GENERATING HIGH FIDELITY IMAGES WITH SUBSCALE PIXEL NETWORKS AND MULTIDIMENSIONAL UPSCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

The unconditional generation of high fidelity images is a longstanding benchmark for testing the performance of image decoders. Autoregressive image models have been able to generate small images unconditionally, but the extension of these methods to large images where fidelity can be more readily assessed has remained an open problem. Among the major challenges are the capacity to encode the vast previous context and the sheer difficulty of learning a distribution that preserves both global semantic coherence and exactness of detail. To address the former challenge, we propose the Subscale Pixel Network (SPN), a conditional decoder architecture that generates an image as a sequence of image slices of equal size. The SPN compactly captures image-wide spatial dependencies and requires a fraction of the memory and the computation. To address the latter challenge, we propose to use Multidimensional Upscaling to grow an image in both size and depth via intermediate stages corresponding to distinct SPNs. We evaluate SPNs on the unconditional generation of CelebAHQ of size 256 and of ImageNet from size 32 to 128. We achieve state-of-the-art likelihood results in multiple settings, set up new benchmark results in previously unexplored settings and are able to generate very high fidelity large scale samples on the basis of both datasets.

1 INTRODUCTION

A successful generative model has two core aspects: it produces targets that have high fidelity and it generalizes well on held-out data. Autoregressive (AR) models trained by conventional maximum likelihood estimation (MLE) have produced superior scores on held-out data across a wide range of domains such as text (Vaswani et al., 2017; Wu et al., 2016), audio (van den Oord et al., 2016a), images (Parmar et al., 2018) and videos (Kalchbrenner et al., 2016). These scores are a measure of the models' ability to generalize in that setting. From the perspective of sample fidelity, the outputs generated by AR models have also achieved state-of-the-art fidelity in many of the aforementioned domains with one notable exception. In the domain of unconditional large-scale image generation, the default AR samples tend to not show long-range structure or semantic coherence.

One source of difficulties impeding high-fidelity image generation is the multi-faceted relationship between the MLE scores achieved by a model and the model's sample fidelity. On the one hand, MLE is a well-defined measure as improvements in held-out scores generally produce visible improvements in the visual fidelity of the samples. On the other hand, MLE forces the model to support the entire empirical distribution as opposed to, for instance, adversarial methods (Arora & Zhang, 2017). This guarantees the model's ability to generalize at the cost of allotting capacity to parts of the distribution that are irrelevant to fidelity. A second source of difficulties arises from the high dimensionality of large images. A $256 \times 256 \times 3$ image has a total of 196,608 positions that need to be architecturally connected in order to learn dependencies among them and that each require sufficient capacity to represent their respective contexts. These requirements translate to large amounts of memory and computation.

To address the former difficulties, we aim to learn the full distribution over 8-bit RGB images of size up to 256×256 well enough so that the samples will have high fidelity. We aim to guide the model to focus first on visually more salient bits of the distribution and later on the visually less

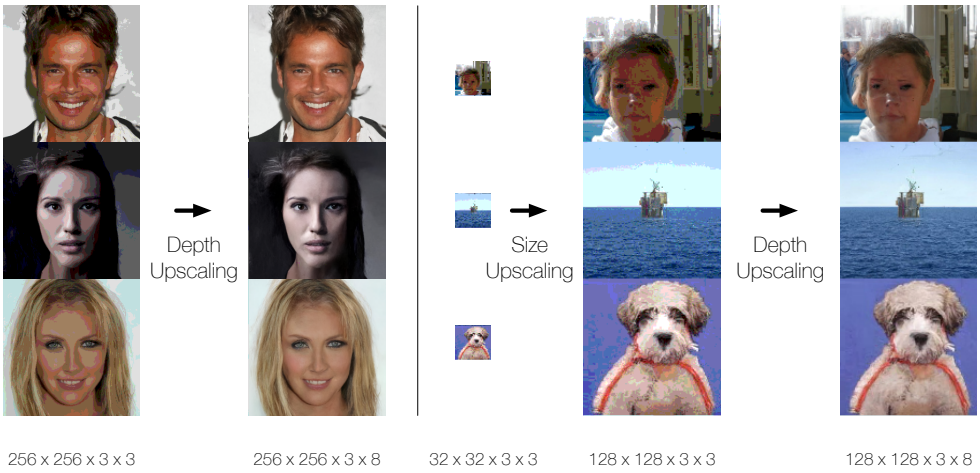
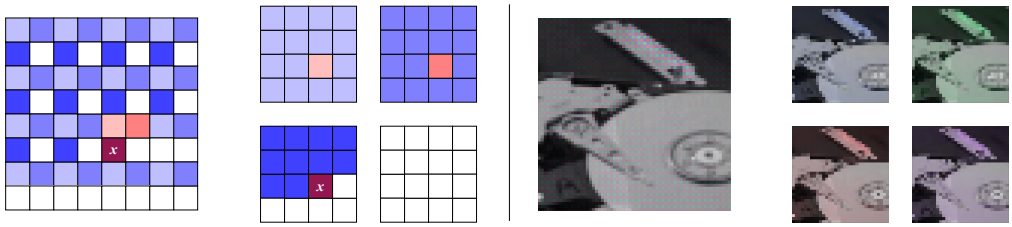


Figure 1: A representation of Multidimensional Upscaling. Left: depth upscaling is applied to a *generated* 3-bit 256×256 RGB subimage from CelebAHQ to map it to a full 8-bit 256×256 RGB image. Right: size upscaling followed by depth upscaling are applied to a *generated* 3-bit 32×32 RGB subimage from ImageNet to map it to the target resolution of the 8-bit 128×128 RGB image.

salient bits. We identify two visually salient subsets of the distribution: first, the subset determined by image slices of smaller size (e.g. 32×32) sub-sampled from the original image at all positions; and secondly, the subset determined by the few (e.g. 3) most significant bits of each RGB channel in the image. We use *multidimensional upscaling* to map from one subset of the distribution to the other one by upscaling images in size or in depth. For example, the generation of a 128×128 8-bit RGB image proceeds by first upscaling it in size from a 32×32 3-bit RGB image to a 128×128 3-bit RGB image; we then upscale the resulting image in depth to the original resolution of the 128×128 8-bit RGB image. We thus train three networks: (a) a decoder on the small size, low depth image slices subsampled at every n pixels from the original image with the desired target resolution; (b) a size-upscaling decoder that generates the large size, low depth image conditioned on the small size, low depth image; and (c) a depth-upscaling decoder that generates the large size, high depth image conditioned on the large size, low depth image. Figure 1 illustrates this process.

To address the latter difficulties that ensue in the training of decoders (b) and (c), we develop the Subscale Pixel Network (SPN) architecture. The SPN itself divides an image of size $N \times N$ into images of size $\frac{N}{S} \times \frac{N}{S}$ sliced out at interleaving positions, which implicitly also captures a form of size upscaling. The decoding part of the SPN acts over image slices with the same spatial structure and it can share weights for all of them. The $N \times N$ image is generated one slice at a time conditioned on previously generated slices in a way that encodes a rich spatial structure. The previously generated slices are embedded by way of the conditioning part of the SPN. The SPN is an independent image decoder with an implicit size upscaling mechanism, but it can also be used as an *explicit* size upscaling network by initializing the first slice of the SPN input with one generated separately during step (a).

We evaluate extensively the performance the SPNs and the size and depth upscaling methods both quantitatively and from a fidelity perspective on two unconditional image generation benchmarks, CelebAHQ-256 and ImageNet of various sizes up to 128. From a MLE scores perspective, we compare with previous work to obtain state-of-the-art results on CelebAHQ-256, both at full 8-bit resolution and at the reduced 5-bit resolution (Kingma & Dhariwal, 2018), and on ImageNet-64. We also establish MLE baselines for ImageNet-128 and ImageNet-256. From a sample fidelity perspective, we show the strong benefits of multidimensional upscaling as well as the benefits of the SPN. We produce CelebAHQ-256 samples (at full 8-bit resolution) with measurable generalization that are of similar visual fidelity to those produced with methods such as GANs that lack an intrinsic measure of generalization (Mescheder, 2018; Karras et al., 2017). We also produce some of the first successful samples on unconditional ImageNet-128 (also at 8-bit) showing again the striking impact of the SPN and of multidimensional upscaling on sample quality and setting a fidelity baseline for future methods.



(a) (b)

Figure 2: The receptive field in a Subscale Pixel Networks (a) and the four image slices subsampled from the original image (b)

2 MODEL

2.1 CONVENTIONAL GENERATION ORDERING

A standard AR image model such as the PixelCNN (van den Oord et al., 2016b) generates an image starting at the top-left position and ending at the bottom-right position, fully generating the three 8-bit channels of each pixel in a given position:

$$P(\mathbf{x}) = \prod_{h=1}^{H} \prod_{w=1}^{W} \prod_{c}^{\{R,G,B\}} P(x_{h,w,c}|\mathbf{x}_{<}) \tag{1}$$

where $\mathbf{x}_{<}$ corresponds to all previously generated intensity values in the ordering. Figure 3(a) illustrates this ordering. Each conditional distribution $P(x_{h,w,c}|\mathbf{x}_{<})$ is parametrized by a deep neural network (van den Oord et al., 2016b).

2.2 SUBSCALE ORDERING IN IMAGES

We define an alternative ordering that divides a large image into a sequence of equally sized slices and has various core properties. It makes it easy to compactly encode long-range dependencies across the many pixels in the large images. It induces a spatial structure over the original image by aligning the subsampled slices; this also has an implicit size upscaling side effect. From the perspective of the neural architecture, it makes it possible for the same decoder within the SPN to be consistently applied to all slices, since they are structurally similar; the smaller slices also allow for self-attention (Vaswani et al., 2017) in the SPN to be used without local contexts. We think of the ordering as the two-dimensional analogue of the one-dimensional Subscale ordering introduced in Kalchbrenner et al. (2018).

The subscale ordering is defined as follows:

$$P(\mathbf{x}) = \prod_{i=1}^{S} \prod_{j=1}^{S} \prod_{h=1}^{H/S} \prod_{w=1}^{W/S} \prod_{c}^{\{R,G,B\}} P(x_{i+S*h,j+S*w,c}|\mathbf{x}_{<}) \tag{2}$$

where $\mathbf{x}_{<}$ corresponds to all previously generated intensity values according to this ordering. Figure 3(d) illustrates the subscale ordering. In essence, a scaling factor S is selected and each slice of size $H/S \times W/S$ is obtained by selecting a pixel every S pixels in both height and width; there are S^2 interleaved slices in the original image.

2.3 SIZE UPSCALING IN SUBSCALE ORDERING

The subscale ordering itself already captures size upscaling implicitly. Analogous to a conventional multi-scale ordering van den Oord et al. (2016b), as depicted in 3(b), we can perform size upscaling *explicitly*, by tranining a single slice decoder on subimages and generate the first slice of a Subscale ordering from the single slice decoder itself. The rest of the image is then generated according to the Subscale ordering by the main network (see 3(e)). The single-slice model can be trained on just the

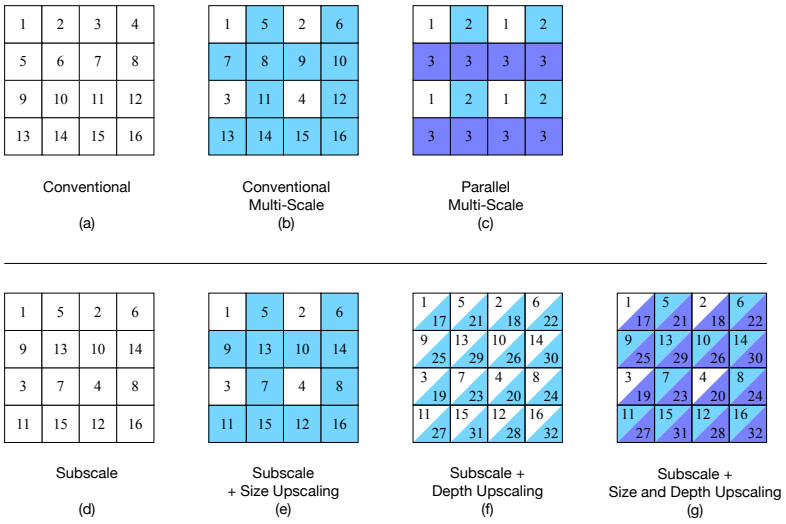


Figure 3: Different generation ordering schemes, where the numbers indicate the step-by-step order. Distinct colors correspond to distinct neural networks. (a) and (b) are from (van den Oord et al., 2016b). (c) is from (Reed et al., 2017). The Subscale ordering alone, with size-only, depth-only and with multidimensional upscaling are, respectively, in blocks (d), (e), (f) and (g).

first slices of images, or on slices at all positions in all images given the shared spatial structure among the slices. For this reason, the same SPN that captures the Subscale ordering can act simultaneously as a full-blown image model as well as a size upscaling model if initialized with the outputs of a single-slice decoder. A separate formulation of size upscaling is the Parallel Multi-Scale ordering where the pixels in an image are doubled at every stage by distinct neural networks and are generated in parallel without sequentiality; see 3(c).

2.4 Depth Upscaling

Multidimensional upscaling applies upscaling not just in the height and width of the image, but also in the remaining dimension that is channel depth. This is performed in stages such that a network first generates the d_1 most significant bits of an image using a conventional or subscale ordering; then a second network generates the next d_2 most significant bits of the image conditioned on *all* the d_1 bits of the image; and so on to further stages. Using the conventional ordering as basis, the first stage of depth upscaling looks as follows:

$$P(\mathbf{x}^{:\mathbf{d_1}}) = \prod_{h=1}^{H} \prod_{w=1}^{W} \prod_{c}^{\{R,G,B\}} P(\mathbf{x}^{:\mathbf{d_1}}_{\mathbf{h,w,c}} | \mathbf{x}^{:\mathbf{d_1}}{}_{<}) \tag{3}$$

Next, a second stage of depth upscaling has the following form, conditioned on the first d_1 bits of each channel:

$$P(\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}) = \prod_{h=1}^{H} \prod_{w=1}^{W} \prod_{c}^{\{R,G,B\}} P(\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}_{\mathbf{h,w,c}} | \mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}{}_{<}, \mathbf{x}^{:\mathbf{d_1}}) \tag{4}$$

We don't share weights among the networks at different stages of depth upscaling. We note that in depth upscaling bits of lower significance are only generated when the more significants bits at all positions have been generated in a previous stage. Just like for size upscaling from the previous section, the goal of multidimensional upscaling is to let the model focus on visually salient bits of an image unaffected by less salient and less predictable bits of the image.

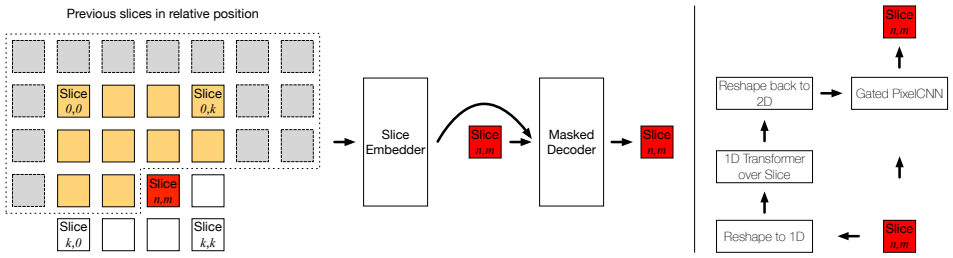


Figure 4: (a) The architecture of a Subscale Pixel Network, with a conditional and a decoding part. (b) Scheme of the parts in the decoder itself.

3 Architecture

3.1 Challenges in Training Conventional Decoders on Large Images

Using a conventional generation ordering, models such as PixelRNN, PixelCNN (van den Oord et al., 2016b) and Image Transformer (Parmar et al., 2018) construct a representation of the generated context for each dimension of each pixel. As the context and the number of positions grow quadratically in the height or width of the image, for large images constructing such representations becomes unfeasible for multiple reasons. On the one hand, the sheer memory requirements of multiple layers for up to 196,608 distinct positions that occur in a 256×256 RGB image can be hard to fit on a single accelerator even at a batch size of 1. In addition, for larger images, presumably more layers are needed for the decoder to capture the larger number of previously generated pixels and the global dependencies between the values - even when these are within the range of a large receptive field - can be significantly harder to uphold. Alternatives such as cropping images within the decoders (Kalchbrenner et al., 2016) does not capture global dependencies, while model parallelism, though technically feasible with the joint use of a very large number of accelerators, does not overcome the challenges in learning the global structure.

3.2 Subscale Pixel Network

To address the challenges, we devise the SPN, an architecture that embodies the Subscale ordering from Section2.2. For an image of size $H \times W \times 3 \times D$, where D is the number of bits used for the current generation stage, one first chooses a scaling factor S and obtains the S^2 slices of the original image of size $H/S \times W/S \times 3 \times D$. We use $H = W = 256$ and $S = 8$ as well as $H = W = 128$ and $S = 4$, for the larger images we process, so that the slices have size $32 \times 32 \times 3 \times D$. The slices are arranged on a two-dimensional grid - analogous to pixels in the image - such that slice at meta-position $0, 0$ is that which contains the pixel at position $0, 0$ in the original image, slice at meta-position $0, 1$ is that which contains the pixel at position $0, 1$, and so on until slice at meta-position $S - 1, S - 1$.

The Subscale Decoder architecture is composed of two parts: an embedding part for slices at preceding meta-positions and the decoder proper for the current slice that is being generated (Figure 4 (a)). The embedding part is a convolutional neural network with residual blocks that takes as input preceding slices that are concatenated along the *depth* dimension. One detail is the way the slices are ordered along the channel dimension when concatenated. As illustrated in Figure 4 (a), empty padding slices are used to preserve *relative* meta-positions of each preceding slice with respect to the current target slice. For example, the slice above any target slice in the two-dimensional meta-grid is always aligned in the same position along the depth axis in the input. These padding slices also ensure that the depth of the input slice tensor remains the same for all target slices. In addition to the slice tensor, the embedding part also receives as input the meta-position of the target slice as an embedding of 8 units tiled spatially across the slice tensor. The pixel intensity values are also embedded as one-hot indices of size 8.

The decoder takes as input the encoded slice tensor in a position-preserving manner: each position in the target slice is given as input the encoded representations of pixels at that same position in the preceding slices. In addition it processes the target slice in the raster-scan order. The decoder that we use is a hybrid architecture combining masked convolution and self-attention (Chen et al., 2017).

Figure 5: 8-bit 256x256 RGB CelebA-HQ samples from SPN with Depth-Upscaling. Temperature is 0.99 for the low-bit-depth image and otherwise 1.0

We employ an initial 1D self-attention network Vaswani et al. (2017) that is used to gather the entire available context in the slice (see Figure 4(b)). The slice is reshaped into a 1D tensor before it is given as input to masked 1D self-attention layers; the masking is performed over the previous pixels only (as opposed to over the current RGB channels) in the self-attention layers. Then the output of the layers is reshaped back into a 2D tensor and given as input to a Gated PixelCNN van den Oord et al. (2016c) with full masking over pixels and channel dimensions. We can see how memory requirements are significantly lower - up to $S^2 = 64\times$ lower with $S = 8$ - due to the smaller size of the slices and their compact concatenation along the channel dimension of the input tensor. In addition, due to this structure, the entire previously generated context is captured at each position of the decoder.

3.3 Size-Upscaling SPN

As seen in Section 2.3, the SPN naturally serves as a size-upscaling network when the first slice of the input tensor is initialized with with an externally trained subimage. In our experiments, we ensure that the smaller subimages used for the initialization and those used in the training of the subimage decoder are identical to each other.

3.4 Depth-Upscaling SPN

Analogously, the SPN can be used to upscale the depth of the channels of an image. The image to be upscaled is itself divided into slices by the subscale method and the slices are then concatenated along the channel dimension into a slice tensor for the conditioning image. The latter is then added as a fixed additional input to the embedding part of the SPN.

Figure 6: Left: 8-bit 128x128 RGB ImageNet samples from SPN with depth upscaling only. Right: 8-bit 128x128 RGB ImageNet samples from SPN with full-blown Multidimensional Upscaling. Temperature is 0.99 for the initial 32x32 sub-image and otherwise 1.0

	ImageNet 32x32	ImageNet 64x64
Gated PixelCNN	3.83	3.57
Parallel Multiscale	3.95	3.70
PixelSNAIL	3.80	-
Image Transformer	**3.77**	-
Glow	4.09	3.81
Decoder baseline	3.79	**3.52**
SPN	-	3.53
SPN + Depth Upscaling	-	3.53 (0.63, 2.90)
SPN + 16x16 slices	3.85	-
SPN + 8x8 slices	3.91	-

Table 1: Negative Log-likelihood (NLL) scores for Downsampled Imagenet (van den Oord et al., 2016b) in bits/dim. The parenthesized numbers in Depth Upscaling rows indicate NLL for $P(\mathbf{x}^{:\mathbf{d_1}})$ and $P(\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}_{\mathbf{h},\mathbf{w},\mathbf{c}} | \mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}{}_{<}, \mathbf{x}^{:\mathbf{d_1}})$ respectively.

4 Experiments

We demonstrate experimentally that our model is capable of high fidelity samples at high resolution, producing unconditional CelebA-HQ samples of comparable quality to "Glow", while dramatically improving the log likelihood. Furthermore, we show that these results extend to high-resolution ImageNet images, with state-of-the-art log-likelihoods at 128x128 by a large margin, and the first benchmark on 256x256 ImageNet. Crucially unconditional samples at these resolutions are characterized by unpredented global coherence.

In all experiments we choose S so that slices are shaped like 32×32 images. For instance, when we model 128×128 images, the subscaling rate S is set to 4. The result of this scheme is that as the image resolution N^2 changes, the only changes to the SPN configuration is the number of weights in the first encoder layer (due to the increase in the number of input channels).

The SPN's context-embedding network contains 5 convolutional layers and 6-8 self-attention layers depending on the dataset. The masked decoder consists of a PixelCNN with 15 layers in all experiments. The 1D Transformer in the decoder has between 8 and 10 layers depending on the dataset. See Table 4 for all dataset-specific hyperparameter details.

	ImageNet 128 x 128	ImageNet 256 x 256
Parallel Multiscale	3.55	-
SPN	**3.08**	**2.97**
SPN + Depth-Upscaling	**3.08 (0.46, 2.62)**	3.01 (0.40, 2.61)

Table 2: NLL scores for high-resolution Imagenet in bits/dim. The parenthesized numbers in Depth Upscaling rows indicate NLL for $P(\mathbf{x}^{:\mathbf{d_1}})$ and $P(\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}_{\mathbf{h},\mathbf{w},\mathbf{c}}|\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}{}_{<}, \mathbf{x}^{:\mathbf{d_1}})$ respectively.

	ImageNet 64 x 64 (5bit)	CelebA-HQ 256 x 256 (5bit)
Glow	1.76	1.03
SPN	**1.41**	**0.61**

Table 3: Negative Log-likelihood scores for 5-bit datasets in bits/dim.

4.1 DOWNSAMPLED IMAGENET AT 32×32 AND 64×64

We first benchmark the performance of the decoder alone (i.e. no subscaling, Figure 4(b)) and show that it compares favorably to state of the art models on 32×32 Downsampled ImageNet (see Table 1). We find that SPN hurts in this low-resolution setting with $S = 2$ and even further with $S = 4$. This is likely because the sizes of the resulting image slices becomes very small and the image highly coarse grained; less spatial structure is captured when the slices are that small and the S factor sufficiently large.

On 64×64 Downsampled ImageNet, we achieve a state of the art log-likelihood of 3.52 bits/dim. We hypothesize that PixelSNAIL would achieve a similar score, but results at this resolution were not reported in Chen et al. (2017). At this resolution, SPN scores similarly with 3.53 bits/dim. The improvement over Glow in the 5-bit setting is particularly striking (Table 3).

4.2 IMAGENET AT 128×128 AND 256×256

Parallel Multiscale PixelCNN (Reed et al., 2017) is the only model in the literature which reports log-likelihood on 128×128 ImageNet. SPN improves the log-likelihood over this model from 3.55 bits/dim to 3.08 bits/dim (see Table 2). Figure 6 gives 128×128 8-bit ImageNet samples for both the setting of depth upscaling only and of complete multidimensional upscaling. These settings do not affect the NLL, but the samples with depth upscaling show significant semantic coherence that is usually lacking in samples without upscaling. In addition, multidimensional upscaling seems to increase the overall rate of success of the samples. Additional intermediate ImageNet samples can be seen in Figures 10, 11 and 12 in the Appendix.

4.3 CELEBAHQ

At 256×256, we can produce high-fidelity samples of celebrity faces; the quality compares well to the samples of other models such as Glow and GANs. We show in Table 3 that the likelihoods are significantly better. Figure 5 showcases some samples for 8-bit CelebAHQ-256 at nearly no changes in temperature. Figure 7 in the Appendix includes 5-bit samples, Figure 8 and Figure 9 include 3-bit samples at respectively no temperature changes and at temperature setting 0.95.

5 CONCLUSION

We have introduced Subscale Pixel Networks and shown that they achieve state-of-the-art on multiple previously unexplored settings on CelebAHQ and ImageNet. We have also shown the strong effectiveness of Multidimensional Upscaling in guiding the generation of SPNs and letting them prioritize visually salient part of the distribution. The generated samples suggests strong semantic coherence and exactness of details even at the large scale size of full 8-bit 128×128 and 256×256 samples.

REFERENCES

Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study. *CoRR*, abs/1706.08224, 2017. URL http://arxiv.org/abs/1706.08224.

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. *CoRR*, abs/1712.09763, 2017. URL http://arxiv.org/abs/1712.09763.

Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *CoRR*, abs/1610.00527, 2016. URL http://arxiv.org/abs/1610.00527.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. *CoRR*, abs/1802.08435, 2018. URL http://arxiv.org/abs/1802.08435.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. URL http://arxiv.org/abs/1710.10196.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *CoRR*, abs/1807.03039, 2018. URL https://arxiv.org/abs/1807.03039.

Lars M. Mescheder. On the convergence properties of GAN training. *CoRR*, abs/1801.04406, 2018. URL http://arxiv.org/abs/1801.04406.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, and Alexander Ku. Image transformer. *CoRR*, abs/1802.05751, 2018. URL http://arxiv.org/abs/1802.05751.

Scott E. Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gomez Colmenarejo, Ziyu Wang, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. *CoRR*, abs/1703.03664, 2017. URL http://arxiv.org/abs/1703.03664.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016a. URL http://arxiv.org/abs/1609.03499.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016b. URL http://arxiv.org/abs/1601.06759.

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016c. URL http://arxiv.org/abs/1606.05328.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

APPENDIX A: SAMPLES

APPENDIX B: FURTHER ARCHITECTURE DETAILS

Please see Table 4 for all the detailed hyperparameters.

Figure 7: 256x256 CelebA-HQ 5bit samples from SPN

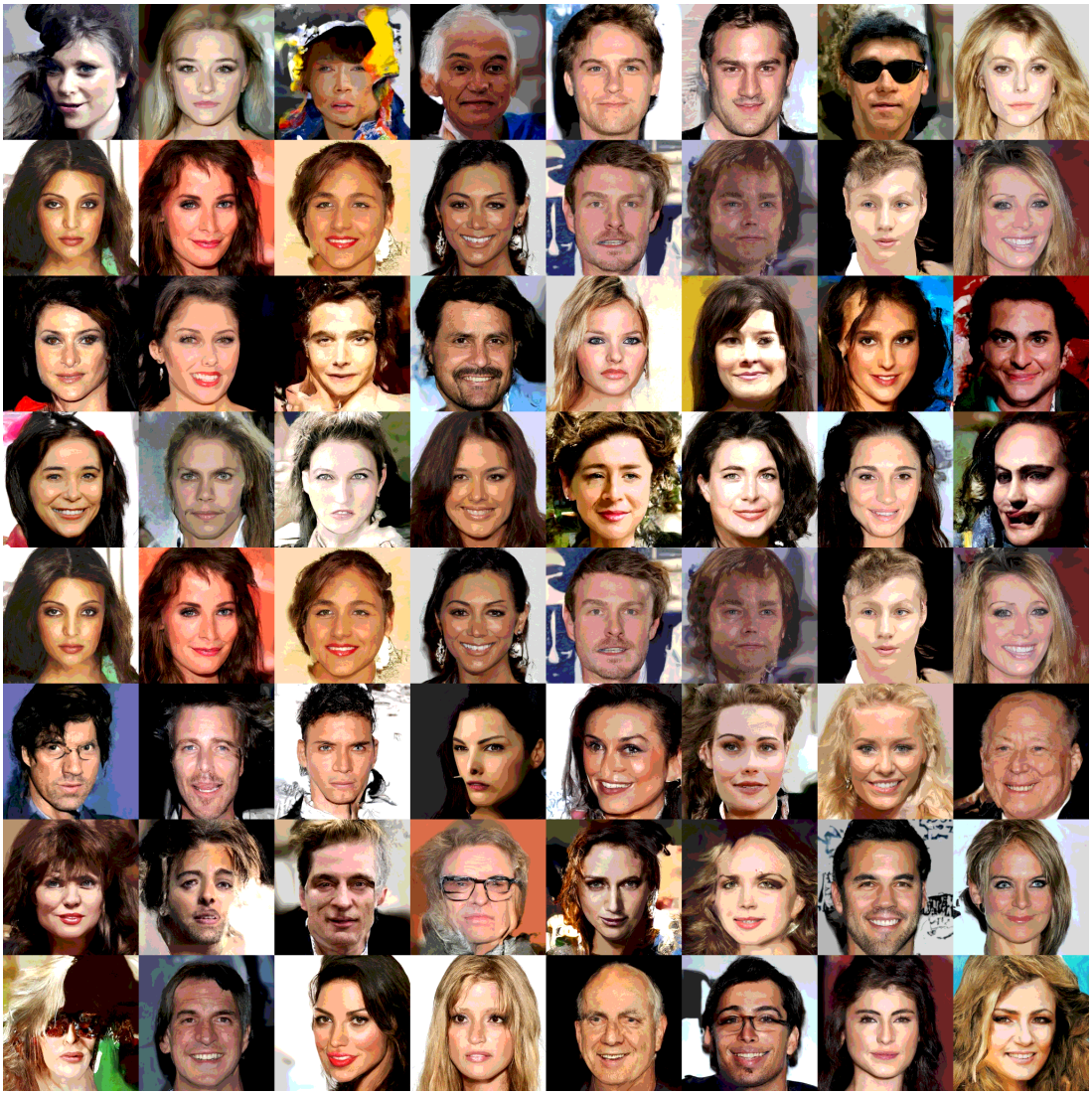


Figure 8: 256x256 CelebA-HQ 3bit samples from SPN

Figure 9: 256x256 CelebA-HQ 3bit samples from SPN with temperature 0.95

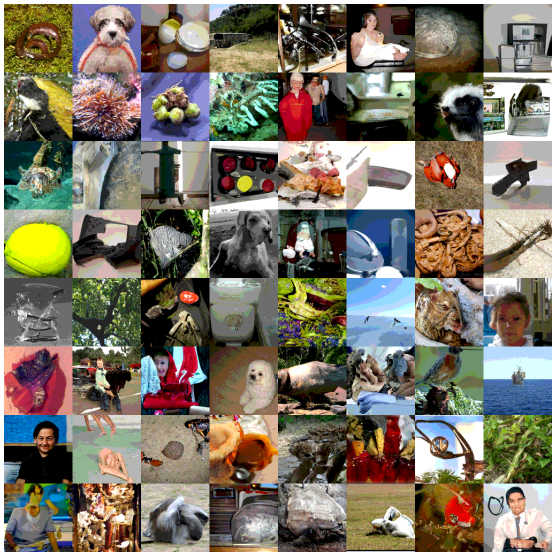


Figure 10: 128x128 ImageNet 3bit; upscaled 32x32 slices

Figure 11: 128x128 ImageNet 3bit samples from model trained on 32x32 slices

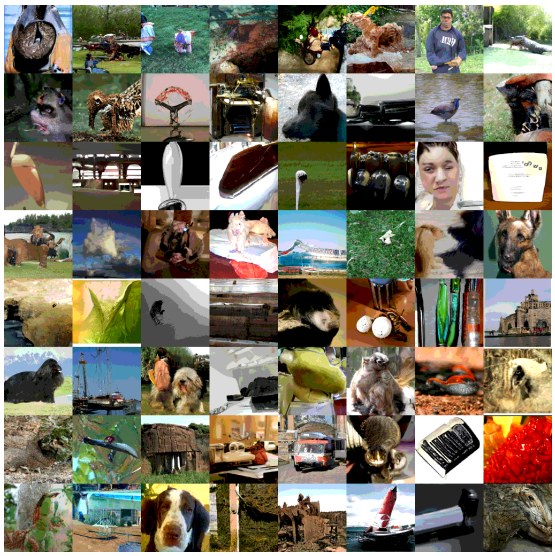


Figure 12: 128x128 ImageNet 3bit samples from SPN

	ImageNet (32, 64, 128, 256)	CelebA HQ
Optimization		
batch size	(1024, 2048, 2048, 2048)	256
learning rate	(sched, sched, 1e-5, 1e-5)	5e-5
rmsprop momentum	0.9	0.9
rmsprop decay	0.95	0.95
rmsprop epsilon	1e-8	1e-8
polyak decay	0.9999	0.9999
Decoder		
PixelCNN layers	15	15
PixelCNN conv channels	(256, 256, 384, 384)	128
PixelCNN residual channels	1280	1280
PixelCNN nonlinearity	gated	gated
PixelCNN filter size	3	3
masked self-attention layers	8	5
attention heads	10	5
attention channels	128	128
attention ffn layer	"parameter_attention"	"parameter_attention"
Slice Embedder		
conv layers	5	5
conv filter size	3	3
conv channels	256	256
residual channels	1024	1024
nonlinearity	relu	relu
self-attention layers	(6, 6, 8, 8)	6
attention heads	(4, 4, 8, 8)	4
attention channels	(64, 64, 128, 128)	64
attention ffn layer	"parameter_attention"	"parameter_attention"

Table 4: SPN Hyperparameters. A learning rate marked "sched" utilizes a piecewise-constant schedule starting at 1e-4, and decreasing to 3e-5 and finally 1e-5 at training steps 50k and 100k respectively.