A. Quantization matrices

Following the Independent JPEG Group standard we use the following quality parameterized quantization matrix $\mathbf{Q}$ to quantize DCT pixel blocks. For quality q in $[1, 100]$ the matrix is defined as:

$$\mathbf{T}_{\text{luma}} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$s(q) = \begin{cases} 5000/q, & \text{if } q < 50 \\ 200 - 2q, & \text{otherwise} \end{cases}$$

$$\mathbf{Q}_{\text{luma}}(q) = \text{floor}\left(\frac{s(q)\mathbf{T}_{\text{luma}} + 50}{100}\right)$$

For the chrominance components we replace $\mathbf{T}_{\text{luma}}$ with an alternative base matrix $\mathbf{T}_{\text{chroma}}$ which provides stronger quantization:

$$\mathbf{T}_{\text{chroma}} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}.$$

When using block sizes other than 8, we use nearest neighbour interpolation to resize the base matrices $\mathbf{T}_{\text{luma}}$ and $\mathbf{T}_{\text{chroma}}$ to the target size.

B. Architecture details

DCTransformer consists of a Transformer encoder that processes partial DCT images, and three stacked Transformer decoders that process slices from the DCT co-ordinate list (Section 3.2). We use a number of modifications to the original architecture that we found to improve stability, training speed, and memory consumption:

Layer norm placement Following Child et al. (2019) and Parisotto et al. (2020) we use Transformer blocks with layer norm placed inside the residual path, rather than applying layer norm

ReZero We use ReZero (Bachlechner et al., 2020; De & Smith, 2020), multiplying each residual connection with a zero-initialized scalar value, which is optimized jointly with the model parameters. In our experiments we found this to improve training speed and stability to a small degree. The combination of ReZero and our chosen layer norm placement results in residual connections of the following form:

$$\mathbf{H}_l = \mathbf{H}_{l-1} + \alpha_l f_l(\text{layernorm}(\mathbf{H}_{\mathbf{l-1}})), \qquad (10)$$

where $\mathbf{H}_l$ is a sequence of activations at layer l, and f is the residual function.

PAR Transformer Mandava et al. (2020) showed through Neural architecture search that the default alternation of fully connected and self attention layers in Transformer blocks is sub-optimal with respect to performance-speed trade offs. Based on the results of the search they proposed the PAR Transformer, which applies a series of fully connected layers after each self-attention layer, resulting in improved inference speed, and memory savings. We use a PAR Transformer style architecture in DCTransformer encoder and decoders, and while we didn't experiment rigorously with the ratio of fully-connected to self-attention layers, we found that architectures using 2-4 fully connected layers per self-attention layer helped to boost the parameter count at a given memory budget.

Table 3 details the architecture configurations used for our main experiments.

C. Training details

Optimization We train our models using Adam Optimizer (Kingma & Ba, 2015) and train for a fixed number of tokens, where the number of tokens processed in a batch is the number of elements in the target chunk that we apply a loss to. We use a linear warmup of 1000 steps up to a maximum learning rate, and use a cosine decay over the course of training. We train all models using Google Cloud TPUv3 (Google, 2018), and detail the number of cores used during training in Table 3.

Chunk selection policy As described in Section 3.1, we train DCTransformer on input and target chunks selected from larger sequences. If the target chunk is sampled uniformly from the sequence representation of an image, the sample gradient is unbiased because it is a Monte Carlo estimate of the gradient computed on the full sequence. As discussed in Section 2, lossy image compression codecs such as JPEG preferentially discard high frequency data when allocating limited storage capacity. It stands to reason that we may benefit from making an analogous decision when allocating limited model capacity. We found it advantageous for sample quality to bias the selection of target chunks toward the beginning of the sequence, which contains more low frequency information (see Figure 2).

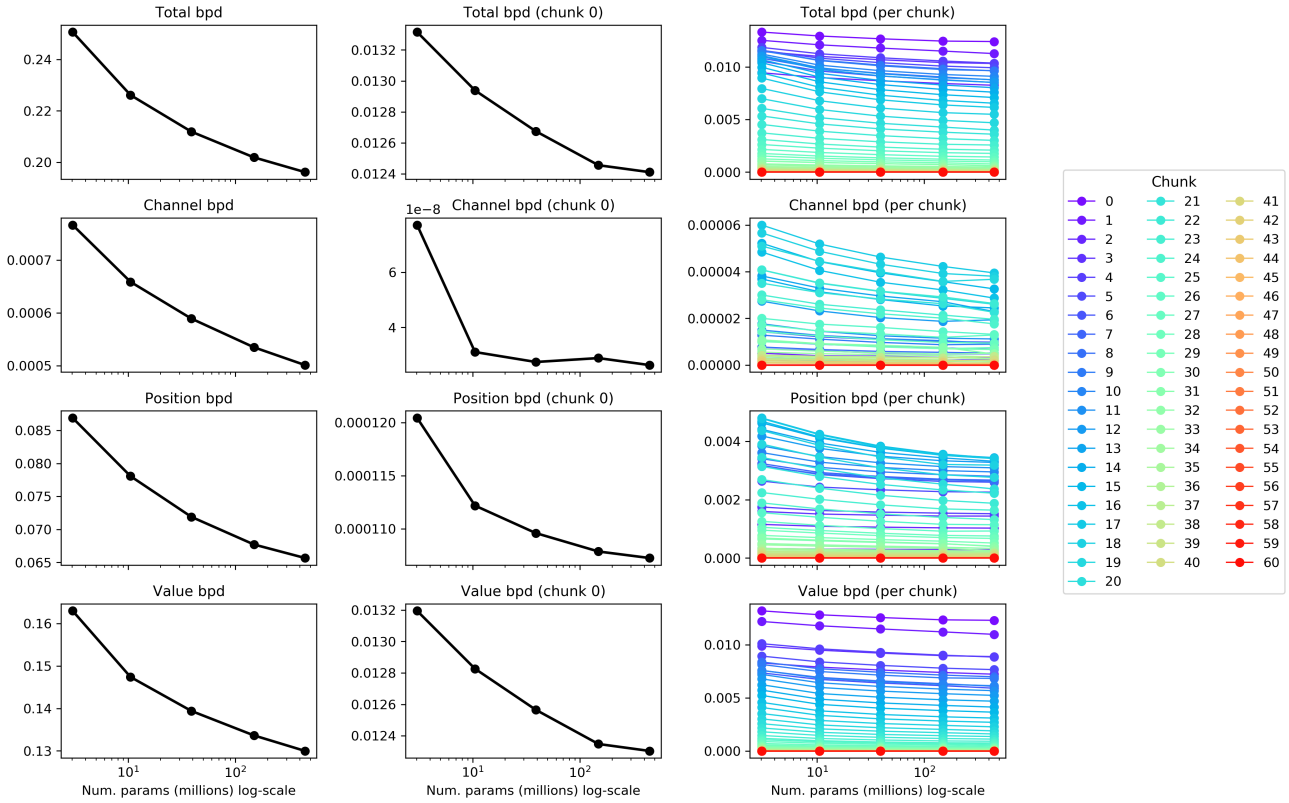

Figure 9. Model performance, reported in bits per image subpixel (bpd) as a function of model size for DCTransformer trained on LSUN bedrooms. We report the total bpd, as well as the contributions from the channel, position and value distributions. We additionally report bpd broken down by sequence chunk, where each chunk corresponds to 896 sequence elements. The model size is reported in terms of the number of total parameters, ranging from roughly 3 million to 448 million.

We select chunks using the following process: A sequence of length L is split into chunks of size C, with the final chunk containing $L \bmod C$ elements. The probability of selecting a chunk with start position l decays to a lower limit p_{min}, with probability proportional to a polynomial in l. By default we decay the probability of selecting a chunk beginning at position l proportional to l^{-3} down to a minimum of $p_{\text{min}} = 0.1$, a hyperparameter choice we fixed early in model development and found no need to adjust.

Sequence length bias adjustment Chunk-based training introduces an issue in unconditional generative modelling: It biases the model towards chunks from shorter sequences. Consider the first chunk, that occurs at the very start of the sequence. For long sequences, this chunk will be selected relatively infrequently compared to short sequences. The model will therefore assign greater probability to initial chunks from short sequences, than initial chunks from long sequences.

We counter the bias by randomly filtering out sequences with a probability that is inversely proportional to the sequence length. We pick a maximum filtering sequence length L_{max} and filter our sequences with probability:

$$p_{\text{filter}}\left([\mathbf{t}_l]_{l=1}^{L}\right) = \text{maximum}\left(\frac{L}{L_{\text{max}}}, 1\right). \qquad (11)$$

D. Model scaling properties

Kaplan et al. (2020) show that for Transformer language models, performance improves reliably subject to constraints on model size, compute and dataset size. In particular, they show that if compute and data are not bottlenecks, then test-loss performance improves log-linearly with model size. Follow-up work Henighan et al. (2020) has shown that this phenomenon applies more generally beyond just language data. We investigate the extent to which DCTransformer scales as a function of model size by training models of varying sizes on LSUN bedrooms. Figure 9 shows the results broken down by channel, position and value prediction contributions, and chunk position. We find that the total bits-per-dimension (bpd) roughly matches the expected log-linear fit, with the exception of the smallest model, which performs worse than the expected trend. Another exception is chunk 0, where the performance improvement is less than expected for the largest model.

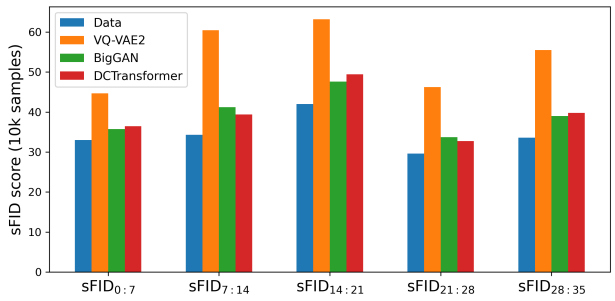

Figure 10. Comparison of sFID scores for different Inception `mixed_6/conv` feature channels using 10k samples. The x-axis label subscripts refer to the indices of the feature channels used. For example $\text{sFID}_{14:21}$ refers to the 14th to 21st channels of the `mixed_6/conv` Inception feature map. We find that although the absolute scores vary across feature channels, the relative performance of different model samples are preserved.

We also find that the value predictions account for the largest portion of the total bpd, followed by positions, and finally channels, which accounts for a very small portion of the total bpd. For the total bpd, the contribution per chunk decreases with the chunk position. This is likely because the total bpd is dominated by the value bpd, and we expect the value bpd to decrease as we transition from lightly quantized low frequency components, to more heavily compressed high frequency components.

E. sFID analysis

In Section 4 we introduced sFID, a variant of Frechet inception distance (FID, Heusel et al. (2017)) that uses features from the Inception network's `mixed_6/conv` layer, rather than the `pool_3` features used in standard FID. The `pool_3` features are preceded by a global average pooling stage in the Inception network, and are therefore highly invariant to the spatial distribution of input image features. This means that FID is unlikely to detect spatial mode-collapse: where model samples fail to reproduce the spatial variability of objects. The `mixed_6/conv` features used in sFID are taken from an intermediate feature map with a spatial resolution of 17×17, so that the degree of spatial invariance is likely to be substantially reduced.

In order to produce a relatively compact feature set, we use the first 7 channels of the `mixed_6/conv` feature maps, resulting in a total of $17 \times 17 \times 7$ features. In Figure 10 we show that the ranking produced by sFID scores is not sensitive to the particular feature channels used.

F. Additional samples

Figures 11 and 12 compare uncurated samples from DCTransformer and baselines to a random selection of real data on the FFHQ and LSUN datasets respectively. Figures 13 and 14 show uncurated upsampling and colorization results on ImageNet and OpenImagesV4 respectively.

Figure 11. Comparison between FFHQ images and uncurated model samples, all at 1024x1024 resolution.

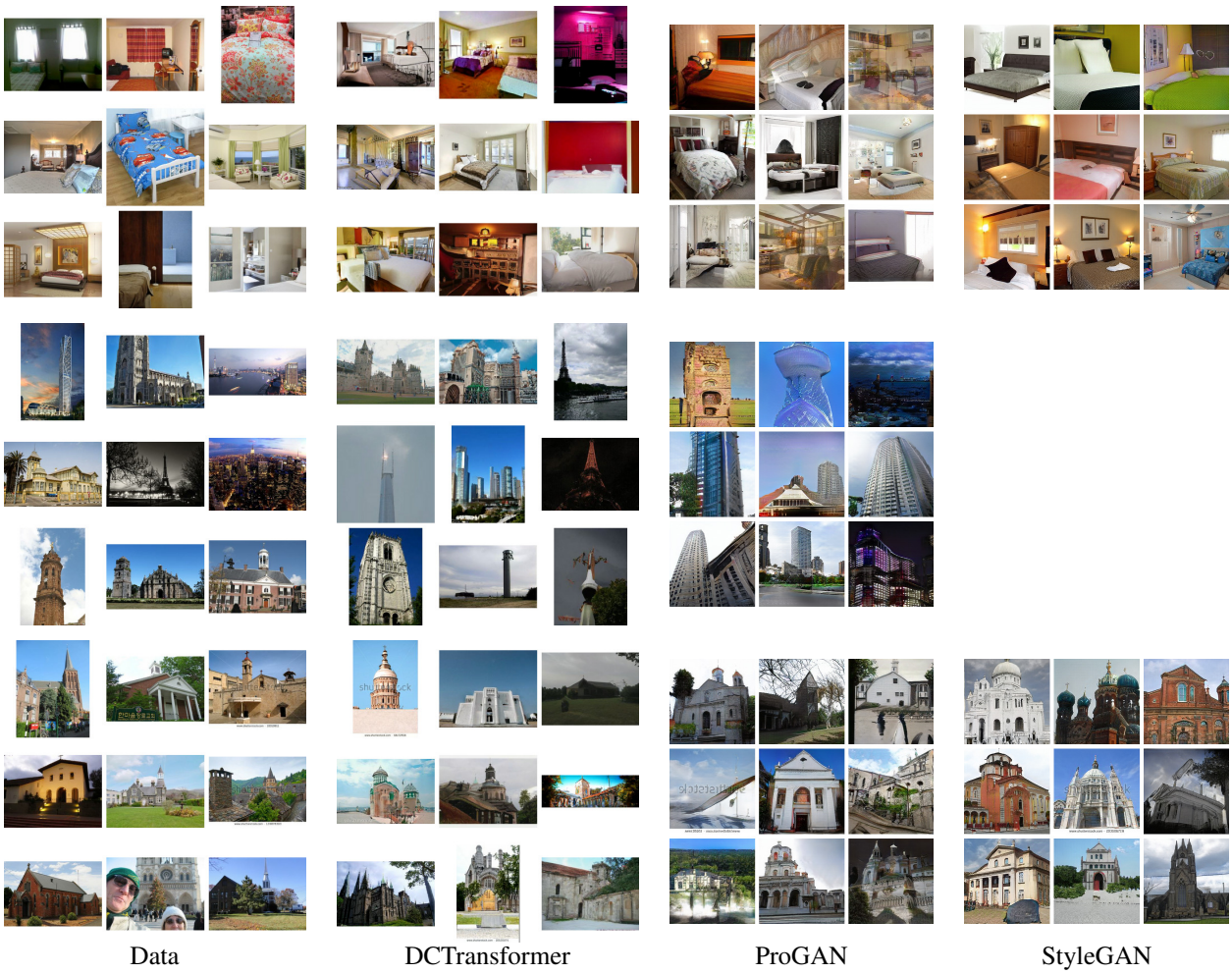

Figure 12. Comparison between LSUN images and uncurated model samples for bedroom,tower and church-outdoor subsets. StyleGAN refers to StyleGAN1 for bedrooms, and StyleGAN2 for church-outdoor samples. DCTransformer produces variable aspect ratio samples with long-side resolution 384. BigGAN and VQ-VAE aretrained and sample at a fixed 256x256 resolution corresponding to a resized long-side crop of the input images. ProGAN and StyleGAN samples use truncation 1.0 to yield maximum diversity.

	LSUN (all)	FFHQ	ImageNet	OpenImagesV4
Image resolution	384	1024	384	640
DCT block size	8	16	8	8
DCT quality	75	35	75	50
DCT clip value	1200	3200	1200	1200
Target chunk size	896	896	896	896
Target chunk overlap	128	128	128	128
Hidden units	896	896	1152	896
Self-attention heads	14	14	14	18
Layer spec (encoder)	[(1,2)] * 4	[(1, 2)] * 4	[(1,2)] * 4	[(1,2)] * 4
Layer spec (channel decoder)	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]
Layer spec (position decoder)	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]
Layer spec (value decoder)	[(1, 2)] * 5 + [(1, 7)]	[(1, 2)] * 6 + [(1, 7)]	[(1, 2)] * 6 + [(1, 7)]	[(1, 2)] * 6 + [(1, 7)]
DCT image downsampling kernel	kernel size 4, stride 2	kernel size 6, stride 3	kernel size 8, stride 4	kernel size 6, stride 3
Batch size	512	448	512	512
Dropout rate	0.1	0.1	0.01	0.01
Learning Rate Start	5e-4	5e-4	5e-4	5e-4
Tokens processed	300e9	250e9	1000e9	1000e9
Parameters	448e6	473e6	738e6	533e6
TPUv3 cores	64	64	128	64

	Plant Leaves	Retinopathy	CLEVR
Image resolution	2048	1024	480
DCT block size	32	16	8
DCT quality	50	75	90
DCT clip value	4000	3200	1200
Target chunk size	896	896	896
Target chunk overlap	128	128	128
Hidden units	768	768	896
Self-attention heads	12	12	14
Layer spec (encoder)	[(1,2)] * 4	[(1, 2)] * 4	[(1,2)] * 4
Layer spec (channel decoder)	[(1, 2)] * 3 + [(1, 3)]	[(1, 2)] * 3 + [(1, 3)]	[(1, 2)] * 3 + [(1, 4)]
Layer spec (position decoder)	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]	[(1, 2)] * 3 + [(1, 4)]
Layer spec (value decoder)	[(1, 2), (1, 2), (1, 3), (1, 3), (1, 7)]	[(1, 2), (1, 2), (1, 3), (1, 3), (1, 7)]	[(1, 2)] * 6 + [(1, 7)]
DCT image downsampling kernel	kernel size 4, stride 2	kernel size 4, stride 2	kernel size 6, stride 3
Batch size	256	512	128
Dropout rate	0.4	0.1	0.5
Learning Rate Start	5e-4	5e-4	5e-4
Tokens processed	100e9	200e9	50e9
Parameters	325e6	318e6	483e6
TPUv3 cores	64	64	32

Table 3. Model and training hyperparameters. The layer spec for the Transformer encoder and decoders is a list of tuples, where each tuple describes the number of self-attention layers, followed by the number of fully-connected layers in a Transformer block. For example [(1, 2)] * 3 + [(1, 4)] expands to [(1,2),(1,2),(1,2),(1,4)], and corresponds to four Transformer blocks, where the first three blocks consist of a single self-attention layer, followed by two fully-connected layers. The final block has one self-attention layer followed by four fully-connected layers.

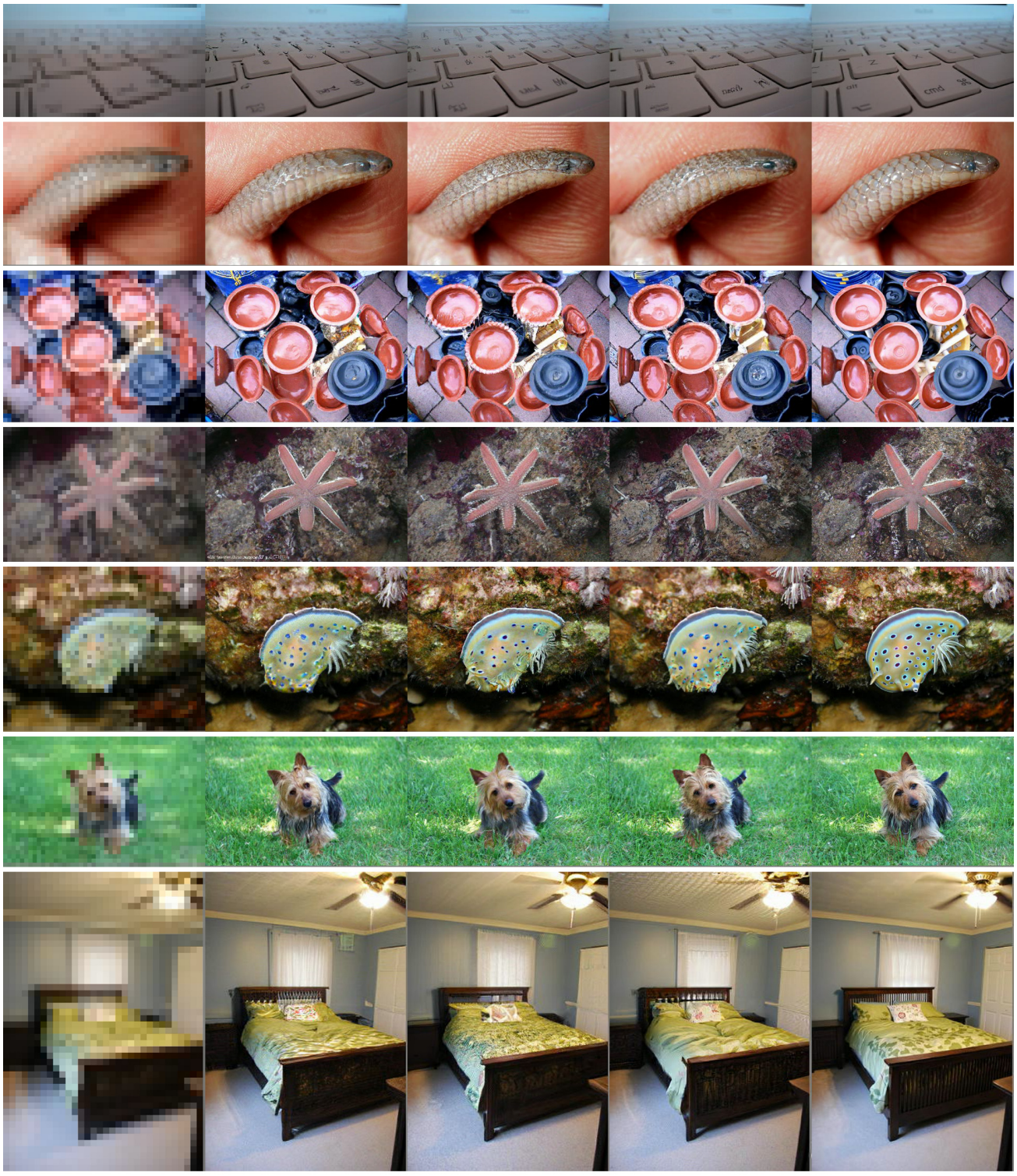

Figure 13. Uncurated 8x image upsampling results on ImageNet validation set. (left) input downsampled image, (middle) three samples generated by DCTransformer, (right) original image.

Figure 14. Uncurated image colorization results on OpenImagesV4 validation set. (left) input grayscale image, (middle) three samples generated by DCTransformer, (right) original image.