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 sub-images of equal size. The SPN compactly captures image-wide spatial dependencies and requires a fraction of the memory and the computation required by other fully autoregressive models. To address the latter challenge, we propose to use Multidimensional Upscaling to grow an image in both size and depth via intermediate stages utilising 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 in many of the aforementioned domains with one notable exception. In the domain of unconditional large-scale image generation, AR samples have yet to 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 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.

These difficulties notwithstanding, 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

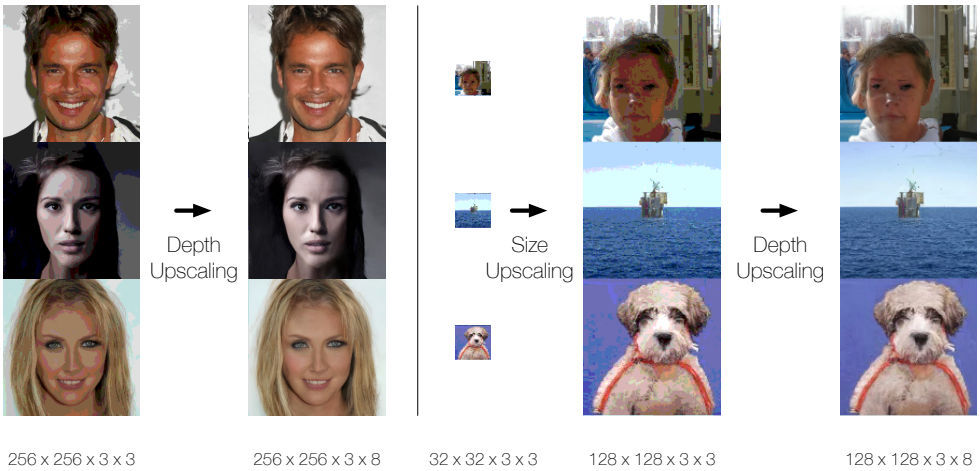

Figure 1: A representation of Multidimensional Upscaling. Left: depth upscaling is applied to a *generated* 3-bit 256×256 RGB subimage from CelebAHQ to map it to a full 8-bit 256×256 RGB image. Right: size upscaling followed by depth upscaling are applied to a *generated* 3-bit 32×32 RGB subimage from ImageNet to map it to the target resolution of the 8-bit 128×128 RGB image. We stress that the rightmost column of both figures are true unconditional samples from our model at full 8bit depth.

to focus first on visually more salient bits of the distribution and later on the visually less salient bits, motivated by the insight that at sampling time, all-else equal, variables later in the ordering are more likely to be conditioned upon low-probability events due to compounding error in the sequence of stochastic choices producing their context. We identify two visually salient subsets of the distribution: first, the subset determined by sub-images ("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 divides an image of size $N \times N$ into sub-images of size $\frac{N}{S} \times \frac{N}{S}$ sliced out at interleaving positions (see Figure 2), which implicitly also captures a form of size upscaling. 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. SPN consists of two networks, a conditioning network which embeds previous slices, and a decoder proper, which predicts a single target slice given the context embedding. 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 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 at sampling time with one generated separately during step (a).

We extensively evaluate the performance of SPN 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.

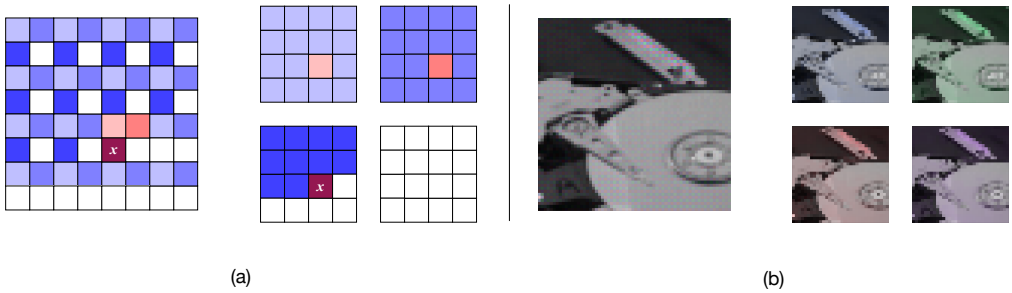

Figure 2: The receptive field in a Subscale Pixel Networks (a) and the four image slices subsampled from the original image (b)

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) 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.

2 Model

2.1 Conventional Generation Ordering

A standard AR image model such as the PixelCNN (van den Oord et al., 2016b) generates an HxW colour 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 and h, w, and c are row, column, and colour channel indices. The raster scan ordering (Figure 3(a)) is conventionally used in Deep AR models. 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. First, it makes it easy to compactly encode long-range dependencies across the many pixels in the large images. It also 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 (Parmar et al., 2018). 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 thus

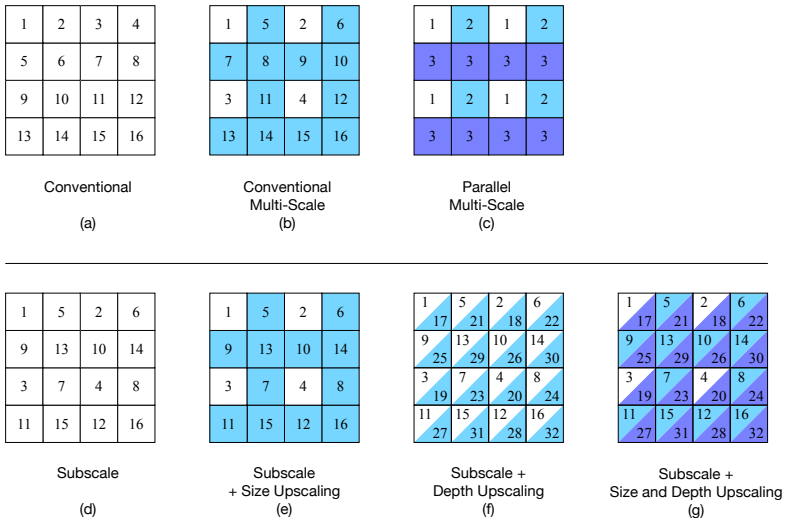

Figure 3: Different generation ordering schemes, where the numbers indicate the step-by-step order. Distinct colors correspond to distinct neural networks. (a) and (b) are from (van den Oord et al., 2016b). (c) is from (Reed et al., 2017). The Subscale ordering alone, with size-only, depth-only and with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

S^2 interleaved slices in the original image, with each specified uniquely by its row and column offset (i, j). We sometimes refer to this offset as the "meta-position" of a slice.

2.3 Size Upscaling in Subscale Ordering

The subscale ordering itself already captures size upscaling implicitly. Analogous to the multi-scale ordering (van den Oord et al., 2016b), and depicted in 3(b), we can perform size upscaling *explicitly*, by training 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 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 (Reed et al., 2017) 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:

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

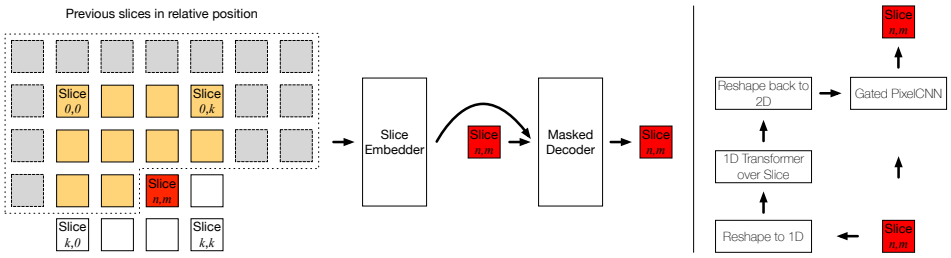

Figure 4: (a) The architecture of a Subscale Pixel Network, with a conditional and a decoding part. (b) Scheme of the parts in the decoder itself.

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.

We note that reorderings with a similar motivation have been considered in Kolesnikov & Lampert (2016a).

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. Existing AR approaches inherently require an amount of computation and memory that is superlinear in the number of pixels. In particular, the quadratic memory requirements of self-attention become severely limiting for images larger than even 32×32, let alone the 196,608 distinct positions we consider in a 256×256 colour image.

Mitigating the sheer memory requirements and computational requirements of encoding the dependencies amongst so many variables often comes at the expense of global context. Modeling choices such as cropping images within the decoders (Kalchbrenner et al., 2016) or performing self-attention over local neighborhoods (Parmar et al., 2018) neglect 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 these challenges, we devise the Subscale Pixel Network (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 $32 \times 32 \times 3 \times D$. This scheme of choosing S such that slices are always 32×32 renders the memory and computation requirements effectively constant as the true image size $H \times W$ changes.

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. This achieves a sort of equivariance in the embedding architecture with respect to the (i, j) offset of a slice. The 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 context embedding network passes its input through a series of self-attention layers and then a series of residual blocks, finally emitting a slice-schaped feature map $\mathbf{s}$ which summarizes the context for the decoder.

The decoder takes as input the encoded slice tensor $\mathbf{s}$ 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). 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, concatenated depth-wise with the output of the slice embedding network, and given as conditioning input to a Gated PixelCNN as in Equation 5 in van den Oord et al. (2016c). The PixelCNN network models the target slice 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 *spatial* 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 Learning

The log-likelihood derived from equation 2 decomposes as a sum over slices. An unbiased estimator of the log-loss is obtained by uniformly sampling a choice of target slice and evaluating its log-probability conditioned upon previous slices as depicted in Figure 4 (a). We perform maximum likelihood learning by doing stochastic gradient descent on this Monte Carlo estimate, with all gradients computed by backpropagation.

3.4 Multidimensional Upscaling with 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 generated subimage. In our experiments, we ensure that the smaller subimages used for the initialization and those used in the training of the SPN decoder are identical to each other.

Analogously, the SPN can be used to upscale the depth of the channels of an image. The image to be depth-upscaled is itself divided into slices by the subscale method (secion 2) and the slices are then concatenated along the channel dimension into a slice tensor for the conditioning image $\mathbf{x}^{:\mathbf{d_1}}$. The latter is then added as a fixed additional input to the embedding part of the SPN in order to model $P(\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}_{\mathbf{h},\mathbf{w},\mathbf{c}}|\mathbf{x}^{\mathbf{d_1}:\mathbf{d_2}}{}_{<}, \mathbf{x}^{:\mathbf{d_1}})$. The model of $P(\mathbf{x}^{:\mathbf{d_1}})$ is a normal SPN, but trained on data with low bit-depth.

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 unprecedented global coherence.

Because our networks operate on small images (32×32 slices), we can train large networks both in terms of the number of hidden units and in terms of network depth. To compensate for noise in the process of sampling a target slice (section 3.3), we use large batch sizes. We comment on the scale of operation in Appendix C.

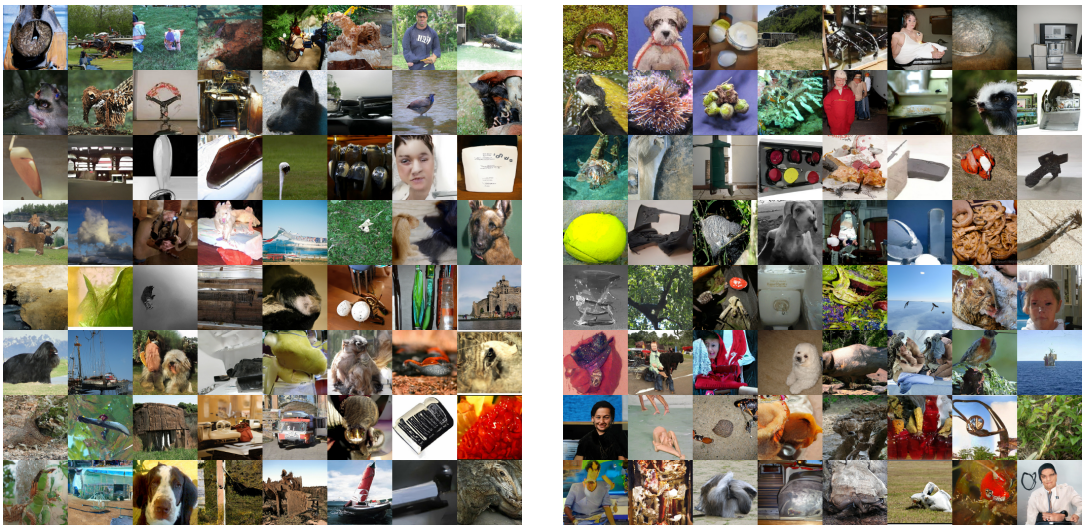

Figure 5: Left: 8-bit 128x128 RGB ImageNet samples from SPN with depth upscaling only. Right: 8-bit 128x128 RGB ImageNet samples from SPN with full-blown Multidimensional Upscaling. Temperature is 0.99 for the initial 32x32 sub-image and otherwise 1.0

	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.

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 (Figure 4(b)) has between 8 and 10 layers depending on the dataset. See Table 4 for all dataset-specific hyperparameter details.

4.1 Downsampled ImageNet at 32×32 and 64×64

We first benchmark the performance of our hybrid 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).

	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.

4.2 IMAGENET AT 128×128 AND 256×256

For these experiments we use the standard ILSVRC Imagenet dataset (Kolesnikov & Lampert, 2016b) resized with Tensorflow's resize_area function. 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 5 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 6 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`.

Norman P. Jouppi, Cliff Young, Nishant Patil, David A. Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, Richard C. Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski,

Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *CoRR*, abs/1704.04760, 2017. URL http://arxiv.org/abs/1704.04760.

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.

Alexander Kolesnikov and Christoph H. Lampert. Deep probabilistic modeling of natural images using a pyramid decomposition. *CoRR*, abs/1612.08185, 2016a. URL http://arxiv.org/abs/1612.08185.

Alexander Kolesnikov and Christoph H. Lampert. Deep probabilistic modeling of natural images using a pyramid decomposition. *CoRR*, abs/1612.08185, 2016b. URL http://arxiv.org/abs/1612.08185.

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.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, François Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018. URL http://arxiv.org/abs/1803.07416.

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

For some samples, the entropy of the softmax output distributions has been artificially reduced via a "temperature" divisor on the logits. Thus when we say the temperature is 0.95, we mean that the logits of a trained model have been divided by this constant at sampling time.

Figure 6: 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

APPENDIX B: FURTHER ARCHITECTURE DETAILS

Please see Table 4 for all the detailed hyperparameters.

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

APPENDIX C: SCALE OF OPERATION

Our experiments operate at a fairly large scale in terms of both the amount of compute used and the size of the networks. To compensate for noise in the process of randomly selecting a target slice, we commensurately increase the batch size so that the number of pixels in a batch is not affected by this subsampling. These large batch sizes (a maximum of 2048) are achieved by increasing the degree of data parallelism by running on Google Cloud TPU pods (Jouppi et al., 2017). For Imagenet 32 we utilized 64 tensorcores. For ImageNet 64, 128 and 256, we used 128 tensorcores. The fast interconnect between these devices affords much faster synchronous gradient computation than would be possible using the same number of GPUs. When overfitting is a problem, as in small datasets like CelebA-HQ, we rather decrease the batch size. Thus on CelebA-HQ, we used only 32 tensorcores.

Our SPN architectures have between $\sim$50M and $\sim$250M parameters depending on the dataset. See Table 4 for the number of parameters in the SPN architecture for each dataset. Bitdepth-upscaling doubles the number of parameters due to using two separate networks with untied weights. Size-upscaling adds more parameters still for the separate decoder-only network which models the first

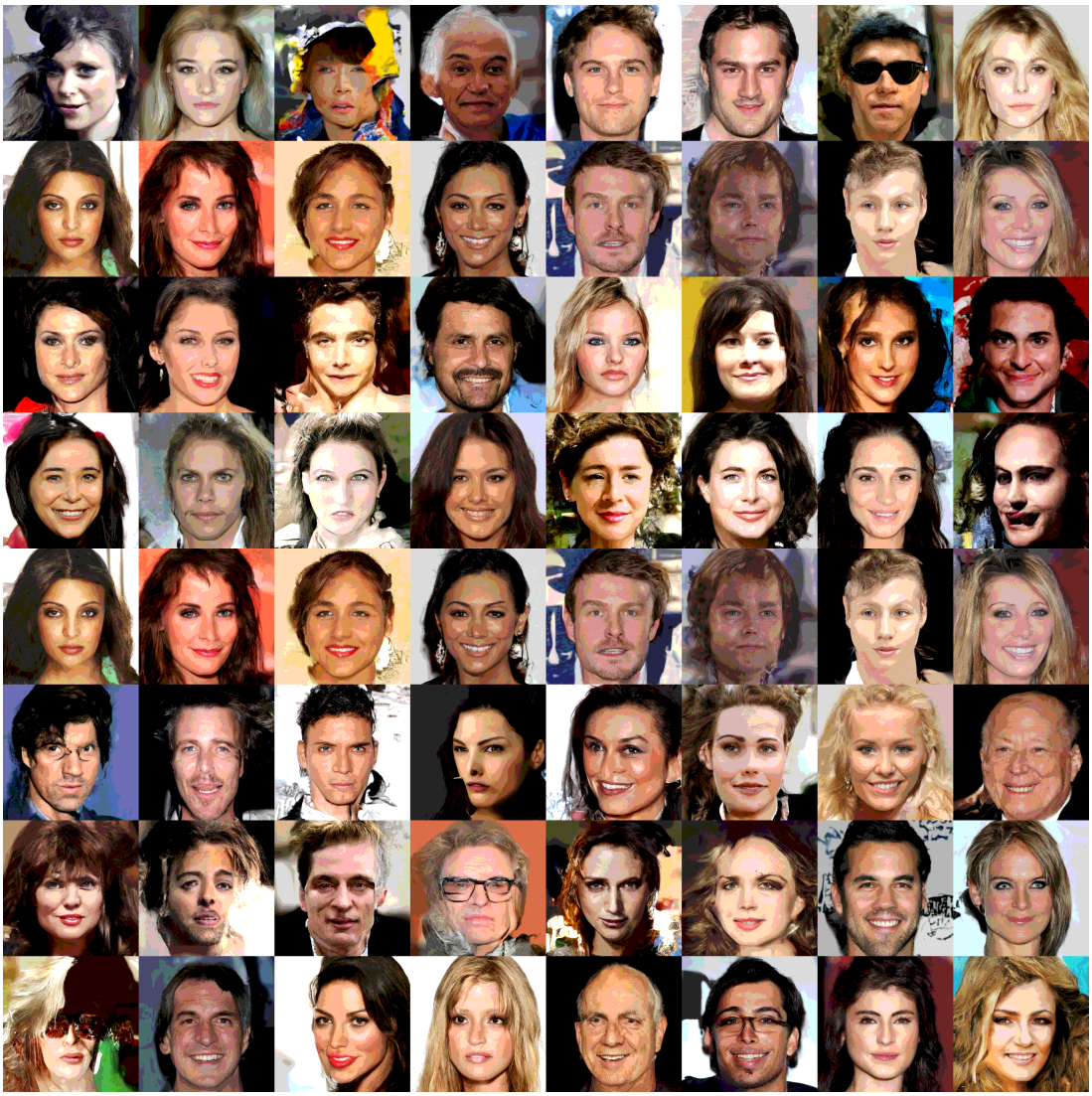

Figure 8: 256x256 CelebA-HQ 3bit samples from SPN

slice as seen in Figure 3 (g). Thus the maximal number of parameters used to generate a sample in the paper is full multidimensional upscaling on ImageNet 128, where the total parameter count reaches $\sim$650M (the decoder-only network used to model the first slice has $\sim$150M parameters).

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

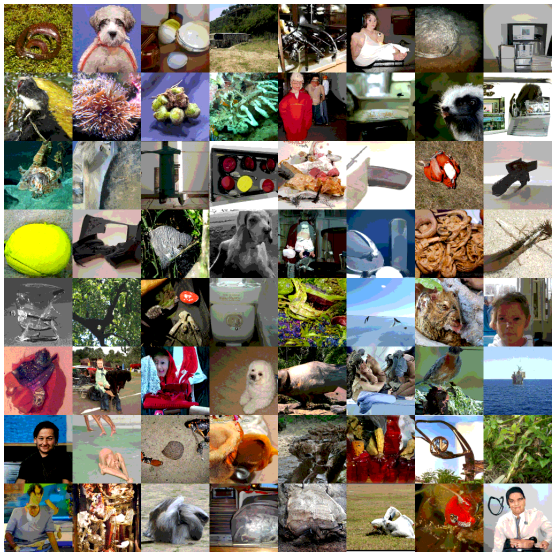

Figure 10: 128x128 ImageNet 3bit; upscaled 32x32 slices

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

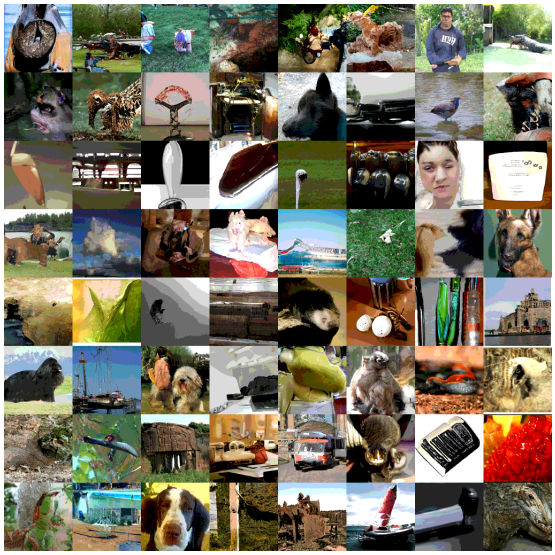

Figure 12: 128x128 ImageNet 3bit samples from SPN

	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"
Number of parameters		
	($\sim$ 150M, $\sim$ 150M, $\sim$ 250M, $\sim$ 250M)	$\sim$ 50M

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. The "attention" parameters listed are configurable hyperparameters of the open source Transformer implementation in Vaswani et al. (2018). The parameter "residual channels" refers to the number of hidden units in residual convolution layers within the Slice Embedder or PixelCNN networks.