MEGAPIXEL IMAGE GENERATION WITH STEP-UNROLLED DENOISING AUTOENCODERS

Alex F. McKinney & Chris G. Willcocks
Department of Computer Science
Durham University
Durham, UK
{alexander.f.mckinney,christopher.g.willcocks}@durham.ac.uk

ABSTRACT

The recent trend in generative modelling research has been to push sample resolutions higher whilst simultaneously reducing computational requirements for training and sampling. We aim to push this trend further via the combination of various recent advancements in deep learning – each representing the pinnacle of efficiency in their respective areas. These include vector-quantized GAN (VQ-GAN), a vector-quantization (VQ) model capable of high levels of compression; hourglass transformers, a highly scaleable self-attention model; and step-unrolled denoising autoencoders (SUNDAE), a non-autoregressive (NAR) text generative model. Unexpectedly, our method highlighted weaknesses in the original formulation of hourglass transformers when used on multi-dimensional data. In light of this, we propose multiple modifications to their architecture, which has applications in the wider field of multi-dimensional data modelling using hierarchical transformers. The combination of these techniques resulted in a highly efficient generative model framework, in terms of scalability of our model and sampling speed. Our proposed framework scales to high-resolutions (1024×1024) easily, and can be trained quickly (4-6 days) on a single, consumer-grade GPU. Crucially, the trained model can produce diverse and realistic megapixel samples in approximately 2 seconds. This is considerably faster sampling (minutes, or even tens of minutes) and training (> 10 days) than existing non-adversarial methods at this resolution. In general, the framework is flexible: supporting an arbitrary number of sampling steps, sample-wise self-stopping, self-correction capabilities, conditional generation, and a NAR formulation that allows for arbitrary inpainting masks.

Figure 1: High-resolution samples produced using our non-autoregressive approach. Each 1024×1024 sample was generated in ≈ 2 seconds on a GTX 1080Ti – including both discrete latent sampling and subsequent decoding. At this resolution, autoregressive and diffusion models may take minutes to sample, or simply do not support this resolution. Training is similarly fast, with the FFHQ256 model only taking 4 days to train on a single V100 32GB.

1 INTRODUCTION

An ideal deep generative model satisfies three key requirements: high-quality samples, mode coverage leading to high sample diversity, and computational inexpensive sampling. Arguably, there are other desirable properties such as a meaningful latent space, exact likelihood calculation, and controlled generation. Nonetheless, no current generative model satisfies all three key requirements – let alone additional attractive properties – forming the so-called generative modelling trilemma [57] that dominates modern generative modelling research.

Models such as generative adversarial networks (GANs) [19] excel at high-quality and fast sampling, but often fail to model the entire data distribution due to not directly optimising for likelihood – using an adversarial loss as a proxy.

Variational auto-encoders (VAEs) [29] offer excellent mode coverage and fast sampling speeds, but the resulting samples are blurry, even at small resolutions, and cannot scale to higher resolutions like a GANs.

Autoregressive (AR) models such as PixelSnail [6], Image Transformer [36], and DALL·E [36] have demonstrated respectable sample quality and mode coverage, even extending to zero-shot image generation [39]. However, they are computationally expensive to sample from, requiring many network iterations to produce a single sample. This makes them infeasible for interactive applications. Denoising diffusion probabilistic models (DDPMs) [21] and score-based models (SBMs) [45, 46, 47] produce samples that rival or even exceed the quality of GANs [13] whilst providing good mode coverage, but are still plagued by potentially requiring thousands of network evaluations.

Vector-quantized image modelling [53, 40, 17] alleviates sampling speed issues in AR methods by reducing the spatial dimension at which discrete AR priors operate at. This results in excellent quality samples whilst improving sampling speeds, but mandates a two-stage training approach and does not match the sampling speed of GANs. Recent work has applied diffusion models to VQ latents [3] allowing for fast parallel sampling. Other recent work uses continuous latent spaces to accelerate sampling [57, 50].

From this brief overview of generative modelling, it is clear that indeed no single model satisfies all three conditions. This motivates research into explicitly addressing this trilemma. In this work, we move towards such a solution, beginning from existing work applying generative models to discrete latents. This provides an excellent starting point in terms of sample quality and a respectable amount of mode coverage, but an unacceptably slow sampling speed despite the reduced spatial dimension of the discrete latent space. We address this issue by instead sampling discrete latents using modern non-autoregressive (NAR) generative models in an effort to close the gap, sampling speed wise, with constant iteration complexity models such as GANs. Specifically, we use discrete step-unrolled denoising autoencoders (SUNDAE) [43] to gradually denoise samples from a uniform prior into one that matches the prior over the discrete latent space defined by a pre-trained vector-quantized GAN (VQ-GAN) model. We find that SUNDAE are an effective discrete prior over VQ-GAN latents.

SUNDAE has only previously be applied to language modelling tasks [43] using transformer [55] architectures to implement their autoencoder. Parallel work introduced a drastically more efficient variant of transformers – the Hourglass Transformer [35] – , leveraging a hierarchical architecture aimed at language modelling. Though able to be applied to discrete latent modelling, we propose a number of improvement that improve performance on multi-dimensional discrete data, including modifications to resampling operations and introduction of axial positional embeddings. Though evaluated on discrete latents, these modifications are also applicable in any scenario with multi-dimensional inputs, which will be valuable in a wider context outside of generative modelling. We also found recommended parameters proposed in Savinov et al. [43] did not always generalise to multi-dimensional data, so we explore new sensible defaults in this work.

Given a fast sampling and efficient transformer architecture, we now possess a highly scaleable generative model, with respect to number of layers and spatial resolution of the input. Only a minority of the layer are operating at the same resolution as the input, reducing the cost of expensive self-attention across the input. Conversely, we can scale the number of layers more cheaply by adding layers only at the downsampled resolution, allowing for considerably larger models with a minor computational cost. To demonstrate the scalability of our approach, we train a VQ-GAN operating on 1024×1024 images and apply our framework to the resulting discrete latent codes. This ultimately resulted in the synthesis of megapixel images in as few as 2 seconds on a consumer-grade GPU. To our knowledge, this is the largest VQ-GAN trained in terms of input size, and the fastest sampling, non-adversarial generative framework at this image resolution.

Our contributions as a result of this research project are as follows:

- The development of a non-autoregressive, non-adversarial generative modelling framework with extremely flexible sampling including self-correction and arbitrary inpainting pattern capabilities. The model can be directly configured for both low- and high-step sampling scenarios, resulting in high quality and diverse samples in mere seconds of sampling time.

- Modifications to methods proposed in Savinov et al. [43] and Nawrot et al. [35] to be more suited for the modelling of multi-dimensional discrete data. Though applied to discrete latents in our work, the modifications are also applicable in a wider context, such as to pixel-level modelling. We also demonstrate the superiority of hierarchical transformers – forming a key component in the scalability of our approach.

- The scaling of VQ-GAN [17] to extremely high resolution images of human faces. 1024×1024 images far exceeds the resolutions of prior work. This ultimately allowed for the **generation of megapixel images in as few as two seconds** on a consumer-grade GPU when combined with our fast and scalable generative

framework. This is in contrast to prior AR methods and NAR diffusion methods that take minutes to generate, or have not scaled to such resolutions entirely.

2 Related Work

This work builds upon much prior research into powerful deep generative models, self-supervised methods, and efficient transformer architectures. We cover relevant prior work into deep generative models in §2.1-2.3, a specific NAR generative model SUNDAE §2.4, and a recent development into a efficient transformer architecture in §2.5. For a full review on generative modelling we direct the reader to Bond-Taylor et al. [4], and for further details on SUNDAE and hourglass transformers to Savinov et al. [43] and Nawrot et al. [35] respectively.

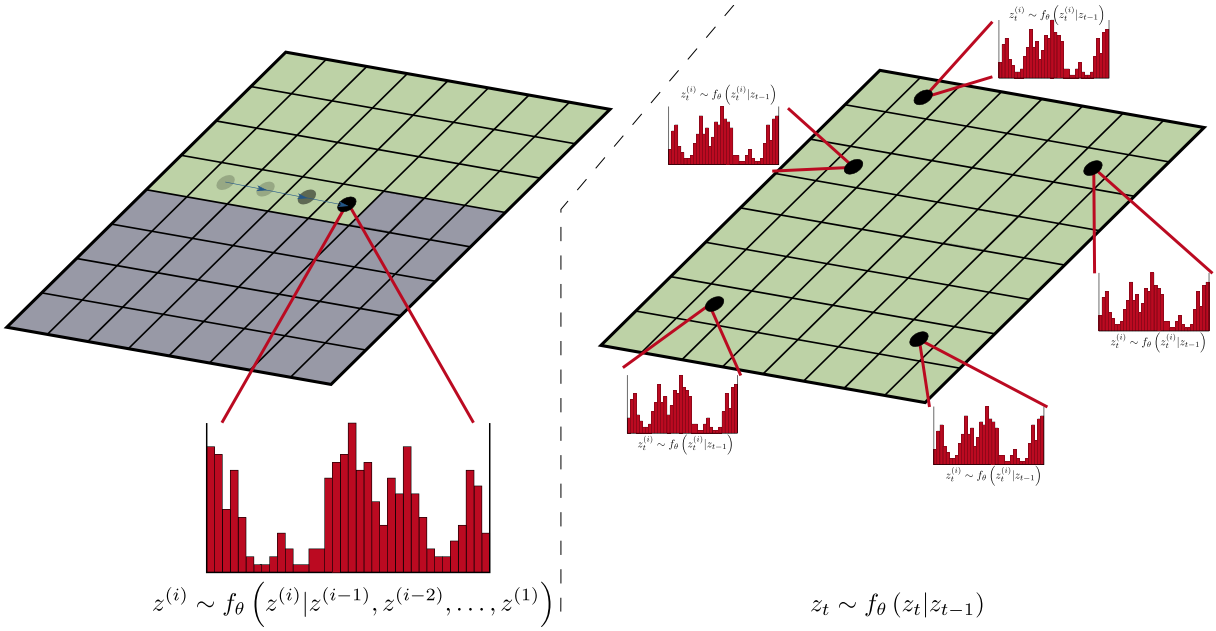


Figure 2: **Left:** Visualization of autoregressive (AR) sampling. AR sampling proceeds one item at a time, resulting in the number of sampling steps being equal to the dimensionality of the input. For each prediction, a probability distribution over possible tokens is predicted and then sampled from. Each prediction can only make use of past context – indicated as a green position – so not to violate the autoregressive property. **Right:** Visualization of non-autoregressive (NAR) sampling. NAR sampling can sample an arbitrary number of items in parallel, including ones previously sampled, allowing for self-correction. It can freely use all context available to it, allowing for flexible inpainting and potentially better predictions.

2.1 Autoregressive Generative Models

One major deep generative model family are autoregressive (AR) models, characterised by a training and inference process based on the probabilistic chain rule. During training, they learn to maximise the likelihood of the data they are trained on, which leads to excellent mode coverage. Prior work using these methods resulted in impressive results in terms of both sample quality and diversity, but are ultimately impractical in real world applications due to their slow sampling speed.

The slow sampling speed is due to their sequential nature, defined by the chain rule of probability. Given an input $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, an AR model $p_\theta(\cdot)$ generates new samples sequentially:

$$p_\theta(\mathbf{x}) = p_\theta(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i|x_1, \ldots, x_{i-1}) \tag{1}$$

meaning that the number of sampling steps is equal to the size of the decomposition of $\mathbf{x}$, making this slow for large inputs.

For certain tasks, the ordering of the decomposition of $\mathbf{x}$ is obvious, for example on text or speech. For images this is less obvious, however typically a raster scan ordering is used [36]. Certain AR models are order-agnostic [23], allow for arbitrary ordering to be used during training and inference.

One class of AR models are recurrent neural networks (RNNs) which are an early example of using neural networks to model sequential data, such as text, audio, time-series data, or even handwriting strokes. Though they can be used as classification or regression models, they are also suited for use as generative models by modelling the relationship in Equation 1. They suffer from a number of issues, most infamously vanishing gradients [37] and inability to model long-range relationships between items in the input. Long short-term memory recurrent neural networks (LSTMs) [22] improved upon RNNs further by introducing dedicated memory units, allowing for the modelling of longer range relationships. Gated recurrent units (GRUs) [8] simplified LSTM architecture whilst retaining good performance. With the advent of transformer architectures [55], modelling even longer relationships became possible, even at a full-document level. It also introduced the capability to train on all sequence elements in parallel through the use of causal masking, therefore not violating the autoregressive property. Inference must still be done in a serial manner, however.

Applying AR models to images followed a similar trend. PixelRNN [51] used two-dimensional recurrent layers and residual connections to model the distribution of raw pixel values. The same paper also introduced PixelCNN, which was claimed to have worse performance but were faster to train. These were extended to allow for conditional generation in [52]. Later work augmented PixelCNN with self-attention mechanism, forming PixelSnail [6], which can model longer relationships than a convolutional or recurrent architecture. Image Transformer [36] later applied transformer architectures to the same task through an effective but conceptually simple approach.

2.2 Non-autoregressive Generative Models

Non-autoregressive (NAR) generative models include GANs, SBMs and DDPMs, flow-based models, energy-based models (EBMs), and implicit models. Though the number of sampling steps is now independent of the data dimensionality (as we no longer use the chain rule of probability to sample) the actual number of steps required varies greatly: from single-step generation in GANs to potentially many thousands in early diffusion models.

Perhaps the most infamous class of NAR generative model – or perhaps generative model altogether – are generative adversarial networks (GANs) [19]. GANs consist of two models: a generator that creates images from a latent variable, and a discriminator that attempts to distinguish images from the dataset from images generated by the generator network [19]. They are known for high-fidelity samples, fast sampling, unstable training, and tendency to collapse onto a subset of modes of the underlying distribution due to not optimising directly for likelihood. This is reflected in its relatively low-diversity samples. Nonetheless, the quality of the samples has made them a popular choice in a variety of applications, including unconditional and conditional generation [26, 5] image, audio synthesis [32], and style transfer [60]. They can even be applied to discrete data [11], but are less effective on such domains due to the non-differentiability of discrete samples.

Another class of generative model are variational auto-encoders (VAEs) [29] which permit sampling in a single forward pass like GANs, but are trained to maximise likelihood. Specifically, VAEs map inputs to latent variables that follow some easy to sample from, but sufficiently complex, prior distribution. A common choice is a multivariate Gaussian with diagonal covariance [29]. A decoder network maps latent codes back to the data distribution. Although this approach is successful on small datasets, on more complex datasets the samples and reconstructions tend to become blurry, suggesting a simple prior is unable to perfectly fit the distribution. Later work extended VAEs to be hierarchical, having multiple Gaussian priors [49, 7] which were found to outperform purely autoregressive models.

Normalizing flows are another class of generative model that allows for exact likelihood calculation [14, 15, 30]. They consist of many invertible layers that gradually transform samples from a known prior distribution into samples from the data distribution. Each transformation must satisfy two properties: being invertible and having an easy to compute Jacobian for the purposes of scaling [14, 15]. This makes the architectural choices restrictive, making them less parameter efficient. They also typically must operate at the same dimensionality for each layer, making the training of deep networks difficult.

Two paradigms of high interest in recent work are denoising diffusion probabilistic models (DDPMs) [21, 13] and score-based models (SBMs) [45, 46, 47, 50]. Both are variants of EBMs, with the former learning to estimate the noise at various levels (which can be used to iteratively move from noise to data), and the latter trained to remove noise, given corrupted samples from a gradually corrupting forward process. Again, this is then used to move from pure noise back to data. Both are slow to sample from, but produce high quality samples that rival those of GANs [13] whilst not suffering from mode collapse. The slow sampling speed can be remedied using a variety of techniques, such as operating on a smaller latent space [50], devising more efficient SDE solvers [25], or by diffusing "velocities", thus

simplifying the denoising task [16]. Unlike GANs, both models can operate on discrete data, such as by first projecting discrete data into a continuous latent space [50] or with a dedicated discrete diffusion framework [1], the latter of which bridges the gap between diffusion, autoregressive, and mask-based representation models such as BERT [12].

2.3 Vector Quantized Image Modelling

Learning useful latent representations, also known as latent codes, in an unsupervised manner is a key challenge in machine learning. Historically, these representations have been in a continuous form, but in recent work they are often discrete. An early example of this is vector-quantized VAE (VQ-VAE) [53], a variant on VAE representation models. A VQ-VAE has three main components: an encoder network, a codebook, and a decoder. The encoder network outputs a compressed representation of the input, and the codebook $\mathcal{C}$ quantizes these representations, outputting a discretized representation of indices from 1 to the codebook size v. Each index i maps to one of the codebook embeddings e_i. The decoder then maps the quantized embeddings back to the original signal, training it in tandem to reconstruct the input signal and to minimize additional codebook loss terms [53]. Once a trained VQ-VAE model has been produced, a powerful auxiliary generative model can be trained to generate these discrete latent representations. The decoder can then generate the final sample given the discrete latent representation. In the original work on VQ-VAE, they use a PixelCNN model to generate the discrete latent codes [53], though any generative model on discrete data can be used.

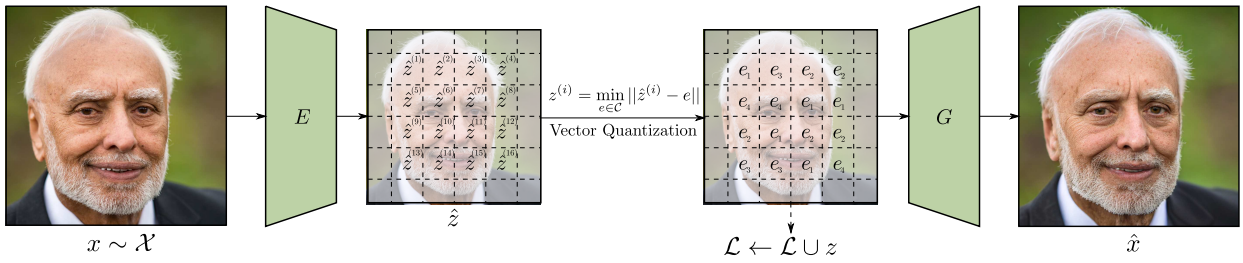


Figure 3: Visualisation of a vector-quantization image model. An encoder model first extracts continuous embeddings from the source image. Vector quantization is then used to map each continuous embedding to the closest entry in the codebook. A decoder model takes the discretized embeddings and attempts to reconstruct the original image. We can produce a dataset of latent embeddings from a source image dataset by sampling $\mathbf{x} \sim \mathcal{X}$ and appending the resulting discretized embeddings $\mathbf{z}$ to a dataset $\mathcal{L}$.

Later approaches extended VQ-VAE to multiple distinct codebooks in VQ-VAE-2 [40]. Though theoretically it can be extended to any number of codebooks, they experimented on two-level and three-level models, applying the latter to 1024×1024 images. They then sampled the resulting combined discrete codes using multiple PixelSnail [6] models, each code conditioned on all previous levels in the hierarchy. Though faster to sample than applying a generative directly to pixels, at such a resolution and with multiple levels to sample, sampling times were still very slow.

Aside from sampling autoregressively, a reason for the slow sampling speed was the large spatial resolution of the discrete codes. Albeit smaller than the original signal, VQ-VAE is limited in how much it can compress the signal via a simple reconstruction objective before it loses too much perceptual quality due to the rate-distortion trade-off. For example, the original VQ-VAE only has a downsampling rate of $f = 4$ [53] and VQ-VAE-2 has a top-level rate of $f = 32$ but requires a total of three discrete latent codes in order to achieve this [40]. By introducing perceptual and adversarial loss terms, VQ-GAN [17] is able to achieve compression rates of $f = 16 \sim 32$ with only a single discrete latent representation whilst retaining high quality reconstructions [17]. However, the greater weighting on perceptual and adversarial loss does mean that VQ-GAN sometimes edits its reconstructions, rather than reconstruct faithfully. Later, improvements such as differential data augmentation [3], codebook improvements, and a transformer-based architecture [58] improved reconstruction quality further, but still rely on adversarial components.

Originally designed for audio compression [59], Residual VQ proposes the use of multiple codebooks to recursively quantize and refine the residual of an input signal. This produces multiple discrete representations, which can later be reconstructed by the decoder to decompress the waveform. Additionally, individual codebooks can be dropped out, allowing for variable bit-rates [59]. Concurrently to our work, Lee et al. [31] used residual VQ to represent images with a compression ratio of $f = 32$ and then trained a transformer model to autoregressively predict the stack of discrete tokens at a given spatial location, allowing for fast sampling despite having multiple levels of discrete latent representations[31]. A summary of each VQ model and their associated latent and codebook sizes are shown in Table 1.

Model	Input Size	Latent Shape	Codebook Size	FID/val	FID/train
VQ-VAE	128×128	32×32	512	–	–
VQ-VAE-2	256×256	64×64 & 32×32	512	–	~ 10
	1024×1024	128×128 & 64×64 & 32×32	512	–	–
DALLE	256×256	32×32	8192	32.01	33.88
VQ-GAN	256×256	16×16	1024	7.94	10.54
	256×256	16×16	16384	4.98	7.41
	256×256	32×32	8192	1.49	3.24
	256×256	64×64 & 32×32	8192	1.45	2.78
RVQ-VAE	256×256	$8 \times 8 \times 2$	16384	–	10.77
	256×256	$8 \times 8 \times 4$	16384	–	3.20
	256×256	$8 \times 8 \times 8$	16384	–	2.69
	256×256	$8 \times 8 \times 16$	16384	–	1.83

Table 1: Summary of various vector-quantization (VQ) methods. The trend with VQ image models is increasing compression rates f and codebook sizes. However, to obtain these compression rates techniques such as perceptual and adversarial losses must be used.

A typical strategy for selecting which codebook vector e_i to map to a particular input is to compute the Euclidean distance between a given continuous input and the codebook centroid, and then pick the $\arg\min$ [53]. This is denoted the k-means strategy. This strategy can result in "codebook collapse", where certain codebook vectors never get used, limiting the effective capacity of the model. An alternative method is to use Gumbel-Softmax [24] to select codebook vectors, which increases codebook utilisation but leads to worse reconstruction quality [3].

The issue of codebook collapse is significant and there have been a number of attempts to remediate it. [58] found that a lower codeword dimension and codeword normalization improved utilisation. [59] proposed setting a threshold for "stale" codes, and reinitialising them to a random vector from the current batch when they fall below this threshold. [31] proposed the sharing of a single codebook across many quantizers and additionally stochastically sampled the codes as a function of their distance to the centroid, rather than deterministically returning the $\arg\min$.

All previously described approaches use VQ models to enhance existing AR models, primarily to improve their sampling speed by reducing the spatial dimension we are operating over. Little work directly addresses the discrete prior model. Discrete diffusion models [1] are NAR approach to generate discrete data. This was applied to VQ-GAN latents in Bond-Taylor et al. [3], allowing for fast sampling, flexible inpainting, and high fidelity outputs.

2.4 Step-unrolled Denoising Autoencoder

One recent NAR model is SUNDAE [43] which was evaluated on three language modelling tasks: unconditional text-generation, inpainting of Python code, and machine translation – setting a new state-of-the-art among NAR models for the machine translation task [43]. It demonstrated exceptionally fast sampling, producing high quality text samples in as few as 10 steps.

SUNDAE is trained using a denoising objective, akin to the BERT denoising objective [56] but with multiple denoising steps. Given a uniform prior p_0 over some space $Z = \{1, \dots, v\}^N$ where N is the size of the space and v is the vocabulary size, consider the Markov process $\mathbf{z}_t \sim f_\theta(\cdot|\mathbf{z}_{t-1})$ where f_θ is a neural network parameterised by θ, then $\{\mathbf{z}_t\}_t$ forms a Markov chain. This gives a t-step transition function:

$$p_t(\mathbf{z}_t|\mathbf{z}_0) = \sum_{\mathbf{z}_1,\dots,\mathbf{z}_{t-1} \in Z} \prod_{s=1}^{t} f_\theta(\mathbf{z}_s|\mathbf{z}_{s-1}) \tag{2}$$

[43] and, given a constant number of steps T, our model distribution $p_T(\mathbf{z}_T|\mathbf{z}_0)p_0(\mathbf{z}_0)$ – which is clearly intractable.

Instead, they propose an *unrolled denoising* training method that uses a far lower T than is used for sampling [43]. To compensate, they unroll the Markov chain to start from corrupted data produced by a *corruption distribution* $\mathbf{z}' \sim q(\cdot|\mathbf{z})$ rather than from the prior p_0 so the model encounters samples more akin to those seen during the full unroll at sample time [43]. Typically, $T = 2$ during training, as a single step would be similar to the training strategy of BERT [12] which not be performant as seen in earlier work using BERT as a random field language model [56].

The training objective of SUNDAE is simply the average of all reconstruction losses of the chain after t steps, which is shown to form an upper bound on the actual negative log-likelihood [43]. Taking more steps T leads to a minor improvement in performance, but considerably slows down training time [43] and increases memory usage.

One advantage of this approach is that sampling starts from random tokens, rather than a dedicated "masking" token [3, 1]. Unmasking approaches means that $T \leq N$ as at minimum, one token is unmasked per step. Additionally, it allows the model to be able to "change its mind" about previously predicted positions during sampling, allowing it to make fine-grained adjustments or fix accumulated errors.

2.5 Hourglass Transformers

Vanilla transformers incur a hefty memory and time complexity of $O(L^2)$ for each block [55]. This is largely due to the multi-head self-attention mechanism, as each input position must attend to every other. Most research into efficient transformers focuses on improving the efficiency of these attention mechanism, such as through sparse attention patterns or approximations of attention.

Recent work however, is now focusing on making the overall architecture more efficient. Funnel-Transformer [10] progressively downsamples the input sequence and hence reduces the computational cost of the model. The saved floating point operationss (FLOPs) can then be reassigned to create deeper or wider models and thus outperform vanilla transformers given the same computational budget [10]. However, the final layer does not operate at the same granularity as the input, making it unusable for tasks that require this such as per-token classification or generative tasks. Hourglass transformers [35] include both up- and down-sampling mechanisms, resulting in a computational saving whilst still being general-purpose models.

2.6 Summary of Trends in Deep Generative Modelling

Deep generative modelling research is a fast moving, complex and turbulent field to follow. Nonetheless, it is worthwhile to distill advancements into a selection of high level trends.

Improving quality and efficiency in non-adversarial generative models – GANs have long dominated as the pinnacle of sample quality and efficiency. Despite this, they are plagued by the previously discussed issues, motivating research into non-adversarial approaches of equal quality to GANs. Certain non-adversarial solutions do beat GANs in quality, but GANs still dominate in terms of sample speed, making them still the standard model in real-world generative modelling applications – bar cutting edge applications [39, 18]. Hence the trend is the simultaneous improvement in efficiency and quality, as it is clear that without both, adoption in practise will not occur. Satisfying these speed and quality requirements with a non-adversarial solution would in turn satisfy the so-called Generative Modelling Trilemma [57].

Autoregressive to non-autoregressive models – NAR models offer a number of advantages over AR models as discussed in earlier sections. It is clear that NAR solutions will soon be a default over AR methods, seen by observing initial proposed methods and subsequent improvements often replace AR components with NAR components. For example: sampling VQ-GAN latents with AR transformers [17] to discrete diffusion models [3]; DALL·E using AR sampling [39] to DALL·E 2 using diffusion models [18].

Class conditional to zero-shot conditional generative models – One especially popular trend is the shift from specifying a set of predefined classes when conditioning a model, to full zero-shot generation. This is usually realised by jointly learning text and image embeddings, then encoding a text prompt at test time to condition the model [39, 18, 41, 31]. This allows for a much higher degree of creative control – an all too often overlooked property – and can be easily integrated with existing architectures. However, they require large amounts of labelled images and associated text prompts, usually scraped from the internet [41, 39, 18]. This makes bias and dangerous content introduced by the training data much harder to control and filter, potentially influencing the resulting samples [33]. Nonetheless, it is clear that the natural language conditioning of generative models will persist.

Use of a vector-quantized discrete prior – Though VQ models have helped produce excellent results both within generative modelling [40, 17, 3, 41, 39, 58, 31] and elsewhere [59], excellent results can also be obtained without the use of them [7, 49, 20], especially with diffusion models [45, 46, 13, 47, 57, 50, 25, 16]. A particularly powerful advantage of using a VQ-based approach is the compression of the spatial resolution that the generator model must operate over, aiding in scaling to higher resolutions and improving sampling speeds. However, the two-stage training approach can be unwieldy in the absence of pretrained VQ models and additionally makes inpainting more involved. Additionally, they are plagued with issues such as codebook collapse [17, 3, 58], limiting the potential capacity of the model. It remains to be seen whether VQ models will continue being a key component in the generative modelling pipeline, or be rendered obsolete.

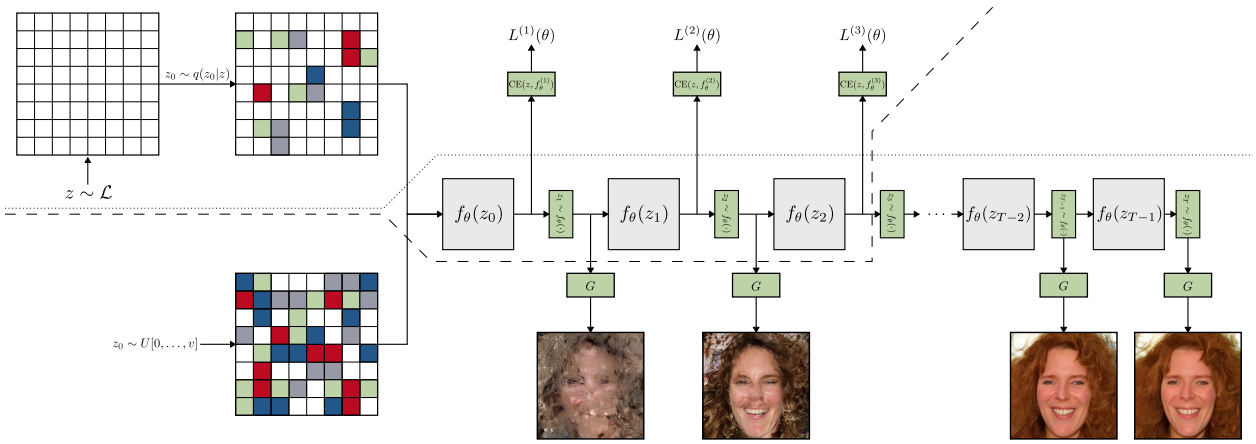


Figure 4: An overview of the SUNDAE training and sampling of discrete latent representations. Above the dashed line represents the process for training, whereas below the dotted line represents the sampling process. The training process begins by sampling $\mathbf{z} \sim \mathcal{L}$ and then sampling from the corruption distribution $q(\mathbf{z}_0|\mathbf{z})$. SUNDAE then denoises for 2 to 3 steps, computing the cross-entropy loss at each step in the chain which is subsequently averaged to produce a final loss. Sampling begins by obtaining $\mathbf{z}_0$ from a uniform prior and iteratively denoising for more steps than trained with. At each step in the chain, the sample $\mathbf{z}_t$ can be decoded by the VQ-GAN decoder G to obtain $\mathbf{y}_t$.

3 Methodology

Our proposed method aims to push the efficiency of generative models to the limit via a combination of current techniques. To do so, we first use pretrained VQ-GANs models from [17] to generate a dataset of discrete latent representations, described in §3.1. By operating at a latent level, we reduce the spatial resolution for our second stage generator. We implement our generator as a modified hourglass transformer, described in §3.2 and trained using a NAR method described in §3.4. This permits extremely fast sampling described in §3.5. To thoroughly test the efficiency and scalability of our approach, we train a megapixel VQ-GAN (described in §3.3) and repeat SUNDAE training and sampling on the resulting discrete latent representations. In §3.6, we explore flexible inpainting using our framework. An overview of training and sampling is shown in Figure 4, and of the latent dataset generation in Figure 3.

Each component represents the pinnacle of performance in their respective areas: compression ratio in VQ image models with VQ-GAN, fast non-autoregressive sampling of discrete data with SUNDAE, and transformer scalability with our modified hourglass transformer. Together, we obtain an extremely efficient generative model that permits sampling at 1024×1024 resolution in mere seconds.

3.1 Latent Dataset Generation

We use the standard two-stage scheme for vector-quantized image modelling [54, 40, 17, 3] using VQ-GAN [17] as our feature extractor. Where such models are available, we use pretrained VQ-GANs for our experiments. For higher resolution experiments (for example, FFHQ1024 [27]), pretrained models are not available and so the training our own VQ-GAN was necessary (see §3.3).

The second stage is to train a discrete prior model over the extracted latent variables. To enable this, we first built a latent dataset using our trained VQ-GAN. This allows for faster training of our second-stage model as the discrete latent representations have been precomputed. A downside of this approach is that it limits the amount of data augmentation that can be applied to the dataset. We apply a simple horizontal flip to all images, effectively doubling the dataset size, with no other augmentation. Formally, given a dataset of images $\mathcal{X}$, a VQ-GAN encoder E with downsample factor f, and VQ codebook $\mathcal{C}$ with number of codewords v, trained on $\mathcal{X}$, we define our latent dataset $\mathcal{L}$ as:

$$\mathcal{L} = \{\mathcal{C}(E(\mathbf{x})) \mid \mathbf{x} \in \mathcal{X}\} \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$ is a single element of the augmented image dataset and $\mathbf{z} = \mathcal{C}(E(\mathbf{x})) \in \{1, \ldots, v\}^{h \times w}$ is the corresponding discrete latent representation. In other words, each $f \times f$ pixels in $\mathbf{x}$ is mapped to a single discrete value from 1 to v (which in turn, corresponds to a vector $\mathbf{e} \in \mathcal{C}$), resulting in a latent representation of shape $\frac{H}{f} \times \frac{W}{f} = h \times w$.

We then use $\mathcal{L}$ to train a discrete prior over the latents. Coupled with the VQ-GAN decoder G, we obtain a powerful generative model by first sampling from a discrete uniform prior distribution, iteratively denoising using SUNDAE,

and then decoding the final latents using the VQ-GAN decoder. The training this discrete prior model, forms the bulk of our work in this paper.

3.2 2D-Aware Hourglass Transformer

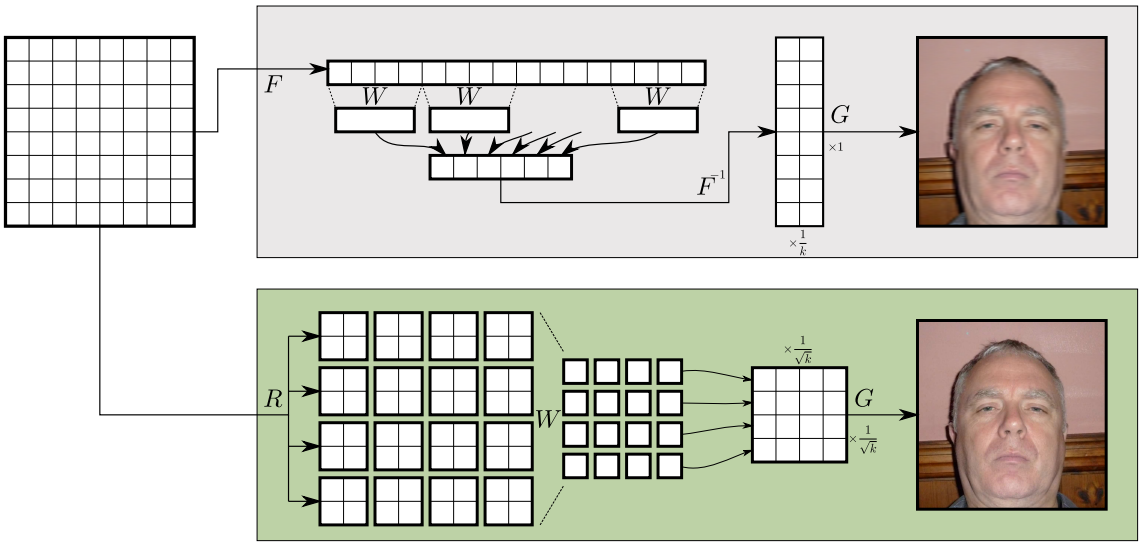


Figure 5: **Top:** Showing effect of resampling sequence embeddings using original formulation. Resampling rate will be applied to only one axis, resulting in resampling more in only one axis of the decoded image. **Bottom:** Our corrected method of resampling, extracting first two-dimensional patches of size $\sqrt{k}$, then applying resampling. The sequence is then again be flattened and passed to subsequent transformer layers.

Inspired by successes in hierarchical transformers for generative language modelling [35], we modify their architecture for use with discrete latent representations of image data. We will later use this architecture to implement a discrete prior over the VQ-GAN latents.

Hourglass transformers have been shown to efficiently handle long-sequences, outperform existing models using the same computational budget, and meet the same performance as existing models more efficiently by using an explicit hierarchical structure [35]. The same benefits should also apply to vector-quantized image modelling.

However, the design and parameters chosen by the original authors were tailored for language modelling [35]. They also experimented with pixel-wise image generation, though we believe that we can improve upon their architectural choices for the task of discrete latent modelling. Some changes may also be applicable to pixel-level generation. In this subsection, modifications to the architecture are outlined.

2D-Aware Downsampling – The original formulation of hourglass transformers [35] introduced both upsampling and downsampling layers, allowing the use of hierarchical transformers in tasks that have output sequence lengths equal to the input sequence lengths. However, we found certain flaws in their original formulation that hinders performance on multi-dimensional inputs.

In their work, resampling is applied to the flattened embedding sequences, meaning that a corresponding two-dimensional vector-quantized image is resampled more in one axis compared to the other. In their work they did not address this, except for experiments on ImageNet32 [42] where they resampled with a rate of $k = 3$, corresponding to three colour channels. However, if they were to resample again by nesting the hourglass transformers, issue of one spatial dimension being downsampled more than others would then occur.

In our formulation, we instead reshape the flattened sequence back into its a two-dimensional formulation and then apply resampling equally in the last two axes. With a resampling rate of k we apply $\sqrt{k}$ in each axis – evenly distributing the resampling among the axes. In our preliminary experiments, we found this to significantly improve the performance of the discrete prior model, and suspect a similar approach could improve performance if applied to pixels directly, which we leave for future work to confirm.

For our resampling operations, we use linear resampling (following recommendation by [35] to use linear resampling on image tasks) and a post-resampling attention layer, providing global context to the resampling operations and aggregating information globally [35]. This means our adjusted resampling method is as follows:

$$h' = A(\boldsymbol{W}^{(\intercal)} \cdot R(\boldsymbol{h}) + \boldsymbol{r}), \quad \boldsymbol{W} \in \mathbb{R}^{\frac{(d \cdot h \cdot w)}{k} \times (d \cdot h \cdot w)} \tag{4}$$

where A is the post-resampling attention layer, $\boldsymbol{h}$ is the current hidden state, $\boldsymbol{r}$ is the residual (with $\boldsymbol{r} = \mathbf{0}$ when downsampling), R is the modified reshape operation, d is the hidden layer dimension, and W is a learned projection matrix. The reshape operation R was implemented as a space-to-depth operation followed by combining the feature and depth dimensions.

Rotary Positional Embeddings – Transformers have no inductive biases that allow it inherently know the position of an element in the sequence. Embeddings that represent positions must be injected into the model in addition to the input itself. In our work, we choose to use rotary positional embeddings [48] as they require no additional parameters, incur only a small runtime cost, and can be easily extended to the multi-dimensional case [2], which we exploit here. Though transformers are clearly capable of learning that elements far apart in a flattened sequence may be close in a multi-dimensional final output, we found that explicitly extending positional embeddings to the multi-dimensional case to provide a modest boost in performance and improve the rate of training convergence. The original hourglass transformer on pixel-wise generation also opted to use rotary embeddings [35]. They note also that rotary embeddings have the advantage of being compatible with any self-attention mechanism.

Removal of Causal Constraints – In the original autoregressive formulation of hourglass transformers they noted that naively resampling could leak to information leakage into future sequence elements – therefore violating the autoregressive property [35]. As our approach is NAR we do not make any special considerations to avoid information leaking into the future. This simplifies the model by avoiding "shifting" and causal masking operations required in the original work.

3.3 Training a megapixel VQ-GAN

Training at higher resolutions means greater computational requirements and sampling speeds. With an autoregressive model, the sampling speed can be especially immense as it scales linearly with data dimensionality, even with an auxiliary vector-quantized image model [17]. With a non-autoregressive model however, the sampling speed is explicitly controlled and does not directly grow as a function of input size – excluding the increase in time for one network pass from using a larger model on a larger input.

We trained a larger variant of VQ-GAN with $v = 8192$ operating on 1024×1024 RGB images. To our knowledge, this is the highest resolution dataset VQ-GAN has been applied to [17]. Once trained, we generate a latent datasets as before, the only difference being an increased sequence length – greater than was ever tested in the original work [43]. Specifically, we obtain a downsampling rate of $f = 32$, resulting in discrete latent size of $32 \times 32 = 1024$.

The resulting reconstructions are overall of good quality given the relatively extreme compression ratio we are using. However, certain artifacts in the reconstructions still remain. Figure 6 shows examples of particularly prevalent artifacts including occasional unrealistic textures in hair (middle reconstruction) and corruption of text (right reconstruction). The corruption of text is a common issue in VQ image models [39], and the unrealistic textures are a result of the extreme compression rate or a simple lack of model capacity.

VQ-GAN is trained to minimise the true error, perceptual loss, and an adversarial loss [17] in addition to a k-means VQ loss. In our implementation, VQ-GAN is trained to minimise the following loss:

$$\begin{aligned}
L_{\text{PIX}} &= \alpha_{\text{PIX}} \cdot |\mathbf{x} - \hat{\mathbf{x}}| \cdot \\
L_{\text{VQ}} &= \alpha_{\text{VQ}} \cdot \left(||\hat{\mathbf{z}} - \mathbf{z}||^2 + ||sg[E(x)] - \mathbf{z}||_2^2 + ||E(x) - sg[\mathbf{z}]||_2^2 \right) \\
L_{\text{GAN}} &= \alpha_{\text{GAN}} \cdot (\log D(x) + \log(1 - D(\hat{x}))) \\
\lambda &= \frac{\nabla_{G_{-1}}[L_{\text{PIX}} + L_{\text{PER}}]}{\nabla_{G_{-1}}[L_{\text{GAN}}] + \epsilon} \\
L &= L_{\text{VQ}} + \lambda \cdot L_{\text{GAN}} \\
\alpha_{\text{PIX}} &= 1.0, \ \alpha_{\text{VQ}} = 1.0, \ \alpha_{\text{GAN}} = 0.5, \ \alpha_{\text{PER}} = 1.0
\end{aligned} \tag{5}$$

[17] where our discriminator is implemented using three layers, perceptual loss L_{PER} implemented using a pretrained VGG16 model [44], $\nabla_{G_{-1}}[\cdot]$ is the gradient with respect to the last layer of the VQ-GAN decoder G, and $sg[.]$ is the stop-gradient operator. The generator and discriminator model parameters are updated separately, as is standard procedure in GAN-based literature [17].

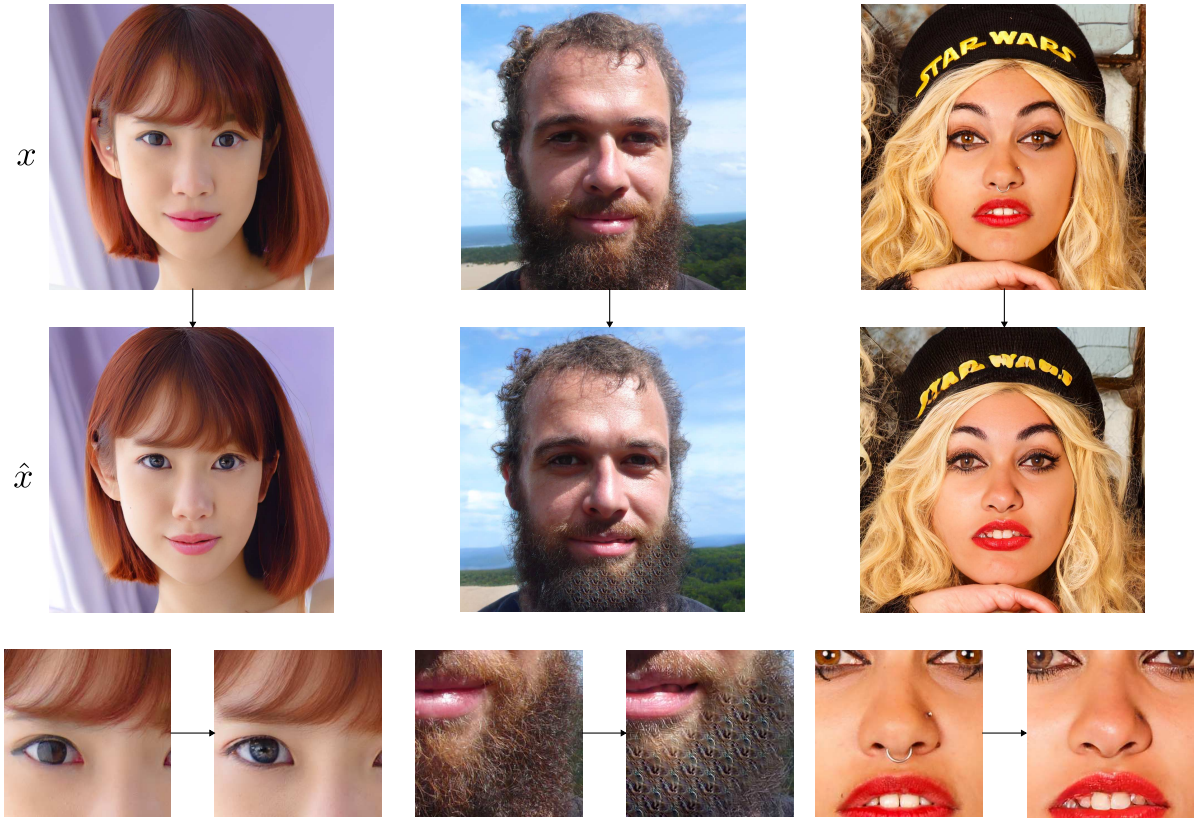


Figure 6: VQ-GAN does not produce entirely faithful reconstructions due to being optimised for perceptual quality rather than a direct error between the reconstructions and inputs. The top row shows example inputs, middle row shows resulting reconstructions, and the final row shows points of interest that have been modified – some perceptually valid and others clear artifacts. **Left**: The eye colour has been brightened, and hair shifted to conceal a piercing, rather than reconstruct it. **Middle**: The most common reconstruction artifact occurs with certain types of hair, where a repeating and unrealistic pattern occurs. The pose of the lip is also altered. **Right**: Another common artifact where text in images is corrupted. This is common across many generative models. Additionally, the model removes nose and lip piercings, in addition to altering eye makeup. This is again a valid image, but does not faithfully reconstruct the input.

This means there is less of a weight on directly minimising the pixel-wise error between the input and reconstruction. This gives rise to an interesting property of VQ-GAN where the reconstructions may be perceptually valid but clearly distinct from the input. The left reconstruction in Figure 6 demonstrates this with a change in eye colour and a shift in hair position – concealing an ear piercing. This even more apparent in the right reconstruction where all piercings are flawlessly removed – along with adjustments to eye makeup.

Using VQ image models to compress images further whilst retaining high quality and faithful reconstructions remains an open and challenging area of research. In our case at very high resolutions, this is especially true. In our preliminary experiments, we found a higher f led to the vast majority of reconstructions being of an untenable quality. Conversely, decreasing f led to latent representations of sizes that led to large memory requirements in the downstream SUNDAE prior, making inference on consumer-grade GPUs impractical.

Training VQ-GAN at this resolution and our chosen downsampling rate is extremely computationally expensive – in our configuration we were limited to a global batch size of 4 across four, 32GiB Tesla V100 GPUs. This made a full hyperparameter sweep of the other parameters of the model not possible. Therefore, we accepted good reconstructions on average with occasional artifacts that could potentially manifest in the final samples. Improving the effectiveness VQ image model itself is not the focus of this research project. Ultimately, we found these artifacts to only appear rarely in the final samples, shown in §4.1.

3.4 Non-Autoregressive Generator Training

We train a SUNDAE model on the flattened (in a raster-scan format) extracted VQ latents $\mathbf{z} = \{\mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(N)}\}$ where $N = h \cdot w$. The function $f_\theta(\cdot)$ is implemented using our proposed 2D-aware hourglass transformer.

Given a latent $\mathbf{z}$, we first apply our corruption distribution. This is done by first sampling a corruption threshold vector t with $t_i \sim U[0, 1]$ and a random matrix $\mathbf{R}$ of the same shape as $\mathbf{z}$ where $R_{i,j} \sim U[0, 1]$. Using this, we construct a mask matrix $\mathbf{M}$ with $M_{i,j} = 1$ when $R_{i,j} < t_i$ and 0 otherwise. This results in $\mathbf{M}_i$ having approximately t_i of its entries be 1.

Then, given $\mathbf{z}_0 \sim p_0$, we compute a new $\mathbf{z}_0$ to start unrolled denoising from:

$$\mathbf{z}_0 \leftarrow \mathbf{M} \cdot \mathbf{z}_0 + (\mathbf{1} - \mathbf{M}) \cdot \mathbf{z}. \tag{6}$$

where $\mathbf{z}$ is sampled from our latent dataset $\mathcal{L}$.

We then iteratively unroll the current sample $\mathbf{z}_{t-1}$ to obtain $\mathbf{z}_t$ for steps $t \in \{1, \ldots, T\}$. To perform one unroll step, simply compute logits $f_\theta(\mathbf{z}_t|\mathbf{z}_{t-1})$ and then sample from the resulting distribution to obtain $\mathbf{z}_t$, storing the logits at each step. Then, compute the cross entropy loss between all logits at each t and the target $\mathbf{z}$. This differs from some other NAR solutions which predict the corruption noise ϵ [21] rather than the target itself. The mean of the cross entropy losses is then computed to produce the final loss:

$$L^{(1:T)}(\theta) = \frac{1}{T}\left(L^{(1)}(\theta) + \cdots + L^{(T)}(\theta)\right) \tag{7}$$

as in §2.4, which then allows for backpropagation of gradients and consequently the updating of parameters θ. Though the default $T = 2$ performed well, we found $T = 3$ to result in more diverse samples.

An alternative corruption distribution would be to instead use a deterministic method $\mathbf{z}_0^{(i)} = \texttt{[MASK]}$, essentially replacing all tokens with $M_{i,j} = 1$ with a special masking token. This bears some similarity to "progressive unmasking" of latents as shown in prior work [3, 1]. This strategy was not considered due to the use of a masking token places an upper bound on the number of inference time sampling steps (updating at most one token per step) as well as not allowing for self-correction, as once a token is unmasked it is now fixed [3, 1].

3.5 Generating High-Resolution Images

During sampling, we simply sample sequentially $\mathbf{z}_t \sim f_\theta(\mathbf{z}_t|\mathbf{z}_{t-1})$ for a constant number of steps T, beginning randomly from $\mathbf{z}_0$ [43]. The original work proposed a number of improved strategies for sampling in smaller number of steps, including low-temperature sampling and updating a random subset of tokens [43], rather than all simultaneously.

Sampling, however, with a lower temperature reduces the diversity of the resulting samples. To alleviate this, we instead anneal the temperature down from a high value (≈ 1.0) down to a lower value towards the end of the sampling process. We found this retained the fast sampling speed whilst also improving diversity.

In certain latent sampling configurations, updating only a random subset of tokens also helps improve diversity. [43] used this strategy when performing low-temperature sampling. However, we found that for low-step sampling ($T < 20$) that all tokens must be able to be updated in order to produce meaningful samples before the maximum number of steps is reached. Hence in these cases, we do not follow this strategy and instead opt for a high sample proportion. In scenarios where we are permitted a time-budget allowing for a large number of sample steps, the sample proportion can be freely reduced for an increase in sample diversity.

Additionally, if a individual sample does not change between step $t-1$ and t, it is prevented from being changed further. If all samples are frozen, sampling terminates early, provided a minimum number of steps have been completed. This improves the sampling speed further with little cost to the final quality. This is significant when performing large-batch sampling or when the maximum number of steps is large.

Once sampling has terminated, the sampled latent code $\mathbf{z}_T$ is given to the VQ-GAN decoder G to produce a final sample $\mathbf{y}$. In fact, any $\mathbf{z}_i$ in the Markov chain is a valid input to the VQ-GAN decoder model. Decoding during this process and comparing the $\mathbf{y}_i$ produced at each step reveals the model gradually denoises the latent and does indeed self-correct errors accumulated during the sampling process.

3.6 Arbitrary Pattern Inpainting

As noted in the original work [43] and other non-autoregressive solutions [3], one clear advantage of non-autoregressive models is that they are not limited to causal inpainting. In general, they support arbitrary inpainting

masks which draw upon context from both the past and the future, enabling them to easily perform inpainting tasks that are more complex to implement with autoregressive models. This property also allows for higher quality and more diverse samples [3].

The inpainting procedure, takes an image $\mathbf{y} \in \mathbb{R}^{H \times W \times 3}$ and a pixel-level binary mask $m_p \in \{0,1\}^{H \times W}$ as input. By taking $f \times f$ regions of m_p and applying a logical AND in them, we obtain a latent level mask $m_{\text{vq}} \in \{0,1\}^{h \times w}$. We then sample as normal from the latents, allowing the model full context, but only update regions that were masked according to m_{vq}. Typically, we use a lower temperature for inpainting ($0.3 \leq \tau \leq 0.5$). Like with sampling, we then use G to decode the sampled latent code, producing the output $\mathbf{y}$.

Sampling at a latent-level means the model is unable to do fine-grained inpainting at a pixel level. The definition of the VQ mask m_{vq} means that some pixels outside the mask may also be altered if the pixel mask is not perfectly aligned with the VQ mask. We found in practise this had little effect on the perceptual quality of the outputs.

Implementation Details

Dataset	FFHQ-256	FFHQ-1024	CelebA	MNIST	FashionMNIST	ImageNet
Dataset Size	$60,000$	$60,000$	$190,000$	$60,000$	$60,000$	1.28M
Codebook Size	1024	8192	1024	256	256	1024
Latent Shape	16×16	32×32	16×16	28×28	28×28	16×16
Unroll Steps	3	3	3	2	2	3
Depth	$3-10-3$	$2-12-2$	$2-12-2$	$2-8-2$	$2-8-2$	$3-14-3$
Dimension	1024	1024	1024	1024	1024	1024
Shorten Factor	4	4	4	4	4	4
Attention Heads	8	8	8	8	8	12
Resample Type	Linear	Linear	Linear	Linear	Linear	Linear
Classes	–	–	–	10	10	1000
Class Dimension	–	–	–	1024	1024	1024

Table 2: Table of parameters for all training experiments. Depth is represented as three numbers corresponding to number of layers before downsampling, number of downsampled layers, and number of layers after upsampling. The dataset size is the size of the training split of the dataset. The latent shape of MNIST experiments is exactly equal to the shape of $\mathbf{x}$, as for these experiments we operate directly on a (discretized) pixel-level.

All SUNDAE models are trained using the Adam optimizer [28] as realised in its AdamW implementation in PyTorch [38]. Similarly, all models and training scripts are implemented in PyTorch. For unconditional generation on 256×256 images, training was done on a single 24GiB Nvidia RTX TITAN GPU. For unconditional generation on 1024×1024 images, training was done on a single 32GiB V100 GPU. Finally, for conditional generation on ImageNet, training was done on a single 80GiB A100 GPU. All parameters used for training the SUNDAE models are shown in Table 2.

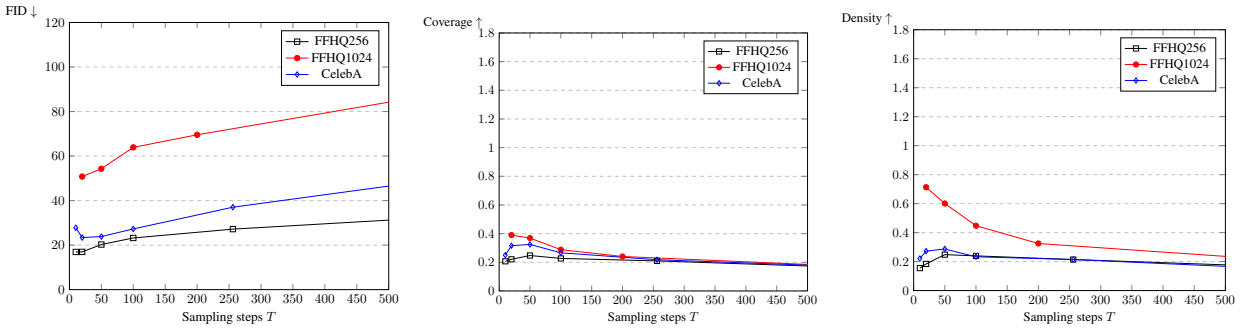
4 Evaluation

4.1 Unconditional Image Generation

We evaluate our method on the task of unconditional image generation on datasets FFHQ256, FFHQ1024, and CelebA. We use pretrained VQ-GAN checkpoints provided by the original VQ-GAN authors [17]. We evaluate our models using FID Infinity [9], Coverage, and Density [34], plotting how these metrics change as number of sampling steps T (Figure 7), sampling temperature τ (Figure 8), and sample proportion (Figure 9) are varied. Additionally, we present representative unconditional samples in Figures 1 & 10.

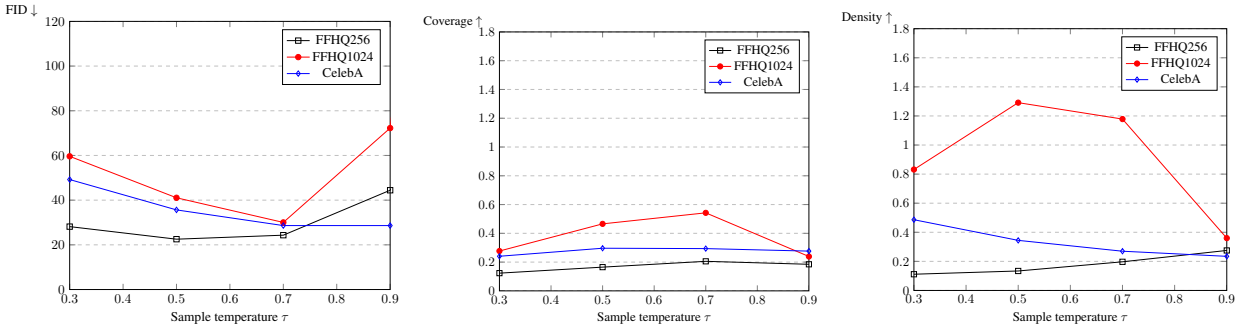
Figure 7 demonstrates a surprising property of our model: additional steps during the sampling process do not improve sample quality further. It is important to note that this only holds if the other parameters remain fixed. Therefore, the results do not suggest that additional sampling steps are always detrimental to performance, as a low number of steps will clearly result in poor quality results. Rather, it indicates that merely adding more sampling steps is not sufficient in our framework, and other parameters must also be adjusted to reflect a greater time budget.

Figure 7 shows that picking a temperature in the range of 0.5 to 0.7 leads to best sample quality across a large set of samples, at the cost of reduced diversity. Our model is capable of sampling with a higher temperature at the cost



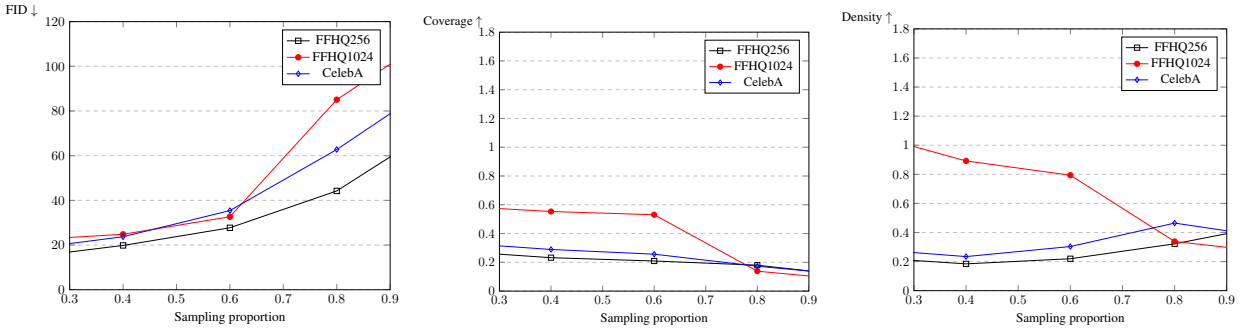
(a) Steps vs. FID (b) Steps vs. Coverage (c) Steps vs. Density

Figure 7: Plots showing sample quality in terms of different metrics as number of sampling steps T increases. Counter-intuitively, the sample quality decreases with number of sampling steps, seen on all metrics and datasets.



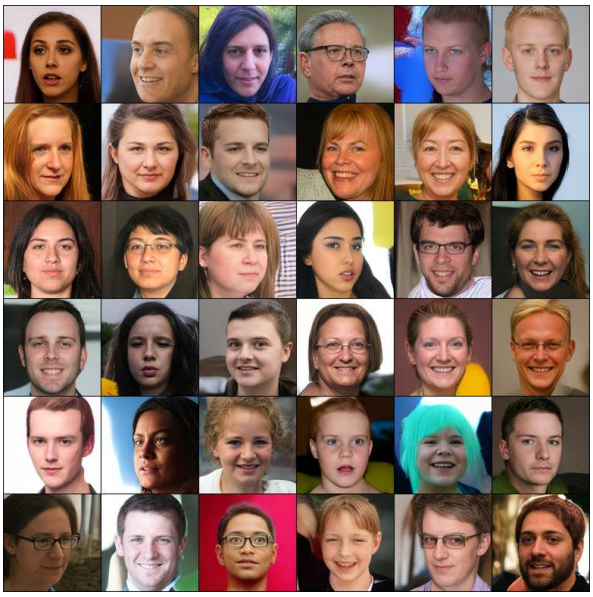
(a) Temperature τ vs. FID (b) Temperature τ vs. Coverage (c) Temperature τ vs. Density

Figure 8: Plots showing sample quality in terms of different metrics as sample temperature τ is changed. Given the other parameters, a good choice of τ falls in the range $0.5 - 0.7$. However, this range may differ depending on the other choice of parameters.



(a) Sample proportion vs. FID (b) Sample proportion vs. Coverage (c) Sample proportion vs. Density

Figure 9: Plots showing sample quality in terms of different metrics as sample proportion is changed. Lower values seem to perform better given the other parameters, but again the optimal range may differ if other parameters are changed.



(a) Non-cherry picked batch of samples from the model trained on FFHQ256.

(b) Non-cherry picked batch of samples from the model trained on CelebA.

Figure 10: Unconditional generation on 256×256 face datasets.

Figure 11: FFHQ256 samples and nearest neighbours from the dataset, based on LPIPS perceptual loss. Left-most column is a sample from our trained model, followed then by nearest neighbours, increasing in distance from left-to-right.

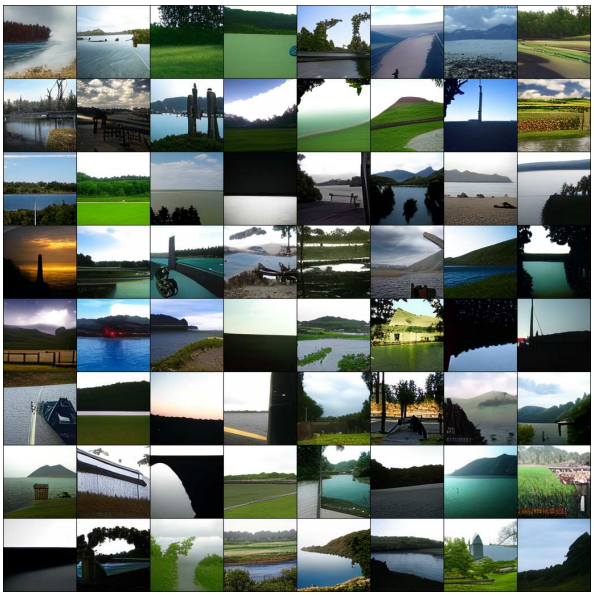
of occasionally producing corrupted examples. Figure 9 confirms the result in Savinov et al. [43] that sampling with a lower proportion leads to higher sample diversity. However, in low-step scenarios, low proportions cannot be used effectively as the majority of elements in $\mathbf{z}_t$ will not have enough opportunities to update, resulting in low quality outputs.

The result samples shown in Figures 1 & 10 demonstrate that our model is capable of generating high quality and diverse samples. Our aim was to push the efficiency of generative models to their limit, however we were still surprised at precisely how fast the model could generate – particularly on megapixel scale experiments. The samples in Figure 1

were created in a mere 2 seconds on a GTX 1080Ti. This can be improved further with more powerful hardware and further optimisation. This kind of speed is unparalleled; significantly faster than prior non-adversarial solutions at this resolution.

4.2 Conditional Image Generation

(a) 256×256 successful samples from the class "Valley".



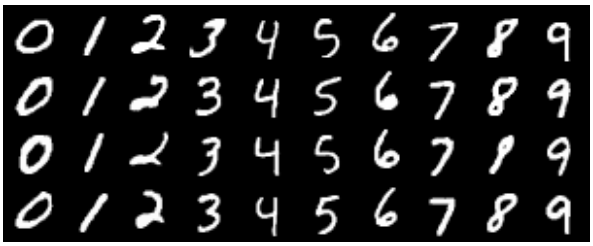
(b) 256×256 successful samples from the class "Lakeside".

(c) 256×256 failed samples from the class "King Penguin".

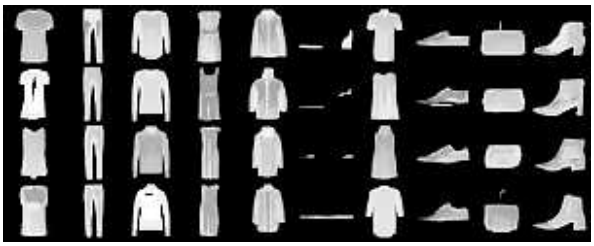
(d) 256×256 failed samples from the class "Giant Panda".

Figure 12: Examples of class-conditioned generation on ImageNet using $T = 50$ sampling steps. Top row contains examples of successful samples whereas bottom row showed failed samples. The contents of the failed samples nonetheless do resemble the target class.

Another critical component of an ideal generative model is the ability to control its generation. We explore class-conditioned image generation of ImageNet at 256×256 resolution, using the pretrained ImageNet VQ-GAN checkpoints provided by the original work [17].



(a) Conditional, pixel-wise generation on MNIST.



(b) Conditional, pixel-wise generation on Fashion-MNIST.

Figure 13: Testing conditional generation using MNIST-style datasets. Coherent samples demonstrate that the proposed conditioning method does inject class information.

There are many valid ways of injecting a conditioning signal into generative models, for example passing one-hot or embedding class vectors. We use a simple solution proposed in [36] to add a learned class embedding to every input position. To test whether their proposed method can also be applied to SUNDAE, we conducted an experiment on discretized MNIST-style datasets. To do so without a trained VQ-GAN, we treat each of the possible 8-bit greyscale colours as if it were a codebook index, resulting in $v = 256$ – generating pixels directly rather than image patches. The result of the experiments are shown in Figure 13 and demonstrate that SUNDAE can indeed incorporate a conditioning signal in this manner.

Despite this, our model fails to produce reasonable samples for all classes present in ImageNet. On classes representing large scenes such as landscapes, the samples are convincing and diverse, however for classes requiring fine-grained detail the outputs merely resemble the target class. Results of conditional generation with four classes are shown in Figure 12. Due to this, we chose not to compute perceptual metrics as the sample quality was clearly insufficient via inspection alone. This could be a result of lack of model capacity, lack of training time, or the conditioning strategy tested on MNIST being insufficient for ImageNet. The training of an effective conditional model is left for future work.

4.3 Arbitrary Image Inpainting

As outlined earlier, non-autoregressive generative models have a number of advantages on inpainting tasks, including supporting arbitrary masks and being able to use the full context available to them. We provide a number of examples of inpainting on FFHQ1024 and ImageNet, showcasing different patterns and different results given the same starting image. As our method utilises a vector quantized image model, it is incapable of doing fine-grained inpainting at a pixel level. Nonetheless, the results show consistently good inpainting results when operating at a VQ latent level.

4.4 Limitations

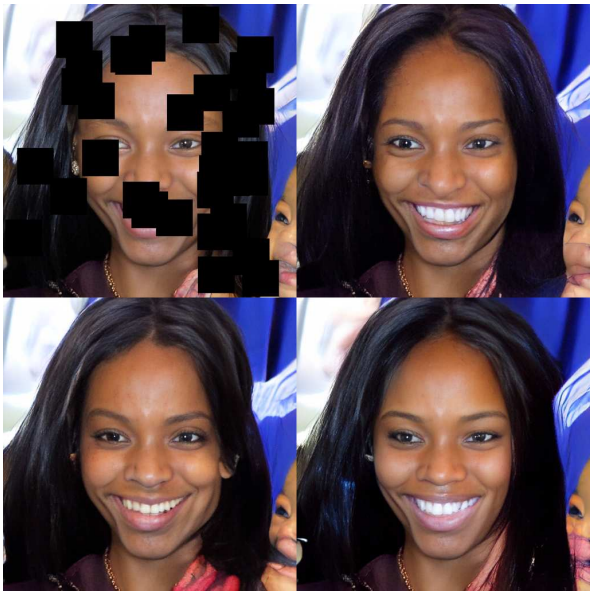
As a result of our evaluation, some limitations of our approach arise. One crucial weakness is that our VQ image model still utilises adversarial components within it. This potentially means that each image patch (corresponding to each codebook entry) could still suffer from mode collapse issues. However, our resulting samples are still diverse, suggesting that patch-wise mode collapse did not have a significant effect on the final samples.

We also encountered great instability during training of our own large VQ-GAN which led to many entirely failed experiments. Additionally, the extreme compression ratio in the large VQ-GAN model resulted in occasional unrealistic artifacts. Though most reconstructions are of good quality, the occasional artifacts did result in certain samples also being corrupted. Further research into high compression VQ models that do not use adversarial components remains and open and challenging area of research. In the case where such a VQ model is created, it can easily be substituted into our proposed framework.

Despite our model demonstrating extremely fast sampling, in terms of perceptual quality metrics, it falls short of many other methods [3]. Though measures of perceptual quality such as FID are known to be flawed [9], other measures such as density and coverage also show inferior sample quality [34]. This is especially true on ImageNet where many classes merely resemble the target class – though this is likely due to lack of model capacity and training time. Despite this, the resulting samples on the FFHQ datasets are still very diverse and of a good perceptual quality. Further work would no doubt improve quality in terms of FID further.

(a) A large example of inpainting on a 1024×1024 image using our model.

(b) Multiple outputs of inpainting using the same block mask.



(c) Multiple outputs of inpainting on the same random mask.

Figure 14: Inpainting results on FFHQ-1024. We compute multiple outputs per input image and mask to demonstrate diversity of outputs. Inpainting using a VQ image model cannot be applied perfectly at a pixel-level. Nevertheless, the model still produces many convincing outputs at very high resolutions.

5 Conclusion

In this work we investigated pushing the efficiency of generative models via the combination of various techniques, following the general trend in generative modelling of simultaneously improving quality and speed of sampling using non-adversarial approaches. We found that the combination of these techniques formed a fast image generation framework. To our surprise, the proposed method was even faster and more scaleable than expected, able to scale with ease to megapixel images, and generate samples at such resolutions in seconds – a wide margin faster than prior non-adversarial methods. Additionally, we found that previously proposed hourglass transformers are not optimally

defined on multi-dimensional inputs and subsequently proposed adjustments to them. Our work demonstrates the superiority of the non-autoregressive paradigm, and joins a rapidly growing space of research into their use as a viable alternative to autoregressive solutions. Additional research is needed into better VQ image models and into a stronger conditional generative model.

REFERENCES

[1] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2021. 5, 6, 7, 12

[2] Stella Biderman, Sid Black, Charles Foster, Leo Gao, Eric Hallahan, Horace He, Ben Wang, and Phil Wang. Rotary embeddings: A relative revolution. blog.eleuther.ai/, 2021. [Online; accessed]. 10

[3] Sam Bond-Taylor, Peter Hessey, Hiroshi Sasaki, Toby P. Breckon, and Chris G. Willcocks. Unleashing transformers: Parallel token prediction with discrete absorbing diffusion for fast high-resolution image generation from vector-quantized codes, 2021. 2, 5, 6, 7, 8, 12, 13, 17

[4] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3116668. URL http://dx.doi.org/10.1109/TPAMI.2021.3116668. 3

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018. URL https://arxiv.org/abs/1809.11096. 4

[6] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, 2017. 2, 4, 5

[7] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images, 2020. URL https://arxiv.org/abs/2011.10650. 4, 7

[8] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014. 4

[9] Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6070–6079, 2020. 13, 17

[10] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing, 2020. 7

[11] Cyprien de Masson d'Autume, Mihaela Rosca, Jack Rae, and Shakir Mohamed. Training language gans from scratch, 2019. URL https://arxiv.org/abs/1905.09922. 4

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 5, 6

[13] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. 2, 4, 7

[14] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 4

[15] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 4

[16] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion, 2021. URL https://arxiv.org/abs/2112.07068. 5, 7

[17] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. 2, 5, 7, 8, 10, 13, 16

[18] Aditya Ramesh et al. Hierarchical text-conditional image generation with clip latents, 2022. URL https://cdn.openai.com/papers/dall-e-2.pdf. 7

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2, 4

[20] Louay Hazami, Rayhane Mama, and Ragavan Thurairatnam. Efficient-vdvae: Less is more. *arXiv preprint arXiv:2203.13751*, 2022. 7

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2, 4, 12

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 4

[23] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021. 4

[24] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016. 6

[25] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models, 2021. URL https://arxiv.org/abs/2105.14080. 4, 7

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018. URL https://arxiv.org/abs/1812.04948. 4

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. 8

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 13

[29] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. 2, 4

[30] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018. 4

[31] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization, 2022. 5, 6, 7

[32] Jen-Yu Liu, Yu-Hua Chen, Yin-Cheng Yeh, and Yi-Hsuan Yang. Unconditional audio generation with generative adversarial networks and cycle regularization, 2020. 4

[33] Pamela Mishkin, Lama Ahmad, Miles Brundage, Gretchen Krueger, and Girish Sastry. Dall·e 2 preview - risks and limitations. 2022. URL https://github.com/openai/dalle-2-preview/blob/main/system-card.md. 7

[34] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. 2020. 13, 17

[35] Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models, 2021. 2, 3, 7, 9, 10

[36] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer, 2018. 2, 4, 17

[37] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2012. 4

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 13

[39] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. 2, 7, 10

[40] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. 2, 5, 7, 8

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 7

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 9

[43] Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation, 2022. 2, 3, 6, 10, 12, 15

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556. 10

[45] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2019. 2, 4, 7

[46] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2020. 2, 4, 7

[47] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models, 2021. 2, 4, 7

[48] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021. 10

[49] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder, 2020. URL https://arxiv.org/abs/2007.03898. 4, 7

[50] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space, 2021. 2, 4, 5, 7

[51] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, 2016. 4

[52] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016. 4

[53] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2017. 2, 5, 6

[54] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. 8

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 2, 4, 7

[56] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model, 2019. 6

[57] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans, 2021. 1, 2, 7

[58] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan, 2021. 5, 6, 7

[59] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec, 2021. 5, 6, 7

[60] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017. URL https://arxiv.org/abs/1703.10593. 4