Hierarchical Autoregressive Image Models with Auxiliary Decoders

Jeffrey De Fauw* [1] **Sander Dieleman*** [1] **Karen Simonyan** [1]

Abstract

Autoregressive generative models of images tend to be biased towards capturing local structure, and as a result they often produce samples which are lacking in terms of large-scale coherence. To address this, we propose two methods to learn discrete representations of images which abstract away local detail. We show that autoregressive models conditioned on these representations can produce high-fidelity reconstructions of images, and that we can train autoregressive priors on these representations that produce samples with large-scale coherence. We can recursively apply the learning procedure, yielding a hierarchy of progressively more abstract image representations. We train hierarchical class-conditional autoregressive models on the ImageNet dataset and demonstrate that they are able to generate realistic images at resolutions of 128×128 and 256×256 pixels.

Figure 1. Selected class conditional 256×256 samples from our models. More are available in the supplementary material and at `https://bit.ly/2FJkvhJ`.

1. Introduction

Generative models can be used to model the distribution of natural images. With enough capacity, they are then capable of producing new images from this distribution, which enables the creation of new natural-looking images from scratch. These models can also be conditioned on various annotations associated with the images (e.g. class labels), allowing for some control over the generated output.

In recent years, adversarial learning has proved a powerful tool to create such models (Goodfellow et al., 2014; Radford et al., 2015; Karras et al., 2018a; Brock et al., 2019; Karras et al., 2018b). An alternative approach is to specify a model in the form of the joint distribution across all pixels, and train the model on a set of images by maximising their likelihood under this distribution (or a lower bound on this likelihood). Several families of models fit into this likelihood-based paradigm, including variational autoencoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014), flow-based models (Dinh et al., 2014; 2017; Kingma & Dhariwal, 2018) and autoregressive models (Theis & Bethge, 2015; van den Oord et al., 2016b;a).

Likelihood-based models currently lag behind their adversarial counterparts in terms of the visual fidelity and the resolution of their samples. However, adversarial models are known to drop modes of the distribution, something which likelihood-based models are inherently unlikely to do. Within the likelihood-based model paradigm, autoregressive models such as PixelCNN tend to be the best at capturing textures and details in images, because they make no independence assumptions and they are able to use their capacity efficiently through spatial parameter sharing. They also achieve the best likelihoods. We describe PixelCNN in more detail in Section 2.

However, autoregressive models are markedly worse at capturing structure at larger scales, and as a result they tend to produce samples that are lacking in terms of large-scale coherence (see supplementary material for a demonstration). This can be partially attributed to the inductive bias embed-

*Equal contribution [1]DeepMind, London, UK. Correspondence to: Jeffrey De Fauw <defauw@google.com>, Sander Dieleman <sedielem@google.com>.

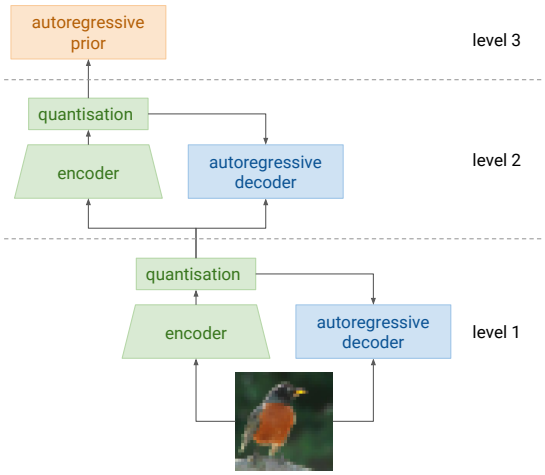


Figure 2. Schematic overview of a hierarchical autoregressive model. The dashed lines indicate different stages, which capture different scales of structure in the input image.

ded in their architecture, but it is also a consequence of the likelihood loss function, which rewards capturing local correlations much more generously than capturing long-range structure. As far as the human visual system is concerned, the latter is arguably much more important to get right, and this is where adversarial models currently have a substantial advantage.

To make autoregressive models pay more attention to large-scale structure, an effective strategy is to remove local detail from the input representation altogether. A simple way to do this for images is by reducing their bit-depth, as explored by Kingma & Dhariwal (2018) and Menick & Kalchbrenner (2019). An alternative approach is to learn new input representations that abstract away local detail, by training encoder models. We can then train autoregressive models of the image pixels conditioned on these representations, as well as autoregressive priors for these representations, effectively splitting the task into two separate stages (van den Oord et al., 2017). We can extend this approach further by stacking encoder models, yielding a hierarchy of progressively more high-level representations (Dieleman et al., 2018), as shown in Figure 2. This way, we can assign model capacity to different scales of structure in the images in a more explicit fashion, and turn the bias these models have towards capturing local structure into an advantage.

Learning representations that remove local detail while preserving enough information to enable a conditional autoregressive model to produce high-fidelity pixel-level reconstructions is a non-trivial task. A natural way to do this would be to turn the conditional model into an autoencoder, so that the representations and the reconstruction model can be learnt jointly. However, this approach is fraught with problems, as we will discuss in Section 3.

Instead, we propose two alternative strategies based on auxiliary decoders, which are particularly suitable for hierarchical models: we use feed-forward (i.e. non-autoregressive) decoders or *masked self-prediction* (MSP) to train the encoders. Both techniques are described in Section 4. We show that the produced representations allow us to construct hierarchical models trained using only likelihood losses that successfully produce samples which exhibit large-scale coherence. Bringing the capabilities of likelihood-based models up to par with those of their adversarial counterparts in terms of scale and fidelity is important, because this allows us to sidestep any issues stemming from mode dropping and exert more control over the mapping between model capacity and image structure at different scales.

We make the representations learnt by the encoders discrete by inserting vector quantisation (VQ) bottlenecks (van den Oord et al., 2017). This bounds the information content of the representations, and it enables more efficient and stable training of autoregressive priors (van den Oord et al., 2016b). Because of their discrete nature, we will also refer to the learnt representations as *codes*. We cover VQ bottlenecks in neural networks in more detail in Section 2. We also include a downsampling operation in the encoders so that higher-level codes have a lower spatial resolution.

The contributions of this work are as follows:

- We study the problems associated with end-to-end training of autoencoders with autoregressive decoders.
- We propose two alternative strategies for training such models using auxiliary decoders: feed-forward decoding and masked self-prediction (MSP).
- We construct hierarchical likelihood-based models that produce high-fidelity and high-resolution samples (128×128 and 256×256) which exhibit large-scale coherence. Selected samples are shown in Figure 1.

2. Background

We will use PixelCNN as the main building block for hierarchical image models. We will also insert vector quantisation bottlenecks in the encoders to enable them to produce discrete representations. We briefly describe both of these components below and refer to van den Oord et al. (2016b;a) and van den Oord et al. (2017) respectively for a more detailed overview.

2.1. PixelCNN

PixelCNN is an autoregressive model: it assumes an arbitrary ordering of the pixels and colour channels of an image, and then models the distribution of each intensity value in the resulting sequence conditioned on the previous values.

In practice, the intensities are typically flattened into a sequence in the raster scan order: from top to bottom, then from left to right, and then according to red, green and blue intensities. Let x_i be the intensity value at position i in this sequence. Then the density across all intensity values is factorised into a product of conditionals as follows:

$$p(\mathbf{x}) = \prod_i p(x_i|\mathbf{x}_{<i}). \quad (1)$$

PixelCNN models each of these conditionals with the same convolutional neural network composed of *masked* convolutions, which ensure that each intensity value in the sequence depends only on the values coming before it.

2.2. Vector quantisation

Vector quantisation variational autoencoders (VQ-VAE) use a vector quantisation (VQ) bottleneck to learn discrete representations. The encoder produces a continuous d-dimensional vector $\mathbf{z}$, which is then quantised to one of k possible vectors from a codebook. This codebook is learnt jointly with the other model parameters. The quantisation operation is non-differentiable, so gradients are backpropagated through it using straight-through estimation (Bengio et al., 2013). In practice, this means that they are backpropagated into the encoder as if the quantisation operation were absent, which implies that the encoder receives approximate gradients.

The model is trained using the following loss function:

$$\mathcal{L} = -\log p(\mathbf{x}|\mathbf{z}') + (\mathbf{z}' - [\mathbf{z}])^2 + \beta \cdot ([\mathbf{z}'] - \mathbf{z})^2, \quad (2)$$

where $\mathbf{x}$ is the input, $\mathbf{z}$ is the output of the encoder and $\mathbf{z}'$ is the quantised output. β is a hyperparameter and square brackets indicate that the contained expressions are treated as constant w.r.t. differentiation[1]. The three terms correspond to the reconstruction log-likelihood, the *codebook loss* and the *commitment loss* respectively. Instead of optimising all three terms using gradient descent, we use an alternative learning rule for the codebook using an exponentially smoothed version of K-means, which replaces the codebook loss and which is described in the appendix of van den Oord et al. (2017).

Although this approach was introduced in the context of autoencoders, VQ bottlenecks can be inserted in any differentiable model, and we will make use of that fact in Section 4.2.

3. Challenges of autoregressive autoencoding

Autoregressive autoencoders are autoencoders with autoregressive decoders. Their appeal lies in the combination of two modelling strategies: using latent variables to capture global structure, and autoregressive modelling to fill in local detail. This idea has been explored by van den Oord et al. (2016a), Gulrajani et al. (2016), Chen et al. (2016), van den Oord et al. (2017), Engel et al. (2017) and Dieleman et al. (2018) among others.

During training, autoregressive models learn to predict one step ahead given the ground truth. This is often referred to as teacher forcing (Williams & Zipser, 1989). However, when we sample from a trained model, we use previous predictions as the model input instead of ground truth. This leads to a discrepancy between the training and inference procedures: in the latter case, prediction errors can accumulate. Unfortunately, autoregressive autoencoders can exhibit several different pathologies, especially when the latent representation has the same spatial structure as the input (i.e., there is a spatial map of latents, not a single latent vector). Most of these stem from an incompatibility between teacher forcing and the autoencoder paradigm:

- When the loss actively discourages the use of the latent representation to encode information (like the KL term does in VAEs), the decoder will learn to ignore it and use only the autoregressive connections. This phenomenon is known as posterior collapse (Bowman et al., 2016).
- On the other hand, when the latent representation has high information capacity, there is no incentive for the model to learn to use the autoregressive connections in the decoder.
- The encoder is encouraged to preserve in the latent representations any noise that is present in the input, as the autoregressive decoder cannot accurately predict it from the preceding pixels. This is counter to the intuition that the latent representations should capture high-level information, rather than local noise. This effect is exacerbated by teacher forcing which, during training, enables the decoder to make very good next-step predictions in the absence of noise. When noise is present, there will be a very large incentive for the model to store this information in the codes.
- The encoder is encouraged to ignore slowly varying aspects of the input that are very predictable from local information, because the ground truth input is always available to the autoregressive decoder during training. This affects colour information in images, for example: it is poorly preserved when sampling image reconstructions, as during inference the sampled intensities will quickly deviate slightly from their original values, which then recursively affects the colour of subsequent pixels. This is demonstrated in Figure 3.

Workarounds to these issues include strongly limiting the

[1] $[x]$ is like `tf.stop_gradient(x)` in TensorFlow.

Figure 3. Illustration of an issue with autoregressive autoencoders caused by teacher forcing. Left: original 128×128 image. Right: reconstruction sampled from an autoregressive autoencoder. Colour information is lost because it varies slowly across the image and is not captured by the encoder.

capacity of the representation (e.g. by reducing its spatial resolution, using a latent vector without spatial structure, inserting a VQ bottleneck and/or introducing architectural constraints in the encoder) or limiting the receptive field of the decoder (Gulrajani et al., 2016; van den Oord et al., 2017; Dieleman et al., 2018). Unfortunately, this limits the flexibility of the models. Instead, we will try to address these issues more directly by decoupling representation learning from autoregressive decoder training.

4. Auxiliary decoders

To address the issues associated with jointly training encoders and autoregressive decoders which were discussed in the previous section, we introduce auxiliary decoders: separate decoder models which are only used to provide a learning signal to the encoders. Once an encoder has been trained this way, we can discard the auxiliary decoder and replace it with a separately trained autoregressive decoder conditioned on the encoder representations. The autoregressive decoder consists of a local model (PixelCNN) and a modulator which maps the encoder representations to a set of biases for each layer of the local model. This separation allows us to design alternative decoders with architectures and loss functions that are tuned for feature learning rather than for reconstruction quality alone.

Although the encoder and decoder are trained using different loss functions, it is still convenient to train them simultaneously, taking care not to backpropagate the autoregressive decoder loss to the encoder. Otherwise, multiple networks would have to be trained in sequence for each level in the hierarchy. We use simultaneous training in all of our experiments. This has a negligible effect on the quality of the reconstructions.

4.1. Feed-forward decoders

The most straightforward form the auxiliary decoder can take is that of a feed-forward model that tries to reconstruct the input. Even though the task of both decoders is then the same, the feed-forward architecture shapes what kinds of information the auxiliary decoder is able to capture. Because such a decoder does not require teacher forcing during training, the issues discussed in Section 3 no longer occur.

When trained on RGB images, we can treat the pixel intensities as continuous and use the mean squared error (MSE) loss for training. For other types of inputs, such as codes produced by another encoder, we can use the same multinomial log-likelihood that is typically used for autoregressive models. Using the MSE would not make sense as the discrete codes are not ordinal and cannot be treated as continuous values. Figure 4 (left) shows a diagram of an autoregressive autoencoder with an auxiliary feed-forward decoder.

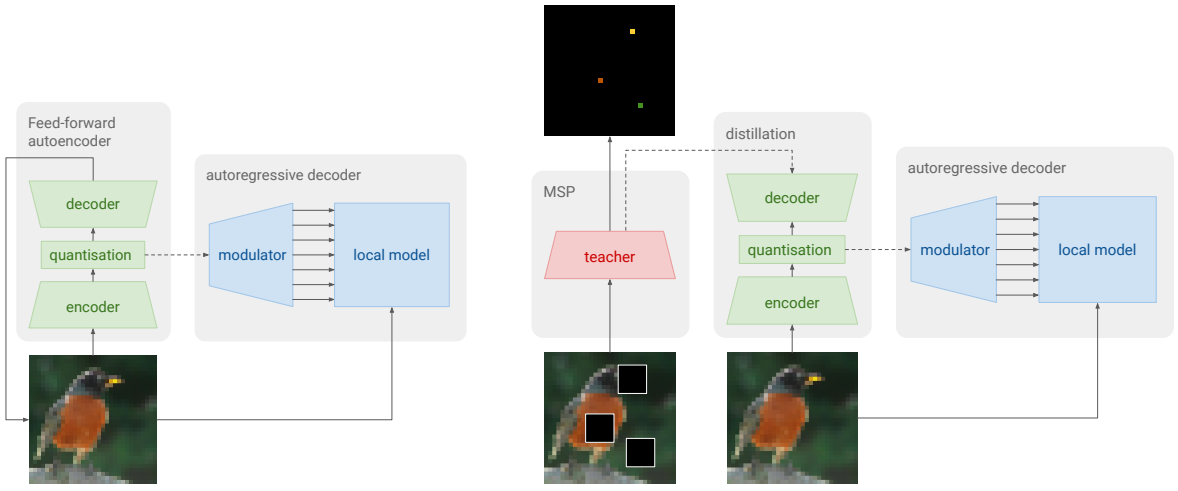
A significant benefit of this approach is its simplicity: we do not stray too far from the original autoencoder paradigm because the model is still trained using a reconstruction loss. However, an important drawback is that the reconstruction task still encourages the model to capture as much information as possible in the codes, even unimportant details that would be easy for the autoregressive decoder to fill in. It affords relatively little control over the nature and the quantity of information captured in the codes.

4.2. Masked self-prediction decoders

Models with feed-forward decoders are encouraged to encode as much information as possible in the codes to help the decoders produce detailed reconstructions. As a result, the codes may not be very compressible (see Section 7.1), which makes stacking multiple encoders to create a hierarchy quite challenging.

Instead of optimising the auxiliary decoder for reconstruction, we can train it to do self-prediction: predict the distribution of a pixel given the surrounding pixels. By masking out some region of the input around the pixel to be predicted, we can prevent the decoder from using strong local correlations, and force it to rely on weaker long-range dependencies instead. As a result, the produced codes will be much more compressible because they only contain information about longer-range correlations in the input. The local detail omitted from these representations can later be filled in by the autoregressive decoder.

In practice, *masked self-prediction* (MSP) entails masking some square region of the input and predicting the middle pixel of this region. If the region is 7×7 pixels large, for example, the model can only rely on correlations between pixels that are at least 4 positions away for its prediction. In the presence of noise or other unpredictable local detail (such as textures), the MSP model will be uncertain about its predictions and produce a distribution across all possibilities. This implies that the encoder representation will capture this uncertainty and incorporate high-level context, rather than trying to encode the exact pixel values. Given an input $\mathbf{x}$,



(a) (b)

Figure 4. Discrete autoencoders with auxiliary decoders. Left: a feed-forward decoder is used to train the encoder. Right: a masked self-prediction (MSP) model is trained and then distilled into a new model with unmasked input to obtain the encoder. Both models feature autoregressive decoders. Note the dashed arrows indicating that no gradients are backpropagated along these connections.

a pixel position (i, j) and an offset s (corresponding to a mask size of $2s + 1$), the MSP objective is to maximise $\log p(x_{ij}|\mathbf{x} \cdot \mathbf{m})$, where the input mask $\mathbf{m}$ is given by:

$$m_{kl} = \begin{cases} 0 \text{ if } i - s \leq k \leq i + s, j - s \leq l \leq j + s \\ 1 \text{ otherwise} \end{cases}. \quad (3)$$

In practice, we can select multiple pixel positions per input to make training more sample-efficient, but the total number of positions should be limited because too much of the input could be masked out otherwise.

Because this approach requires input masking, we would have to run a forward pass through the encoder with a different mask for each spatial position if we wanted to compute representations for the entire input. This is computationally prohibitive, so instead, we use distillation (Hinton et al., 2015) to obtain an encoder model that does not require its input to be masked. We train a *teacher* model with masked input (a simple feed-forward network), and simultaneously distill its predictions for the selected pixel positions into a *student* model with unmasked input and a vector quantisation bottleneck. Because it is convolutional, this student model will learn to produce valid representations for all spatial positions, even though it is only trained on a subset of spatial positions for each input. Representations for an input can then be computed in a single forward pass. The full setup is visualised in Figure 4 (right).

5. Related work

Recent work on scaling generative models of images to larger resolutions has been focused chiefly on adversarial models. Karras et al. (2018a;b) trained generative adversarial networks (GANs) that can generate various scenes and human faces at resolutions of 256×256 and higher (up to 1 megapixel for the latter). Brock et al. (2019) generate 512×512 images for each of the 1000 classes in the ImageNet dataset, all with the same GAN model.

Reed et al. (2017) train a multiscale autoregressive model which gradually upsamples images, starting from 4×4 pixels, and makes some independence assumptions in the process. Conditioned on textual captions and spatial keypoints, the model is capable of producing realistic 256×256 images of birds. It can also be used to upsample low-resolution images. Although using low-resolution images as representations that abstract away local detail is appealing, this necessarily removes any high-frequency information. Learning these representations instead is more flexible and allows for capturing high-frequency structure.

Menick & Kalchbrenner (2019) train a variant of PixelCNN dubbed 'subscale pixel network' (SPN), which uses a different, hierarchical ordering of the pixels rather than the raster scan order to factorise the joint distribution into conditionals. Their best models consist of two separate SPNs, where one models only the 3 most significant bits of the intensity values at a lower resolution, and another conditionally fills in the remaining information. Trained on the ImageNet dataset, this model is able to generate visually compelling unconditional samples at 128×128 resolution.

Dieleman et al. (2018) train a hierarchical autoregressive model of musical audio signals by stacking autoregressive discrete autoencoders. However, the autoencoders are

trained end-to-end, which makes them prone to the issues we identified in Section 3. This makes training the second level autoencoder cumbersome and fragile, requiring expensive population-based training (Jaderberg et al., 2017) or alternative quantisation strategies to succeed.

Masked self-prediction is closely related to representation learning methods such as context prediction (Doersch et al., 2015) and context encoders (Pathak et al., 2016), which also rely on predicting pixels from other nearby pixels. Contrastive predictive coding (Oord et al., 2018) on the other hand relies on prediction in the feature domain to extract structure that varies predictably across longer ranges. Although the motivation behind approaches such as these is usually to extract high-level, semantically meaningful features, our goal is different: we want to remove some of the local detail to make the task of modelling large-scale structure easier. Our representations also need to balance abstraction with reconstruction, so they need to retain enough information from the input.

Context prediction is also a popular representation learning approach in natural language processing, with well-known examples such as word2vec (Mikolov et al., 2013), ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). Other related work includes PixelNet (Bansal et al., 2017), which uses loss functions defined on subsets of image pixels (much like MSP) to tackle dense prediction tasks such as edge detection and semantic segmentation.

6. Evaluation

Likelihood-based models can typically be evaluated simply by measuring the likelihood in the pixel domain on a held-out set of images. With the hierarchical approach, however, different parts of the model are trained using likelihoods measured in different feature spaces, so they are not directly comparable. For a given image, we can measure the likelihood in the feature space modelled by the prior, as well as conditional likelihoods in the domains modelled by each autoregressive decoder, and use these to calculate a joint likelihood across all levels of the model. Let $\mathbf{x}$ be an image, and $\mathbf{z}_n$ the code representation of the image for each level n in a hierarchy of N levels. Then we can compute:

$$p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_N) \cdot p(\mathbf{z}_{N-1}|\mathbf{z}_N) \cdot \ldots \cdot p(\mathbf{x}|\mathbf{z}_1) \quad (4)$$

Because all encoders are deterministic, they associate only one set of codes with each image. If the corresponding decoders had infinite capacity, this would imply that the marginal distribution $p(\mathbf{x})$ is equal to the joint distribution $p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_N)$. In practice however, the decoders are imperfect and they will assign some non-zero probability to an input when conditioned on other codes: let $\tilde{\mathbf{z}}_1 \neq \mathbf{z}_1$, then $p(\mathbf{x}|\tilde{\mathbf{z}}_1) > 0$. This implies that to calculate $p(\mathbf{x})$ correctly, we would have to integrate $p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_N)$ over all codes. This is intractable, but instead we can use $p(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_N)$ as a lower bound for $p(\mathbf{x})$. Although we will report this bound for some models, we stress that likelihoods measured in the pixel domain are not suitable for measuring whether a model captures large-scale structure (Theis et al., 2016) – indeed, this is the primary motivation behind our approach.

Instead, we turn to evaluation metrics that are commonly used for implicit generative models (for which computing likelihoods is intractable): the Inception Score (IS) (Salimans et al., 2016) and the Fréchet Inception Distance (FID) (Heusel et al., 2017). Both rely on a pre-trained ImageNet classification model and require a large number of model samples, so they are somewhat expensive to compute. Nevertheless, sampling from our models is fast enough for this to be tractable: when amortizing over the batch dimension on a single NVIDIA V100 GPU, generating class conditional 256×256 images from the models used in Figure 1 takes less than 5 minutes per image, whereas for 128×128 samples it takes less than 1 minute (more details can be found in the supplementary material). Note that these metrics are not perfect, and some of their flaws have recently been described by Barratt & Sharma (2018); Bikowski et al. (2018). To account for this, we will show samples from various models to allow for a subjective visual comparison in addition to reporting these metrics.

When generating samples, we can change the temperature of the multinomial distribution which we sequentially sample from for each channel and spatial position. We find that slightly reducing the temperature from the default of 1.0 to 0.97 or 0.98 more consistently yields high quality samples, so we regularly make use of this procedure.

7. Experiments and results

All experiments were performed on the ImageNet dataset (Deng et al., 2009) with full bit-depth RGB images (8 bits per channel). For experiments on resolutions of 128×128 and 256×256, we rescale the shortest side of the image to the desired size and then randomly crop a square image during training (to preserve the aspect ratio). We also use random flipping and brightness, saturation and contrast changes to augment the training data (Szegedy et al., 2015). For 64×64 experiments, we use the images made available by van den Oord et al. (2016b), without augmentation.

7.1. Auxiliary decoder design

As the auxiliary decoders are responsible for shaping the representations learnt by the encoders, their architecture can be varied to influence their information content. We use residual networks for both the encoders and auxiliary decoders (He et al., 2016a) and vary the number of layers. Each additional layer extends the receptive field of the

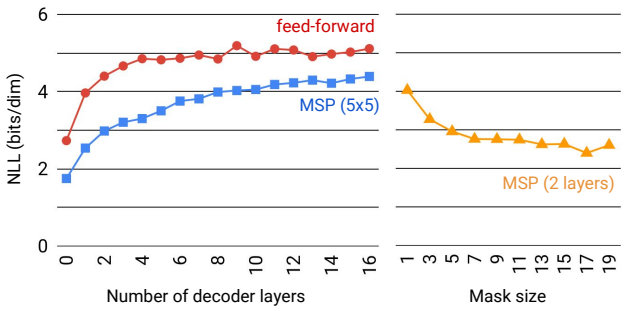


Figure 5. Predictability of codes produced by different encoders, as measured by the validation set NLL achieved by a small PixelCNN prior (see text for details). Left: increasing the number of auxiliary decoder layers makes the codes harder to predict. NLLs for codes from feed-forward decoders (red circles) flatten out more quickly than those from MSP decoders (mask size 5×5, blue squares). Right: increasing the mask size for MSP decoders makes the resulting codes easier to predict (orange triangles).

decoder, which implies that a larger neighbourhood in the code space can affect any given spatial position in the output space. As a result, the information about each pixel is spread out across a larger neighbourhood in the code space, which allows for more efficient use of the discrete bottleneck.

To measure the effect this has on the compressibility of the codes, we first train some autoencoder models on 64×64 colour images using a discrete bottleneck with a single 8-bit channel (256 codes) and downsampling to 32×32 using a strided convolutional layer. The codes are upsampled in the decoder using a subpixel convolutional layer (Shi et al., 2016). We then train prior models and measure the validation negative log-likelihood (NLL) they achieve at the end of training. The priors are modestly sized PixelCNN models with 20 layers and 128 units per layer.

The results for both feed-forward and MSP decoders (mask size 5×5) are shown in Figure 5 (left). It is clear that the codes become less predictable as the receptive field of the auxiliary decoder increases. As expected, the MSP codes are also more predictable than the feed-forward codes. The predictability of the feed-forward codes seems to flatten out after about 8 decoder layers, while that of the MSP codes decreases more gradually.

For MSP decoders, we repeat this experiment fixing the number of layers to 2 and varying the mask size instead. The results are shown in Figure 5 (right). As expected, increasing the mask size reduces the information content of the codes and makes them more predictable. In the supplementary material, we also discuss the effect of the auxiliary decoder design on reconstruction quality.

Figure 6. Autoregressive autoencoder reconstructions of 128×128 images. Left: original images. Middle: two different sampled reconstructions from models with a feed-forward auxiliary decoder trained with the MSE loss. Right: two different sampled recontructions from models with an MSP auxiliary decoder with mask size 3×3. The sampling temperature was 0.99. More examples can be found in the supplementary material.

7.2. Codes abstract away local detail

In this subsection and the next, we evaluate the building blocks that we will use to construct hierarchical models of 128×128 and 256×256 RGB images. To verify that the codes learnt using auxiliary decoders abstract away local detail, we show the variability of sampled reconstructions. Note that the codes for a given input are deterministic, so all stochasticity comes from the autoregressive decoder. We compress all images into single-channel 8-bit codes (256 bins) at 32×32 resolution. We could obtain higher-fidelity reconstructions by increasing the code capacity (by adding more channels, increasing their bit-depth or increasing the resolution), but we choose to use single-channel codes with a modest capacity, so that we can train powerful priors that do not require code channel masking to ensure causality.

For 128×128 images (48× compression[2]), we use a feed-forward auxiliary decoder with 8 layers trained with the MSE loss, and an MSP decoder with 8 layers and a 3×3 mask size. Reconstructions for both are shown in Figure 6. Note how the MSE decoder is better at preserving local structure (e.g. text on the box), while the MSP decoder is better at preserving the presence of textures (e.g. wallpaper behind the pots, body of the bird). Both types of reconstructions show variation in colour and texture.

For 256×256 images (192× compression), we use a stack of two autoencoders, because using a single autoencoder would result in a significant degradation in visual fidelity (see supplementary material). In Figure 7, we show reconstructions from a stack trained with feed-forward auxiliary decoders, where the first level model compresses to single-channel 8-bit codes at 128×128 resolution, so the first and second level models compress by factors of 12× and 16×

[2]From 128 × 128 × 3 × 8 bits to 32 × 32 × 1 × 8 bits

Figure 7. Autoregressive autoencoder reconstructions of 256×256 images. Left: original images. Middle: three different sampled two-level reconstructions from models with a feed-forward auxiliary decoder trained with the MSE loss. Right: three different sampled two-level recontructions from models with an MSP auxiliary decoder with mask sizes 5×5 (level 1) and 3×3 (level 2). The sampling temperature was 0.99. More examples can be found in the supplementary material.

Figure 8. Selected class conditional 128×128 samples from our models with feed-forward auxiliary decoders (top) and MSP decoders (bottom). More are available in the supplementary material and at `https://bit.ly/2FJkvhJ`.

respectively (using 1 and 12 auxiliary decoder layers respectively). We also show reconstructions from a stack trained with MSP auxiliary decoders, where the first level compresses to 8-bit codes at 64×64 resolution with 3 channels, so the first and second level models compress by factors of 16× (with 4 auxiliary decoder layers) and 12× (with 8 layers) respectively.

7.3. Hierarchical models

We construct hierarchical models using the autoencoders from the previous section, by training class-conditional autoregressive priors on the codes they produce. Once a prior is trained, we can use ancestral sampling to generate images. Most of the model capacity in our hierarchical models should be used for visually salient large-scale structure, so we use powerful prior models, while the decoder models are relatively small.

For 128×128 images, we report results and show samples for two priors: one trained on the feed-forward codes from the previous section, and one for the MSP codes. The prior models are large gated PixelCNNs augmented with masked self-attention layers (Vaswani et al., 2017), which are inserted after every few convolutional layers in the vein of PixelSNAIL (Chen et al., 2018). Selected samples are shown in Figure 8. Samples for all classes are available at `https://bit.ly/2FJkvhJ`. IS and FID are reported in Table 1, as well as the joint NLL over the pixels and codes, which represents a bound on the marginal NLL over the pixels only. We do not directly compare with results from previous papers as we cannot compute exact likelihoods, and differences in preprocessing of the images can significantly affect these measurements. The IS and FID are much worse than those reported for recent adversarial models (Brock et al., 2019), but they are in the same ballpark as those reported for PixelCNN on 32×32 ImageNet by Ostrovski et al. (2018) (IS 8.33, FID 33.27).

Table 1. IS and FID for autoregressive priors trained on 32x32 codes obtained from 128x128 images. We also report the joint NLL as discussed in Section 6.

AUX. DECODER	IS	FID	JOINT NLL
Feed-forward	18.10 ± 0.96	44.95	3.343 bits/dim
MSP	17.02 ± 0.79	46.05	3.409 bits/dim

We also conducted a human evaluation study to complement the reported metrics. We asked human raters to compare pairs of images and select the one which looks the most realistic. Each pair consists of one sample from the model trained with feed-forward auxiliary decoders, and one sample from the model trained with MSP auxiliary decoders. We selected a varied set of 20 classes in an attempt to cover the different types of object classes available in the dataset, and used 50 samples for each class (the same ones that were made available online). We then created 500 random sample pairs per class, in such a way that every sample is used exactly 10 times (using all possible pairs would require too many ratings). Each pairing was rated by 3 raters, for a total of 30,000 ratings.

Accounting for 53 missing answers (0.18%), all raters agreed in 63.68% of cases. The samples from the MSP model are preferred in 50.89% of cases, and the samples from the feed-forward model are preferred 48.93% of cases.

Although this result is very balanced, there are larger differences between the models within each class. More details about our human evaluation setup are provided in the appendix, as well as a few nearest neighbour comparisons of samples with images from the dataset in different feature spaces.

For 256×256 images, we also train two priors, one on feed-forward codes and one on MSP codes. Samples from both are available in the supplementary material and at https://bit.ly/2FJkvhJ.

8. Conclusion

We have discussed the challenges of training autoregressive autoencoders, and proposed two techniques that address these challenges using auxiliary decoders. The first uses a feed-forward network to reconstruct pixels, while the second relies on predicting missing pixels. This enabled us to build hierarchical autoregressive models of images which are capable of producing high-fidelity class-conditional samples with large-scale coherence at resolutions of 128×128 and 256×256 when trained on ImageNet. This demonstrates that our hierarchical approach can be used to effectively scale up likelihood-based generative models. In future work, we would like to compare both techniques in more challenging settings and further explore their relative strengths and limitations.

Acknowledgements

We would like to thank the following people for their help and input: Aäron van den Oord, Ali Razavi, Jacob Menick, Marco Cornero, Tamas Berghammer, Andy Brock, Jeff Donahue, Carl Doersch, Jacob Walker, Chloe Hillier, Louise Deason, Scott Reed, Nando de Freitas, Mary Chesus, Jonathan Godwin, Trevor Back and Anish Athalye.

References

Bansal, A., Chen, X., Russell, B., Gupta, A., and Ramanan, D. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv:1702.06506*, 2017.

Barratt, S. and Sharma, R. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bikowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pp. 10–21, 2016.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *CoRR*, abs/1611.02731, 2016.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. Pixelsnail: An improved autoregressive generative model. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pp. 863–871. JMLR.org, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

Dieleman, S., van den Oord, A., and Simonyan, K. The challenge of realistic music generation: modelling raw audio at scale. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8000–8010. Curran Associates, Inc., 2018.

Dinh, L., Krueger, D., and Bengio, Y. NICE: non-linear independent components estimation. *CoRR*, abs/1410.8516, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. 2017.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1068–1077, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.

Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Gulrajani, I., Kumar, K., Ahmed, F., Ali Taiga, A., Visin, F., Vazquez, D., and Courville, A. PixelVAE: A latent variable model for natural images. *arXiv e-prints*, abs/1611.05013, November 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *ECCV*, 2016b.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018a.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018b.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10236–10245. Curran Associates, Inc., 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Menick, J. and Kalchbrenner, N. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *International Conference on Learning Representations*, 2019.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ostrovski, G., Dabney, W., and Munos, R. Autoregressive quantile networks for generative modeling. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pp. 3933–3942. JMLR.org, 2018.

Paine, T. L., Khorrami, P., Chang, S., Zhang, Y., Ramachandran, P., Hasegawa-Johnson, M. A., and Huang, T. S. Fast wavenet generation algorithm. *CoRR*, abs/1611.09482, 2016. URL http://arxiv.org/abs/1611.09482.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1202.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30 (4):838–855, July 1992. ISSN 0363-0129. doi: 10.1137/0330046. URL http://dx.doi.org/10.1137/0330046.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Razavi, A., Oord, A. v. d., Poole, B., and Vinyals, O. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.

Reed, S. E., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., Belov, D., and de Freitas, N. Parallel multiscale autoregressive density estimation. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2912–2921. PMLR, 2017.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. In *CVPR*, pp. 1874–1883. IEEE Computer Society, 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Theis, L. and Bethge, M. Generative image modeling using spatial lstms. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1927–1935. Curran Associates, Inc., 2015.

Theis, L., van den Oord, A., and Bethge, M. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016.

van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, k., Vinyals, O., and Graves, A. Conditional image generation with pixelcnn decoders. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4790–4798. Curran Associates, Inc., 2016a.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1747–1756, New York, New York, USA, 20–22 Jun 2016b. PMLR.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6306–6315. Curran Associates, Inc., 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

A. Architectural details

In this section, we describe the architecture of the different components of our models, and provide hyperparameters for all experiments. All hierarchical models consist of one or two autoencoder models and a prior, which are trained separately and in sequence. We use Polyak averaging (Polyak & Juditsky, 1992) for all models with a decay constant of 0.9999.

Each autoencoder consists of a number of subnetworks: an encoder, an autoregressive decoder and an auxiliary decoder. A quantisation bottleneck is inserted between the encoder and both decoders. In the case of MSP training, there is also an additional teacher network. The autoregressive decoder in turn consists of a modulator and a local model. The local model is always a gated PixelCNN (van den Oord et al., 2016a). The modulator is a residual net and is responsible for mapping the code input to a set of biases for each layer in the local model. The encoder, auxiliary decoder, and teacher networks are all residual nets as well. For all residual networks, we use the ResNet v2 'full pre-activation' formulation (He et al., 2016b) (without batch normalisation), where each residual block consists of a ReLU nonlinearity, a 3×3 convolution, another ReLU nonlinearity and a 1×1 convolution, in that order. Note that we chose not to condition any of the autoencoder components on class labels in our experiments (only the priors are class-conditional).

We train all models on 64×64 crops, unless the input representations already have a resolution of 64×64 or smaller. When training MSP models, we use a different number of masks per image, depending on the mask size. For mask sizes 1×1 and 3×3 we use 30 masks. For mask sizes 5×5 and 7×7 we use 10 masks. For sizes 9, 11, 13 and 15 we use 3 masks, and for 17 and 19 we us a single mask per image. In preliminary experiments, these settings enabled us to get the best self-prediction likelihoods. Note that we present the images and codes to the MSP teacher and encoder in a one-hot representation, so that masked pixels ($[0, 0, \ldots, 0]$) can be distinguished from black pixels ($[1, 0, \ldots, 0]$).

We first describe the architecture details for the autoregressive autoencoders with feed-forward and masked self-prediction auxiliary decoders for the different resolutions. The priors on the resulting codes are described jointly for all models in Section A.3.

A.1. 2-level models for 128×128 images

A.1.1. With feed-forward decoder

The encoder, auxiliary decoder, and modulator are all residual networks with 512 hiddens and a residual bottleneck of 128 hiddens. The encoder and modulator both have 16 layers whereas the auxiliary decoder has only 2 layers. In the encoder the features are downscaled at the end using a strided convolution with a stride of 2. In the auxiliary decoder and the modulator the upsampling (by a factor of 2) is done in the beginning using subpixel convolutional layer (Shi et al., 2016). The local model, a gated PixelCNN, has 16 layers with 128 units. The VQ bottleneck has 1 channel with 9 bits[3] (512 bins). The model was trained with the Adam optimizer for 300000 iterations.

A.1.2. With MSP decoder

The teacher, encoder, decoder and modulator are all residual networks with 128 hiddens. The encoder, teacher and modulator have 16 layers whereas the auxiliary decoder has 8 layers. We use a mask size of 3×3. The local model has 20 layers and the VQ bottleneck has 1 channel with 8 bits. The model was trained with the Adam optimizer for 200000 iterations. Other than that, the setup matches the one used for the feed-forward decoder.

A.2. 3-level models for 256×256 images

A.2.1. With feed-forward decoder

For the first level model, which maps 256×256 RGB images to 128×128 single channel codes with 8 bits, we can make the components relatively small. We use the same model as described in Section A.1.1 except for the following: we use only 4 encoder layers, 1 layer in the auxiliary decoder and 8 autoregressive layers.

The second level model, which maps the 128×128 codes to single channel 32×32 codes with 8 bits, uses 16 encoder and 16 modulator layers with 1024 hiddens and residual bottlenecks of 256 hiddens. The auxiliary decoder has 12 layers and the autoregressive decoder is also only 8 layers but now has 384 hiddens per layer.

A.2.2. With MSP decoder

For the first level model, which maps 256×256 RGB images to 64×64 3-channel codes with 8 bits, we use the same model as described in Section A.1.2, except that the auxiliary decoder has 4 layers and the mask size is 5×5.

The second level model, which maps the 64×64 codes to single channel 32×32 codes with 8 bits is also the same, but has an auxiliary decoder with 8 layers and the mask size is 3×3.

A.3. Prior details

Our autoregressive priors are very similar to those described in (Razavi et al., 2019), which are in turn closely related to PixelSNAIL (Chen et al., 2018). We list their details in

[3] We use 9 bits instead of the 8 bits used in other experiments because there was a significant increase in reconstruction quality when using 9 bits instead of 8.

Table 2. Architecture details for the priors on the different codes: FF denotes the feed-forward decoder model and MSP denotes the masked self-prediction model. l is the number of layers, h the number of hiddens for each layer, r is residual filter size, $timing$ denotes if the timing signal was added or concatenated with the input, a is number of attention layers, ah is the number of attention heads, do is the probability of dropout, bs is the batch size and $iters$ is the number of iterations the model has been trained for.

	128×128 FF	128×128 MSP	256×256 FF	256×256 MSP
l	20	20	20	20
h	640	640	640	640
r	2048	2048	2048	2048
$timing$	add	add	concat	add
a	6	5	5	5
ah	10	10	15	10
do	0.2	0.1	0.0	0.1
bs	2048	2048	2048	2048
$iters$	490200	267000	422700	185000

Table 2.

B. Duration of sampling

We report all sampling timings using a single NVIDIA V100 GPU. We use a version of incremental sampling (Paine et al., 2016) which uses buffers to avoid unnecessary recomputation. Because our current version of incremental sampling does not support models with attention, we use naive sampling for sampling for the autoregressive priors: at every point we simply pass in the entire previously sampled input to the model. For 128×128 it takes roughly 23 minutes to sample a batch of 25 codes from the prior and 9 minutes to use the level 1 model to map these codes to 128×128 images. For a batch of 9 images at 256×256 resolution, it takes 10 minutes to first sample the level 2 codes from the prior, less than 2 minutes to sample level 1 codes conditioned on these level 2 codes, and finally 9 minutes to use these codes to sample the 256×256 images themselves.

C. PixelCNN bias towards local structure

To demonstrate that autoregressive models like PixelCNN are inherently biased towards capturing local structure, we trained a class-conditional gated PixelCNN model (van den Oord et al., 2016a) with 20 layers, 384 hidden units and 1024 hidden units for the residual connections on 64×64 ImageNet images. Some conditional samples from this model are shown in Figure 9. While these samples feature recognisable textures associated with the respective classes, they are not globally coherent. Generating coherent samples would require a model that is too large to train in a feasible amount of time.

Figure 9. Samples from a gated PixelCNN trained on 64×64 ImageNet images. From left to right, the classes are 'lion', 'ambulance' and 'cheeseburger'.

D. Effect of auxiliary decoder design on reconstructions

In the main paper, we discussed the effect of the architecture of the auxiliary decoder on the information content and compressibility of the codes. Here, we show how this affects reconstruction quality by using the same trained models to reconstruct some example 64×64 images and visualising the result. Figure 10 shows how changing the number of layers and the mask size of an MSP decoder affects sampled reconstructions. Note that the autoregressive decoder is stochastic, so repeated sampling yields slightly different reconstructions (not shown). We also show difference images to make subtle differences easier to spot. Smaller decoders lead to worse reconstruction quality overall. Larger mask sizes have the same effect and particularly affect local detail.

E. Human evaluation

We conducted a human evaluation of 128×128 samples produced by models trained with feed-forward auxiliary encoders and with MSP auxiliary decoders. In addition to the results provided in the main paper, we report the preferences for either model across the 20 classes we selected in Table 3. While the overall result is quite balanced, there are clear differences at the class level.

F. Nearest neighbours

Although overfitting in likelihood-based models can be identified directly be evaluating likelihoods on a holdout set, we searched for nearest neighbours in the dataset in different feature spaces for some model samples. Following Brock et al. (2019), we show nearest neighbours using L2 distance in pixel space, as well as in 'VGG-16-fc7' and 'ResNet-50-avgpool' (feature spaces obtained from pre-trained discriminative models on ImageNet) in Figures 11, 12 and 13 respectively.

G. Additional reconstructions

We provide some additional autoregressive autoencoder reconstructions of 128×128 images in Figure 14 to further

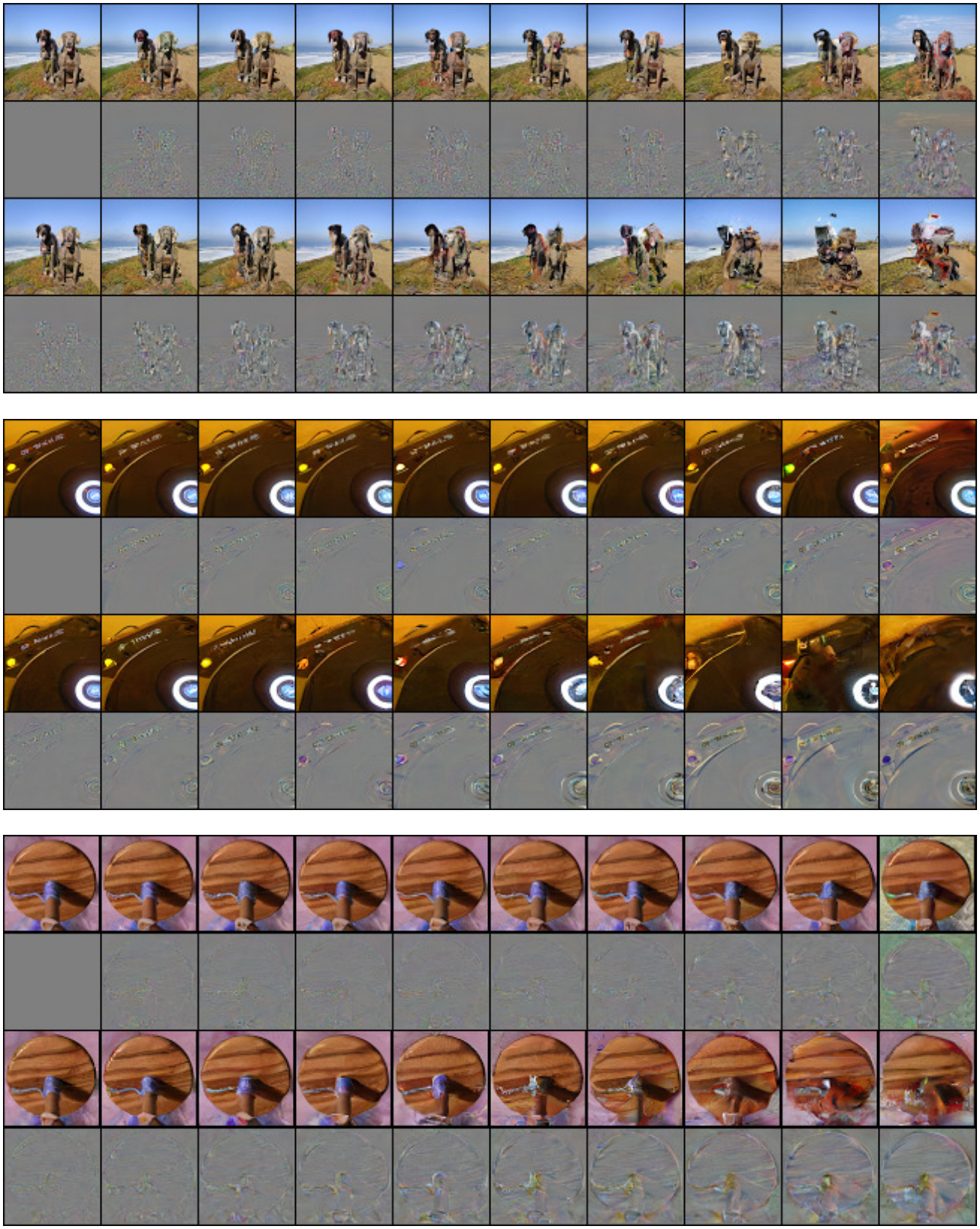


Figure 10. Reconstructions from MSP-trained codes with different auxiliary decoder hyperparameters. The top left image in each grid is the original 64×64 image. The rest of row 1 shows sampled reconstructions for decreasing decoder depth (16, 14, 12, 10, 8, 6, 4, 2, 0 layers respectively) with mask size 5×5. Row 3 shows reconstructions for increasing mask size (1, 3, 5, 7, 9, 11, 13, 15, 17, 19 respectively) with 2 decoder layers. Rows 2 and 4 show the difference between the rows above and the original image.

Figure 11. Nearest neighbours in pixel space. The generated image is in the top left.

Figure 12. Nearest neighbours in VGG-16-fc7 (Simonyan & Zisserman, 2015) feature space. The generated image is in the top left.

Figure 13. Nearest neighbors in ResNet-50-avgpool (He et al., 2016a) feature space. The generated image is in the top left.

Table 3. Preference for 128×128 samples from hierarchical models trained with feed-forward auxiliary decoders and with MSP auxiliary decoders, for 20 classes (50 samples per model per class, 500 random pairwise comparisons by 3 raters, 1,500 answers per class in total).

	PREFERENCE	
CLASS	FEED-FORWARD	MSP
megalith (649)	36.24%	63.62%
giant panda (388)	36.53%	63.20%
cheeseburger (933)	41.07%	58.80%
Geoffroy's spider monkey (381)	42.00%	57.80%
coral reef (973)	43.20%	56.67%
schooner (780)	46.07%	53.60%
Pomeranian (259)	47.40%	52.60%
white stork (127)	47.63%	51.97%
seashore (978)	50.33%	49.67%
starfish (327)	50.80%	49.00%
volcano (980)	50.93%	48.93%
bookcase (453)	51.13%	48.80%
Granny Smith (948)	51.40%	48.47%
monarch butterfly (323)	51.87%	48.13%
yellow garden spider (72)	52.13%	47.73%
ambulance (407)	52.60%	47.00%
frying pan (567)	55.07%	44.73%
grey whale (147)	55.33%	44.47%
tiger (292)	56.93%	42.87%
Dalmatian (251)	59.93%	39.80%

demonstrate their variability and the differences between both auxiliary decoder strategies. Figure 15 contains additional reconstructions of 256×256 images. Figure 16 shows reconstructions from an autoencoder that directly compresses 256×256 images to single-channel 32×32 8-bit codes (MSP, 8 auxiliary decoder layers, mask size 5×5), resulting in a significant degradation in visual fidelity.

H. Additional samples

More samples are available online at `https://bit.ly/2FJkvhJ` in original quality (some figures in the paper are compressed to save space). Figure 17 are 128×128 samples for a model using a feed-forward auxiliary decoder. Figure 18 are 128×128 samples for a model using a MSP auxiliary decoder. Figure 19 are 256×256 samples from a model using feed-forward auxiliary decoders. Figure 20 are 256×256 samples from a model using MSP auxiliary decoders.

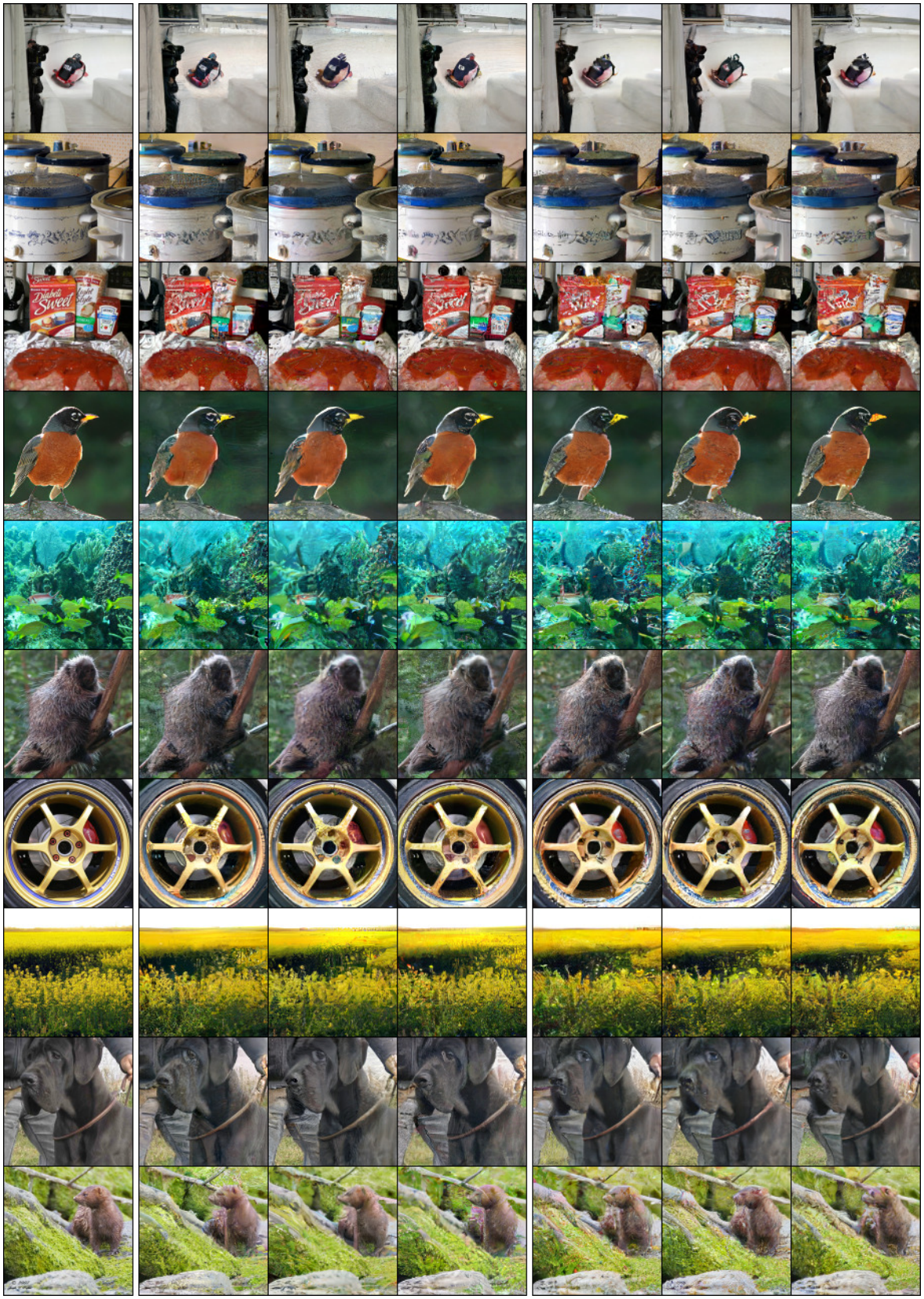


Figure 14. Additional autoregressive autoencoder reconstructions of 128×128 images. Left: original images. Middle: three different sampled reconstructions from models with a feed-forward auxiliary decoder trained with the MSE loss. Right: three different sampled recontructions from models with an MSP auxiliary decoder with mask size 3×3. The sampling temperature was 0.99.

Figure 15. Additional autoregressive autoencoder reconstructions of 256×256 images. Left: original images. Middle: three different sampled reconstructions from models with a feed-forward auxiliary decoder trained with the MSE loss. Right: three different sampled recontructions from models with an MSP auxiliary decoder with mask sizes 5×5 (level 1) and 3×3 (level 2). The sampling temperature was 0.99.

Figure 16. Autoregressive autoencoder reconstructions of 256×256 images, using a single autoencoder (MSP auxiliary decoder, mask size 5×5). The reconstructed images are significantly degraded in terms of visual fidelity. The sampling temperature was 0.99. Please refer to Figure 14 for the original images and a comparison with 2-level reconstructions.

Figure 17. Lorikeet.

Figure 18. Lorikeet.

Figure 19. Ostrich.

Figure 20. Red admiral.