by a discriminator [45] (Fig. 10b).

8 Conclusion

While GANs have led the way in terms of sample quality for some time now, the gap between other approaches is shrinking; the diminished mode collapse and simpler training objectives make these models more enticing than ever, however, the number of parameters required in addition to slow run-times pose a substantial handicap. Despite this, recent work notably in hybrid models offers a balance between extremes at the expense of extra model complexity that hinders broader adoption. A clear stand out is the application of innovative data augmentation strategies within generative models, offering benefits impressively without necessitating more powerful architectures. When it comes to scaling models to high-dimensional data, attention is a common theme, allowing long-range dependencies to be

learned; recent advances in linear attention will aid scaling to even higher resolutions. Implicit networks are another promising direction, allowing efficient synthesis of arbitrarily high resolution and irregular data. Similar unified generative models capable of modelling continuous, irregular, and arbitrary length data, over different scales and domains will be key for the future of generalisation.

REFERENCES

[1] Guillaume Alain, Yoshua Bengio, Li Yao, Jason Yosinski, éric Thibodeau-Laufer, Saizheng Zhang, and Pascal Vincent. GSNs: generative stochastic networks. *Information and Inference*, 5(2):210–249, 2016.

[2] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized Energy Based Models. In *ICLR*, 2021.

[3] Martin Arjovsky and Leon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. In *ICLR*, 2017.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv:1701.07875*, 2017.

[5] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic Variational Video Prediction. In *ICLR*, 2018.

[6] Matthias Bauer and Andriy Mnih. Resampled Priors for Variational Autoencoders. In *AISTATS*, pages 66–75, 2019.

[7] Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible Residual Networks. In *ICML*, 2019.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *JMLR*, 3:1137–1155, 2003.

[9] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432*, 2013.

[10] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized Denoising Auto-Encoders as Generative Models. *NeurIPS 26*, 2013.

[11] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized Denoising Auto-Encoders as Generative Models. *NeurIPS*, 26, 2013.

[12] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester Normalizing Flows for Variational Inference. In *UAI*, 2018.

[13] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018.

[14] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the Latent Space of Generative Networks. *arXiv:1707.05776*, 2019.

[15] Sam Bond-Taylor and Chris G. Willcocks. Gradient Origin Networks. In *ICLR*, 2021.

[16] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *arXiv:1511.06349*, 2016.

[17] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv:1809.11096*, 2019.

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165*, 2020.

[19] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. In *ICLR*, 2016.

[20] Luis A. Caffarelli and Mario Milman. Monge Ampère Equation: Applications to Geometry, Optimization. American Mathematical Soc., 1999.

[21] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On Contrastive Divergence Learning. In *AISTATS*, volume 10, pages 33–40, 2005.

[22] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is Secretly an Energy-based Model and You Should Use Discriminator Driven Latent Sampling. *NeurIPS*, 33, 2020.

[23] Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. VFlow: More Expressive Generative Flows with Variational Data Augmentation. In *ICML*, 2020.

[24] Ricky T. Q. Chen, Jens Behrmann, David K. Duvenaud, and Joern-Henrik Jacobsen. Residual Flows for Invertible Generative Modeling. *NeurIPS*, 2019.

[25] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. *NeurIPS 31*, 2018.

[26] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-Supervised GANs via Auxiliary Rotation Loss. In *IEEE CVPR*, 2019.

[27] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. In *ICLR*, 2017.

[28] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. PixelSNAIL: An Improved Autoregressive Generative Model. *ICML*, 2, 2017.

[29] Rewon Child. Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. In *ICLR*, 2021.

[30] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv:1904.10509*, 2019.

[31] Francois Chollet. Xception: Deep Learning With Depthwise Separable Convolutions. In *IEEE CVPR*, 2017.

[32] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking Attention with Performers. In *ICLR*, 2021.

[33] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555*, 2014.

[34] Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows. In *ICML*, 2020.

[35] Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. *arXiv:1903.05789*, 2019.

[36] Nicola De Cao, Ivan Titov, and Wilker Aziz. Block Neural Autoregressive Flow. In *UAI*, 2019.

[37] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *NeurIPS*, 28, 2015.

[38] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-Linear Independent Components Estimation. In *ICLR Workshop*, 2015.

[39] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *ICLR*, 2017.

[40] Laurent Dinh, Jascha Sohl-Dickstein, Razvan Pascanu, and Hugo Larochelle. A RAD approach to deep mixture models. In *ICLR Workshop*, 2019.

[41] Alexey Dosovitskiy and Thomas Brox. Generating Images with Perceptual Similarity Metrics based on Deep Networks. *NeurIPS*, 2016.

[42] Yilun Du and Igor Mordatch. Implicit Generation and Generalization in Energy-Based Models. *NeurIPS*, 33, 2019.

[43] Leo L. Duan. Transport Monte Carlo. *arXiv:1907.10448*, 2020.

[44] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. *NeurIPS 32*, 2019.

[45] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative Models as Distributions of Functions. *arXiv:2102.04776*, 2021.

[46] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-Spline Flows. In *ICML Workshop*, 2019.

[47] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural Spline Flows. *NeurIPS 32*, 2019.

[48] Conor Durkan, Iain Murray, and George Papamakarios. On Contrastive Learning for Likelihood-free Inference. In *ICML*, 2020.

[49] Patrick Esser, Robin Rombach, and Björn Ommer. Taming Transformers for High-Resolution Image Synthesis. *arXiv:2012.09841*, 2021.

[50] Chris Finlay, Joern-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to Train Your Neural ODE: the World of Jacobian and Kinetic Regularization. In *ICML*, 2020.

[51] Ruiqi Gao, Erik Nijkamp, Diederik P. Kingma, Zhen Xu, Andrew M. Dai, and Ying Nian Wu. Flow Contrastive Estimation of Energy-Based Models. In *IEEE CVPR*, 2020.

[52] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P. Kingma. Learning Energy-Based Models by Diffusion Recovery Likelihood. In *ICLR*, 2021.

[53] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf. From Variational to Deterministic Autoencoders. In *ICLR*, 2020.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *NeurIPS 27*, 2014.

[55] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[56] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. In *ICLR*, 2019.

[57] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the Stein Discrepancy for Training and Evaluating Energy-Based Models without Sampling. In *ICML*, 2020.

[58] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. In *ICLR*, 2020.

[59] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No MCMC for me: Amortized sampling for fast and stable training of energy-based models. In *ICLR*, 2021.

[60] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A Recurrent Neural Network For Image Generation. In *ICML*, 2015.

[61] Paulina Grnarova, Yannic Kilcher, Kfir Y Levy, Aurelien Lucchi, and Thomas Hofmann. Generative Minimization Networks: Training GANs Without Competition. *arXiv:2103.12685*, 2021.

[62] Paulina Grnarova, Kfir Y Levy, Aurelien Lucchi, Nathanael Perraudin, Ian Goodfellow, Thomas Hofmann, and Andreas Krause. A Domain Agnostic Measure for Monitoring and Evaluating GANs. *NeurIPS*, 32, 2019.

[63] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. In *AAAI*, 2018.

[64] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *arXiv:2001.06937*, 2020.

[65] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. *NeurIPS 30*, 2017.

[66] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: A Latent Variable Model for Natural Images. In *ICLR*, 2017.

[67] Ishaan Gulrajani, Colin Raffel, and Luke Metz. Towards GAN Benchmarks Which Require Generalization. In *ICLR*, 2019.

[68] Michael Gutmann and Aapo Hyvärinen. Noise-Contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models. In *AISTATS*, pages 297–304, 2010.

[69] David Ha and Jürgen Schmidhuber. World Models. *arXiv:1803.10122*, 2018.

[70] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *ICLR*, 2020.

[71] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating Back-Propagation for Generator Network. *AAAI*, 31(1), 2017.

[72] Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Variational Inference with Orthogonal Normalizing Flows. In *Workshop on Bayesian Deep Learning, NIPS*, 2017.

[73] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. In *ICLR*, 2019.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE CVPR*, 2016.

[75] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *NeurIPS 30*, 2017.

[76] G E Hinton and T J Sejnowski. Optimal Perceptual Inference. In *IEEE CVPR*, 1983.

[77] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[78] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006.

[79] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. In *ICML*, 2019.

[80] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *arXiv:2006.11239*, 2020.

[81] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[82] Matthew Hoffman, Pavel Sountsov, Joshua V. Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. NeuTra-lizing Bad Geometry in Hamiltonian Monte Carlo Using Neural Transport. *arXiv:1903.03704*, 2019.

[83] Matthew D Hoffman and Matthew J Johnson. ELBO surgery: yet another way to carve up the variational evidence lower bound. In *NeurIPS Workshop*, 2016.

[84] Emiel Hoogeboom, Rianne van den Berg, and Max Welling. Emerging Convolutions for Generative Normalizing Flows. In *ICML*, 2019.

[85] Emiel Hoogeboom, Victor Garcia Satorras, Jakub Tomczak, and Max Welling. The Convolution Exponential and Generalized Sylvester Flows. *NeurIPS*, 33, 2020.

[86] Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented Normalizing Flows: Bridging the Gap Between Generative Flows and Latent Variable Models. *arXiv:2002.07101*, 2020.

[87] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural Autoregressive Flows. In *ICML*, 2018.

[88] Chin-Wei Huang, Ahmed Touati, Laurent Dinh, Michal Drozdzal, Mohammad Havaei, Laurent Charlin, and Aaron Courville. Learnable Explicit Density for Continuous Latent Space and Variational Inference. *arXiv:1710.02248*, 2017.

[89] Huaibo Huang, zhihang li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae: Introspective Variational Autoencoders for Photographic Image Synthesis. *NeurIPS*, 31, 2018.

[90] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked Generative Adversarial Networks. In *IEEE CVPR*, 2017.

[91] Peter J. Huber. Robust Estimation of a Location Parameter. In *Breakthroughs in Statistics: Methodology and Distribution*, Springer Series in Statistics, pages 492–518. New York, NY, 1992.

[92] Ferenc Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv:1511.05101*, 2015.

[93] Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 6:695–709, 2005.

[94] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, 2015.

[95] Priyank Jaini, Kira A. Selby, and Yaoliang Yu. Sum-of-Squares Polynomial Flow. In *ICML*, 2019.

[96] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. TransGAN: Two Transformers Can Make One Strong GAN. *arXiv:2102.07074*, 2021.

[97] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. *ICLR*, 2018.

[98] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[99] Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution Augmentation for Generative Modeling. In *ICML*, 2020.

[100] Mahdi Karami, Dale Schuurmans, Jascha Sohl-Dickstein, Laurent Dinh, and Daniel Duckworth. Invertible Convolutional Flow. *NeurIPS*, 2019.

[101] Animesh Karnewar and Oliver Wang. MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks. In *IEEE CVPR*, 2020.

[102] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *ICLR*, 2018.

[103] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data. *NeurIPS*, 33, 2020.

[104] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *IEEE CVPR*, 2019.

[105] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. In *IEEE CVPR*, 2020.

[106] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *ICML*, 2020.

[107] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The Lipschitz Constant of Self-Attention. *arXiv:2006.04710*, 2020.

[108] Taesup Kim and Yoshua Bengio. Deep Directed Generative Models with Energy-Based Probability Estimation. *arXiv:1606.03439*, 2016.

[109] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-Amortized Variational Autoencoders. In *ICML*, 2018.

[110] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *ICLR*, 2014.

[111] Durk P Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *NeurIPS 31*, 2018.

[112] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved Variational Inference with Inverse Autoregressive Flow. *NeurIPS 29*, 2016.

[113] I. Kobyzev, S. Prince, and M. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE TPAMI*, pages 2008–2026, 2020.

[114] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[115] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C. Courville. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. *NeurIPS*, 32, 2019.

[116] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. VideoFlow: A Flow-Based Generative Model for Video. In *ICML Workshop*, 2019.

[117] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum Entropy Generators for Energy-Based Models. *arXiv:1901.08508*, 2019.

[118] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

[119] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A Tutorial on Energy-Based Learning. *A Tutorial on Energy-Based Learning*, 1(0), 2006.

[120] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic Adversarial Video Prediction. *arXiv:1804.01523*, 2018.

[121] Zengyi Li, Yubei Chen, and Friedrich T. Sommer. Learning Energy-Based Models in High-Dimensional Spaces with Multi-scale Denoising Score Matching. *arXiv:1910.07762*, 2019.

[122] Zengyi Li, Yubei Chen, and Friedrich T. Sommer. A Neural Network MCMC sampler that maximizes Proposal Entropy. *arXiv:2010.03587*, 2020.

[123] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv:1705.02894*, 2017.

[124] Ji Lin, Richard Zhang, Frieder Ganz, Song Han, and Jun-Yan Zhu. Anycost GANs for Interactive Image Synthesis and Editing. *arXiv:2103.03243*, 2021.

[125] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards Faster and Stabilized GAN Training for High-fidelity Few-shot Image Synthesis. In *ICLR*, 2021.

[126] Kanglin Liu, Guoping Qiu, Wenming Tang, and Fei Zhou. Spectral Regularization for Combating Mode Collapse in GANs. In *IEEE ICCV*, 2019.

[127] Xuanqing Liu and Cho-Jui Hsieh. Rob-GAN: Generator, Discriminator, and Adversarial Attacker. In *IEEE CVPR*, 2019.

[128] Gabriel Loaiza-Ganem and John P. Cunningham. The continuous Bernoulli: fixing a pervasive error in variational autoencoders. *NeurIPS*, 32, 2019.

[129] Cheng Lu, Jianfei Chen, Chongxuan Li, Qiuhao Wang, and Jun Zhu. Implicit Normalizing Flows. In *ICLR*, 2021.

[130] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. *NeurIPS 31*, 2018.

[131] Yucen Luo, Alex Beatson, Mohammad Norouzi, Jun Zhu, David Duvenaud, Ryan P. Adams, and Ricky T. Q. Chen. SUMO: Unbiased Estimation of Log Marginal Probability for Latent Variable Models. In *ICLR*, 2020.

[132] Xuezhe Ma, Xiang Kong, Shanghang Zhang, and Eduard Hovy. MaCow: Masked Convolutional Generative Flow. *NeurIPS*, 2019.

[133] Xuezhe Ma, Chunting Zhou, and Eduard Hovy. MAE: Mutual Posterior-Divergence Regularization for Variational AutoEncoders. In *ICLR*, 2019.

[134] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling. *NeurIPS*, 33, 2019.

[135] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083*, 2019.

[136] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. *arXiv:1511.05644*, 2016.

[137] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. In *IEEE CVPR*, 2017.

[138] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. In *ICLR*, 2017.

[139] Chenlin Meng, Jiaming Song, Yang Song, Shengjia Zhao, and Stefano Ermon. Improved Autoregressive Modeling with Distribution Smoothing. In *ICLR*, 2021.

[140] Chenlin Meng, Lantao Yu, Yang Song, Jiaming Song, and Stefano Ermon. Autoregressive Score Matching. *NeurIPS*, 34, 2020.

[141] Jacob Menick and Nal Kalchbrenner. Generating High Fidelity Images with Subscale Pixel Networks and Multidimensional Upscaling. In *ICLR*, 2019.

[142] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784*, 2014.

[143] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *ICLR*, 2018.

[144] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural Importance Sampling. *ACM TOG*, 38(5):145:1–145:19, 2019.

[145] Charlie Nash and Conor Durkan. Autoregressive Energy Machines. In *ICML*, 2019.

[146] Kirill Neklyudov, Evgenii Egorov, and Dmitry P. Vetrov. The Implicit Metropolis-Hastings Algorithm. *NeurIPS*, 32, 2019.

[147] M. Ngxande, J. Tapamo, and M. Burke. DepthwiseGANs: Fast Training Generative Adversarial Networks for Realistic Image Synthesis. In *SAUPEC/RobMech/PRASA*, pages 111–116, 2019.

[148] Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. *arXiv:2102.09672*, 2021.

[149] Weili Nie, Nina Narodytska, and Ankit Patel. RelGAN: Relational Generative Adversarial Networks for Text Generation. In *ICLR*, 2019.

[150] Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows. *NeurIPS*, 33, 2020.

[151] Erik Nijkamp, Ruiqi Gao, Pavel Sountsov, Srinivas Vasudevan, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Learning Energy-based Model with Flow-based Backbone by Neural Transport MCMC. *arXiv:2006.06897*, 2020.

[152] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the Anatomy of MCMC-Based Maximum Likelihood Learning of Energy-Based Models. In *arXiv:1903.12370*, 2020.

[153] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning Non-Convergent Non-Persistent Short-Run MCMC Toward Energy-Based Model. *NeurIPS 32*, 2019.

[154] Erik Nijkamp, Bo Pang, Tian Han, Linqi Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning Multi-layer Latent Variable Model via Variational Optimization of Short Run MCMC for Approximate Inference. In *ECCV*, Lecture Notes in Computer Science, Cham, 2020.

[155] Sajad Norouzi, David J. Fleet, and Mohammad Norouzi. Exemplar VAE: Linking Generative Models, Nearest Neighbor Retrieval, and Data Augmentation. *NeurIPS*, 33, 2020.

[156] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *NeurIPS 29*, 2016.

[157] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *ICML*, 2017.

[158] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport. In *AAAI*, 2021.

[159] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *ICML*, 2018.

[160] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499*, 2016.

[161] Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive Quantile Networks for Generative Modeling. *ICML*, 9, 2018.

[162] A. Oussidi and A. Elhassouny. Deep Generative Models: Survey. In *IEEE ISCV*, pages 1–8, 2018.

[163] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning Latent Space Energy-Based Prior Model. *NeurIPS*, 34, 2020.

[164] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *JMLR*, 22(57):1–64, 2021.

[165] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *NeurIPS 30*, 2017.

[166] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image Transformer. In *ICML*, 2018.

[167] L. S. Pontryagin. *Mathematical Theory of Optimal Processes*. Routledge, 2018.

[168] R. Prenger, R. Valle, and B. Catanzaro. WaveGlow: A Flow-based Generative Network for Speech Synthesis. In *IEEE ICASSP*, pages 3617–3621, 2019.

[169] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR*, 2016.

[170] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation. *arXiv:2102.12092*, 2021.

[171] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. *NeurIPS 32*, 2019.

[172] Scott Reed, Aäron Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando Freitas. Parallel Multiscale Autoregressive Density Estimation. In *ICML*, 2017.

[173] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *ICML*, 2, 2015.

[174] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[175] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *NeurIPS*, 2016.

[176] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *ICLR*, 2017.

[177] Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *ICML*, 2015.

[178] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep Energy Estimator Networks. *arXiv:1805.08306*, 2018.

[179] Jürgen Schmidhuber. Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments. 1990.

[180] Jürgen Schmidhuber. Generative Adversarial Networks are special cases of Artificial Curiosity (1990) and also closely related to Predictability Minimization (1991). *Neural Networks*, 127:58–66, 2020.

[181] Jiaxin Shi, Shengyang Sun, and Jun Zhu. A Spectral Approach to Gradient Estimation for Implicit Distributions. In *ICML*, 2018.

[182] Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. Small-GAN: Speeding up GAN Training using Core-Sets. In *ICML*, 2020.

[183] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. *arXiv:2006.09661*, 2020.

[184] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *ICML*, 2015.

[185] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *ICLR*, 2021.

[186] Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-NICE-MC: Adversarial Training for MCMC. *NeurIPS*, 30, 2017.

[187] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. *NeurIPS 32*, 2019.

[188] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In *UAI*, pages 574–584, 2020.

[189] Yang Song and Diederik P. Kingma. How to Train Your Energy-Based Models. *arXiv:2101.03288*, 2021.

[190] Yang Song, Chenlin Meng, and Stefano Ermon. MintNet: Building Invertible Neural Networks with Masked Convolutions. *NeurIPS*, 32, 2019.

[191] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*, 2021.

[192] Yuxuan Song, Qiwei Ye, Minkai Xu, and Tie-Yan Liu. Discriminator Contrastive Divergence: Semi-Amortized Generative Modeling by Exploring Energy of the Discriminator. *arXiv:2004.01704*, 2020.

[193] Ilya Sutskever and Tijmen Tieleman. On the Convergence Properties of Contrastive Divergence. In *AISTATS*, pages 789–795, 2010.

[194] Kevin Swersky, Marc'Aurelio Ranzato, David Buchman, Benjamin M Marlin, and Nando de Freitas. On Autoencoders and Score Matching for Energy Based Models. In *ICML*, 2011.

[195] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP Inference for Image Super-resolution. In *ICLR*, 2017.

[196] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder Variational Autoencoders. *NeurIPS*, 29, 2016.

[197] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS*, 33, 2020.

[198] H. Thanh-Tung and T. Tran. Catastrophic forgetting and mode collapse in GANs. In *IJCNN*, pages 1–10, 2020.

[199] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv:1511.01844*, 2016.

[200] Michalis Titsias and Petros Dellaportas. Gradient-based Adaptive Markov Chain Monte Carlo. *NeurIPS*, 32, 2019.

[201] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein Auto-Encoders. *arXiv:1711.01558*, 2019.

[202] Jakub Tomczak and Max Welling. VAE with a VampPrior. In *AISTATS*, pages 1214–1223, 2018.

[203] Dustin Tran, Rajesh Ranganath, and David M. Blei. Variational Gaussian Process. In *ICLR*, 2016.

[204] Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete Flows: Invertible Generative Models of Discrete Data. *NeurIPS 32*, 2019.

[205] N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung. On Data Augmentation for GAN Training. *IEEE TIP*, 30:1882–1897, 2021.

[206] Ngoc-Trung Tran, Viet-Hung Tran, Bao-Ngoc Nguyen, Linxiao Yang, and Ngai-Man (Man) Cheung. Self-supervised GAN: Analysis and Improvement with Multi-class Minimax Game. *NeurIPS*, 32, 2019.

[207] Richard Eric Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time series models. In *Bayesian Time Series Models*, pages 104–124. Cambridge, 2011.

[208] Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-Hastings Generative Adversarial Networks. In *ICML*, 2019.

[209] Belinda Tzen and Maxim Raginsky. Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit. *arXiv:1905.09883*, 2019.

[210] Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *COLT*, pages 3084–3114, 2019.

[211] Arash Vahdat and Jan Kautz. NVAE: A Deep Hierarchical Variational Autoencoder. *NeurIPS*, 33, 2020.

[212] Gabriele Valvano, Andrea Leo, and Sotirios A Tsaftaris. Learning to Segment from Scribbles using Multi-scale Adversarial Attention Gates. *IEEE T-MI*, 2021.

[213] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional Image Generation with PixelCNN Decoders. *NeurIPS 29*, 2016.

[214] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In *ICML*, 2016.

[215] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural Discrete Representation Learning. *NeurIPS 30*, 2017.

[216] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *NeurIPS 30*, 2017.

[217] Cédric Villani. *Topics in Optimal Transportation*. Number 58. American Mathematical Soc., 2003.

[218] Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.

[219] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[220] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv:2006.04768*, 2020.

[221] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On Solving Minimax Optimization Locally: A Follow-the-Ridge Approach. In *ICLR*, 2020.

[222] Antoine Wehenkel and Gilles Louppe. Unconstrained Monotonic Neural Networks. *NeurIPS*, 32, 2019.

[223] Max Welling and Geoffrey E. Hinton. A New Learning Algorithm for Mean Field Boltzmann Machines. In *ICANN*, volume 2415, pages 351–357. Berlin, Heidelberg, 2002.

[224] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *ICML*, 2011.

[225] Hao Wu, Jonas Köhler, and Frank Noé. Stochastic Normalizing Flows. *NeurIPS*, 2020.

[226] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. VAEBM: A Symbiosis between Variational Autoencoders and Energy-based Models. In *ICLR*, 2021.

[227] J. Xie, Y. Lu, R. Gao, S. Zhu, and Y. N. Wu. Cooperative Training of Descriptor and Generator Networks. *IEEE TPAMI*, 42(1):27–45, 2020.

[228] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A Theory of Generative ConvNet. In *ICML*, 2016.

[229] L. Yang and G. E. Karniadakis. Potential Flow Generator With L2 Optimal Transport Regularity for Generative Models. *IEEE TNNLS*, pages 1–11, 2020.

[230] Xin Yi, Ekta Walia, and Paul Babyn. Generative Adversarial Network in Medical Imaging: A Review. *MedIA*, 58, 2019.

[231] Lantao Yu, Yang Song, Jiaming Song, and Stefano Ermon. Training Deep Energy-Based Models with f-Divergence Minimization. In *ICML*, 2020.

[232] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv:1605.07146*, 2017.

[233] Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation Capabilities of Neural Ordinary Differential Equations. *arXiv:1907.12998*, 2019.

[234] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. *arXiv:1805.08318*, 2019.

[235] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. StackGAN: Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks. In *IEEE CVPR*, 2017.

[236] Linfeng Zhang, Weinan E, and Lei Wang. Monge-Ampère Flow for Generative Modeling. *arXiv:1809.10188*, 2018.

[237] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based Generative Adversarial Network. In *ICLR*, 2017.

[238] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning Hierarchical Features from Deep Generative Models. In *ICML*, 2017.

[239] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards Deeper Understanding of Variational Autoencoding Models. *arXiv:1702.08658*, 2017.

[240] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Balancing Learning and Inference in Variational Autoencoders. *AAAI*, 33(01):5885–5892, 2019.

[241] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable Augmentation for Data-Efficient GAN Training. *NeurIPS*, 33, 2020.

[242] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image Augmentations for GAN Training. *arXiv:2006.02595*, 2020.

[243] Jiachen Zhong, Xuanqing Liu, and Cho-Jui Hsieh. Improving the Speed and Quality of GAN by Adversarial Training. *arXiv:2008.03364*, 2020.

[244] Zachary M. Ziegler and Alexander M. Rush. Latent Normalizing Flows for Discrete Sequences. In *ICML*, 2019.