Appendix: Efficient Image Generation with Variadic Attention Heads

A. Appendix

In our Appendix we cover issues of reproducibility and focus on image analysis which requires visually inspecting figures. All images are embedded using vector graphics, to allow the reader to zoom in, but note that the source images are not vectored. To organize our Appendix we first include sections to make our work reproducible. Appendix B covers the model architecture, Appendix C includes important details about the parameters and throughput measurements, in Appendix D we have a small discussion on the limitations of metrics to help clarify the motivation for our visual analysis, which is coverend in Appendix E. The visual analysis includes a discussion of the attention maps (Appendix F), including maps at multiple resolutions for both StyleNAT and StyleSwin, and for both FFHQ (Appendices F.1.1 and F.1.2) and Church (Appendices F.2.1 and F.2.2) datasets.

B. Model Architecture

Level	Kernel Size	Dilation	Dilated Size
4	-	-	-
8	7	1	7
16	7	2	14
32	7	4	28
64	7	8	56
128	7	16	112
256	7	32	224
512	7	64	448
1024	7	128	896

Table A1. StyleNAT 2-Partition Model Architecture. First level uses Multi-headed Self Attention and not DiNA. This model is used for all FFHQ results, at all resolutions.

Figure 4 includes a depiction of the first 2 resolution levels of the StyleNAT architecture. At each resolution we use $N = 2$ transformer blocks, but this is configurable for scale For all experiments the layers have the same attention parameters, but there is nothing preventing one from making these distinct. Similarly, we use a constant kernel (window), k, size of 7. In all experiments we use a combination of no dilation (1) to create local dense windowing, and larger but more sparse receptive fields (dilation > 1). There are no restrictions in the architecture that require this, but we felt that these choices would be most clear in demonstrating the effectiveness of our work, due to the closeness to StyleSwin's kernel size of 8.

Level	Kernel Size	Dilations
4	-	-
8	7	1
16	7	1,2
32	7	1,2,4
64	7	1,2,4,8
128	7	1,2,4,8,16
256	7	1,2,4,8,16,32
512	7	1,2,4,8,16,32,64
1024	7	1,2,4,8,16,32,64,128

Table A2. Example of progressive dilation with 8 heads minimum, referred to "pyramid dilation."

We include Table A1 and Table A2 which show the parameters for the "split head" experiments and the progressive, or "pyramid dilation", used in the ablation study for LSUN Church experiments (Table 2). For "pyramid dilation" we always use a growth based on powers of 2, but this is not a requirement of the architecture.

For our Church experiments we found a more progressive dilation solution to work better. We provide an example of our "pyramid" style dilation, wherein we continually grow progression mixing several levels of dilation.

B.1. Hydra-NA Code

We include some PyTorch code for implementing Hydra-NA in Figure A1. Note that this code is not optimized and that all the natten operations could be performed in parallel for increased performance.

B.2. Other Configurations

The configurations are truly variadic, and allows for a large number of possible combinations. The primary reason for our parameter choices was to minimize the compounding factors in our experiments and minimize the influence of our parameter choices to the results, compared to our nearest competitor, and while fitting within our computational budget.

```python
class HydraNeighborhoodAttention(nn.Module):
    def __init__(self, dim : int, kernel_sizes : list[int], num_heads : int,
                 qkv_bias : bool=True, qk_scale : Optional[float]=None, attn_drop : float=0.,
                 proj_drop : float=0., dilations : list[int]=[1]) -> None:
        super().__init__()
        self.num_splits, self.num_heads = len(kernel_sizes), num_heads
        self.kernel_sizes, self.dilations = kernel_sizes, dilations
        self.head_dim = dim // self.num_heads
        self.scale = qk_scale or self.head_dim ** -0.5
        self.window_size = []
        for i in range(len(dilations)):
            self.window_size.append(self.kernel_sizes[i] * self.dilations[i])
        self.qkv = nn.Linear(dim, dim * 3, bias=qkv_bias)
        if num_heads % len(kernel_sizes) == 0:
            self.rpb = nn.ParameterList([nn.Parameter(
                    torch.zeros(num_heads//self.num_splits, (2*k-1), (2*k-1)))
                for k in kernel_sizes])
            self.clean_partition = True
        else:
            diff = num_heads - self.num_splits * (num_heads // self.num_splits)
            rpb = [nn.Parameter(torch.zeros(
                    num_heads//self.num_splits, (2*k-1), (2*k-1)))
                for k in kernel_sizes[:-diff]]
            for k in kernel_sizes[-diff:]:
                rpb.append(nn.Parameter(torch.zeros(
                    num_heads//self.num_splits + 1, (2*k-1), (2*k-1))
                ))
            self.rpb = nn.ParameterList(rpb)
            self.clean_partition = False
            self.shapes = [r.shape[0] for r in rpb]
        [trunc_normal_(rpb, std=0.02, mean=0.0, a=-2., b=2.) for rpb in self.rpb]
        self.attn_drop = nn.Dropout(attn_drop)
        self.proj = nn.Linear(dim, dim)
        self.proj_drop = nn.Dropout(proj_drop)

    def forward(self, x:torch.Tensor) -> torch.Tensor:
        B, H, W, C = x.shape
        qkv = self.qkv(x)
        qkv = qkv.reshape(B, H, W, 3, self.num_heads, self.head_dim)
        q,k, v = qkv.permute(3, 0, 4, 1, 2, 5).chunk(3,dim=0)
        q = q.squeeze(0) * self.scale
        k,v = k.squeeze(0), v.squeeze(0)
        if self.clean_partition:
            q = q.chunk(self.num_splits, dim=1)
            k = k.chunk(self.num_splits, dim=1)
            v = v.chunk(self.num_splits, dim=1)
        else:
            i, _q, _k, _v = 0, [], [], []
            for h in self.shapes:
                _q.append(q[:, i:i+h, :, :])
                _k.append(k[:, i:i+h, :, :])
                _v.append(v[:, i:i+h, :, :])
                i = i+h
            q, k, v = _q, _k, _v
        attention = [natten2dqkrpb(_q, _k, _rpb, _kernel_size, _dilation)
                        for _q, _k, _rpb, _kernel_size, _dilation in \
                        zip(q, k, self.rpb, self.kernel_sizes, self.dilations)]
        attention = [self.attn_drop(a.softmax(dim=-1)) for a in attention]
        x = [natten2dav(_attn, _v, _k, _d) for _attn, _v, _k, _d
                in zip(attention, v, self.kernel_sizes, self.dilations)]
        x = torch.cat(x, dim=1).permute(0, 2, 3, 1, 4).reshape(B, H, W, C)
        return self.proj_drop(self.proj(x))
```

Figure A1. Full code for StyleNAT's Hydra-NA module. Type hinting included for added clarity. *Requires NATTEN package*. Using NATTEN v0.14.6, subsequent versions may need modifications. Code is unoptimized, intended for research and clarity.

During our exploratory phase, we performed a few short experiments where we replaced the Swin Attention in StyleSwin with Hydra-NA. In general we found increased performance under the conditions that we did not use kernels sized 3 and that we did not use a very large and very small kernel when replacing the last layer. For example, using a split head design with one kernel sized 3 and the other kernel sized 45, both with dilation 1, showed worse performance when compared to StyleSwin. But we did find that there was increased performance if instead we used kernels size 45 for both partitions. The trade-off is that the larger kernels requires significantly more GPU memory and computation time. For example, while StyleSwin would utilize $\approx$ 36 GB of DRAM per GPU, when using a kernel size of 45 we used $\approx$ 76GB of DRAM per GPU and approximately 6x wall time (measured at a resolution of 1k iterations). Our final StyleNAT architecture uses slightly less DRAM than StyleSwin and $\approx$ 30s more per 1k iterations. Note that these DRAM requirements are not due exclusively to the model itself but these also include the EMA model, batch images, pre-fetches, and other such additional data that may be included when training. We show some results in Table A3, comparing to our training of StyleSwin and include comparisons with our final StyleNAT result. These results give evidence that the model can scale and that its performance may be proportional to the effective computation. These runs were never taken to convergence and thus we believe such experiments would be worthwhile, but are outside the compute budgets of our lab. We also performed similar replacement experiments with other resolution levels, but did not explore larger kernels. Without dilation we still found incremental improvements over StyleSwin.

kernel size	10k	25k	50k
StyleSwin	84.63	29.89	18.85
3/45	85.17	31.95	22.25
7/45	86.32	41.15	25.66
45/45	68.65	28.36	17.95
StyleNAT	145.57	20.10	10.81

Table A3. FID results, showing effects of replacing last resolution level's Swin Attention with a split head NA. First column shows the kernel sizes. No dilation was used. StyleSwin and StyleNAT included for comparison.

B.3. Potential Configurations

In this section we discuss the potential configurations for Hydra-NA. We believe this discussion will help others determine how to optimize this architecture and better understand the potential flexibility of the network. Our hyperparmeters were fixed in favor of ensuring higher interoperability of results given compute constraints. Optimization of the network will require further search, but we demonstrate that these are reasonable choices to start with. This discussion will also help practitioners optimize not just for the final fidelity of the images, but for computational constraints, noting that smaller kernels will increase inference speeds and reduce GPU memory.

Our kernels, k, can range from a size of 3 to the nearest odd integer smaller than the resolution, $\mathcal{R}$. This means that there will be $\frac{\mathcal{R}}{2} - 1$ potential kernels. We can also note that any kernel sized $\geq \frac{\mathcal{R}}{2}$ cannot have any dilation without exceeding the image size[1]. There are $\frac{\mathcal{R}}{4}$ such kernels. The max dilation size for a given kernel is $\left\lfloor \frac{\mathcal{R}}{k} \right\rfloor$. Thus the total number of configurations, per head, is

$$N_c = \sum_{i=1}^{\mathcal{R}/2-1} \left\lfloor \frac{\mathcal{R}}{2i+1} \right\rfloor \tag{1}$$

$$= \frac{\mathcal{R}}{4} + \sum_{i=1}^{\mathcal{R}/4-1} \left\lfloor \frac{\mathcal{R}}{2i+1} \right\rfloor \tag{2}$$

This, of course, is assuming that we are using square kernels, square images, and images with an even number of pixels. None of these are actual constraints to our formulation, so this could actually double[2].

For our FFHQ-256 configuration we have 16 heads for resolutions 4×4 to 64×64, 8 heads for resolution 128×128, and 4 heads for all other resolutions, and 2 transformers per resolution, this results in $2 \times (16 \times (4 + 14 + 37 + 97) + 8 \times 237 + 4 \times 565) = 13176$ potential configurations! ($\approx$ 47k for 1024×1024 resolution images) With 8 minimum heads in our LSUN Church experiments that allows for 17696 potential configurations. This is an extraordinary number of possible configurations for our architecture, with allows for potentially high rates of expressibility. In our experiments we manually selected parameters to ensure similarity to our best comparator *but these hyper-parameters can be learned.*

C. Parameters and Throughputs

We measured all throughputs via the official code bases and their respective sampling practices. Modifications to code were only done when necessary for equivalent evaluation and we do not believe these would result in meaningful differences. When checkpoints were provided we used those and followed the run formulas provided in a project's README file. Several models did not provide checkpoints, so we used the configurations most similar to those that

[1] It is technically possible to extend NA/DiNA to work for kernels larger than the image size but we will operate under this condition for practical purposes.

[2] For an image with an odd number of pixels there are $\left\lfloor \frac{\mathcal{R}}{2} \right\rfloor$ kernels, $\left\lceil \frac{\mathcal{R}}{4} \right\rceil$ kernels with no potential dilations

would be used during training. For parameter counts we used the open source tool *graftr*[3], to directly probe checkpoints.

Many StyleGAN models provide checkpoints as pickle files, which we loaded using the official "legacy._LegacyUnpickler" method and converted to PyTorch checkpoints. Parameters were based on the generator's Exponential Moving Average (EMA) models if provided, otherwise the generator. We did not count parameters included in the discriminator or elsewhere, focusing on what parameters are required for synthesis. For Unleashing Transformers [5] we include the total counts for the VQGAN (83.12M) and the Absorbing EMA's denoising function (76.84M). For LDM [52] we find 603M parameters within the state_dict, but this includes the EMA model. The "model" contains 274M parameters, matching Table 12 in their Appendix E.1, but we include all non-EMA parameters, believing these are still necessary for synthesis. We also note that LDM's training time and memory load can likely be significantly improved due to the implementation for loading the EMA model, which requires more DRAM than necessary.

For HiT GAN we were unable to gather throughputs and relied upon the paper's numbers. HiT GAN was written in tensorflow v1 and we were unable to find python wheels that would satisfy our system's requirements. In Table 4 of the HiT GAN paper [75], they report that on FFHQ-256 HiT-L has a throughput of 20.67 imgs/s and 97.64M parameters when run on a single NVIDIA V100 GPU. They similarly note that StyleGAN2 achieves 95.79 imgs/s using 30.03M parameters. The original StyleGAN2 paper [35] reports only reports results for FFHQ-1024 (Section 6), at 61 imgs/s. Their results were gathered on a NVIDIA DGX-1 with 8 V100 GPUs and during training. We note that their throughput measurements for StyleGAN2 are a bit over 10% higher than our measurements. We experienced similar issues when attempting to measure performance for GANformer and their GitHub page was ambiguous as to corresponding checkpoints. We did not reach out to the authors but similar questions were raised on their GitHub issues page, which appears to be inactive.

All measurements were performed on a single NVIDIA A100 GPU using Python 3.10.13, PyTorch 2.1.0, and CUDA 12.1 (installed with the official PyTorch instructions. The system CUDA version was 12.0). All measurements are normalized to batch size to make them independent of memory constraints. We utilize 50 rounds of warmup before sampling 100 rounds, which we average over. For diffusion models we used the highest batch size we could fit in memory since this yielded the best results. We took the highest throughput from multiple measurements, noting that variance was generally quite low.

[3] https://github.com/lmnt-com/graftr

For performance we enabled TF32 through torch's backends and utilized torch's inference_mode as opposed to no_grad. We used no other optimizations and made no attempt to use PyTorch's compile feature or NVIDIA's TensorRT, which should boost performance for all models. Several models have cuDNN enabled, and we left this at the default value.

We also note that variance in Style-based models can vary greatly due to their dependence on custom CUDA kernels written for the bias activation, conv2d gradient fix and resampling, fused-multiple-add (FMA), grid sampling gradient fix, and upfirdn2d[4] Karras *et al*. noted that these implementations could account for upwards of 40% improvement in throughputs. For example, ProjectedGAN [54] makes use of all of these as well as introduces some additional CUDA kernels[5]. Additionally, StyleGAN3 [36] (and consequently StyleGAN-XL [55]) introduced additional CUDA kernels. In Appendix D Karras *et al*. note that the speedup is $\approx 20 - 40\times$ when comparing against native PyTorch operations, with an overall training speedup of $\approx 10\times$ but do not specify if there are any differences from the StyleGAN2 implementations. On the other hand, GANformer, StyleSwin, and StyleNAT rely far less upon these implementations and thus likely results from suboptimal throughput performance. Note that StyleNAT also has a custom CUDA implementation for NA/DiNA, but not the head splitting parts of the architecture. Please refer to Figure A1 for more clarity.

For throughput measurements we use the procedure shown in Figure A2. This code is also available on our GitHub project.

D. Metrics and Image Quality

As the quality of our synthesized images increases we more quickly approach the limitations our our metrics. We note that if one compares the FID of the first 10k images of FFHQ-256 to the bottom 60k this yields a FID of 2.25. While this is not a completely fair comparison to how measurements are formed when comparing synthesised images to training images[6], it illustrates limitations in FID and the natural variance within the data itself. A true metric measuring the visual fidelity to humans still remains elusive [4, 39, 40, 46, 48, 49, 59, 77].

Additionally, to measure the preformance of models we aggregate over samples, and such results will not convey the difference between the quality of a single image and that of the average image. These make truly evaluating the quality of generative networks difficult and illustrates the need to further develop metrics and for evaluators to pay close attention to samples from the networks.

[4] https://github.com/NVlabs/stylegan2-ada-pytorch/torch_utils/ops
[5] https://github.com/autonomousvision/projected-gan/torch_utils/ops
[6] this is 10k vs 60k which does change results

```
@torch.inference_mode()
def calculate_throughput(args):
    torch.backends.cuda.matmul.allow_tf32 = True
    times = torch.empty(rounds)
    noise = torch.randn((batch_size, latent_dim),
                        device="cuda")
    for _ in range(warmup):
        imgs = generator(noise)
    for i in range(rounds):
        starter = torch.cuda.Event(
                    enable_timing=True)
        ender   = torch.cuda.Event(
                    enable_timing=True)
        starter.record()
        imgs = generator(noise)
        ender.record()
        torch.cuda.synchronize()
        times[i] = starter.elapsed_time(ender) \
                   / 1000
    imgsPerSecond = total_imgs / total_time \
                    / batch_size
    print(f"{torch.std_mean(imgsPerSecond)}")
```

Figure A2. Pseudo code for throughput measurements

A recent work by Stein *et al*. [59] performed on of the, if not the, largest human study to date, investigating the predictive power of various metrics when compared to a human's ability to distinguish synthesized images from real images. Their results include StyleNAT, but we note that the authors of this work have no affiliation with Stein *et al*. nor have contacted them in any way. Their results demonstrate that FID and other metrics are generally unreliable when predicting the human error rate for distinguishing synthesized images from real ones. We refer to their work for a full discussion of metrics and for an independent evaluation of differing metrics. We note that their model selection has a significant overlap with those in Table 2. Additionally, we note that StyleNAT is a outlier in their results, and its samples are the most difficult for humans to distinguish when compared to other models. While this result correlates with FID, they show that this is not true for many other models and datasets.

E. Visual Analysis and Artifacts

Given the limitations to metrics, as discussed in Appendix D, we investigate the visual quality of synthesized images and look for different biases that different networks exhibit. We encourage the reader to not completely rely on the examples within this work and to generate their own samples. For this section we embed the highest quality images that we can, using pdf embeddings, but note that the source images are in JPEG. This allows readers to zoom in if using an electronic PDF reader. Note that some artifacts may not be obvious at the standard resolution, but if one zooms into 300% or more then these artifacts will become much more apparent and most readers will still be able to identify them after returning to their normal reading settings. We believe that such investigations are essential for the evaluation of generative models and encourage the community to perform similar such studies.

E.1. StyleGAN3 vs StyleSwin vs StyleNAT

First we perform a comparison between StyleGAN3, StyleSwin, and StyleNAT on FFHQ-1024 samples. We use FFHQ because there is universal familiarity with faces and humans are biologically primed to recognize faces, meaning that we are more likely to notice subtle details where this may not be as true for images of other classes of objects. We believe that this makes it the best choice for identifying generation errors. We also note that the FIDs of these networks are 2.79, 5.07, and 4.17, respectively. If we rely exclusively on FID then we expect StyleGAN3 to be significantly better than StyleNAT and StyleSwin, and for StyleNAT to be moderately better than StyleSwin. This is difficult to truely measure and would require a costly human study, similar to Stein *et al*.'s [59]. Instead we try to focus on the *best possible* samples, and investigate the visual artifacts.

We remind the reader that common locations for visual artifacts can be found when looking at ears, eyes (including eyebrows, eyelashes, and pupils), the neck, and hair. In particular one can often quickly identify synthetic images by closely looking at eyes. We believe that all networks always produce artifacts that humans can easily identify given sufficient training, but leave the subjective conclusions to the reader. We note that we find a strong bias for high fidelity images to have simple backgrounds with a Bokeh effect. That is, where the image subject is sharp and in focus while the background is out of focus. This is achieved by using a high f-stop or large aperture size when taking a photo. We believe this is likely a dataset bias as this aesthetic is common for professional portraits. We will not focus on the background as when images like these are used for creating deep fakes it is not uncommon for these to be replaced. They are also a common source of errors, which the Bokeh style often makes harder to identify.

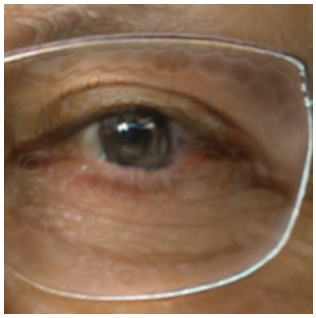
E.1.1. StyleGAN3

Karras *et al*. [36] has made public a set of curated images[7], which we searched through. We searched for the best quality image we could find and provide the link for independent analysis. The authors note that a significant part of their work was in removing aliasing from images. Aliasing being defined as overlapping frequency components that lead to distortion and other potential spatial artifacts.

[7] https://nvlabs-fi-cdn.nvidia.com/stylegan3

(a) 1024 FFHQ Sample from StyleGAN3

(b) Forehead bead pattern. Two bands at top and bottom third.



(c) Glasses with hexagonal artifacts around edges. Upper right of glasses.

Figure A3. Artifacts from StyleGAN3 FFHQ 1024 samples (sample 0068 from link). We show the banding effect that is common in StyleGAN3 photos, especially on foreheads, as well as hexagonal patterns that happen in glasses.

In Figure A3 we show our best found sample and in subfigures Figure A3b and Figure A3c we provide zoomed in sections where we identify frequency based artifacts. The most obvious artifact is the banding, which we show a zoomed in section in Figure A3b, but this appears throughout the face. Another horizontal band can be easily identified by the eye on the right side, near the glasses (under the disappearing Temple) as well as vertical banding across the neck. In Figure A3c we show a zoomed in section of the right glasses, where there are clear hexagonal patterns. Similar patterns can be found on the other side. By careful inspection of the eyes we can notice that the pupils are irregular and that neither the iris nor pupil is not circular, being more pronounced in the eye on the left side.

While these were not obvious at first glance to many some of our colleagues, all were able to see if they zoomed in, and would continue to see them after returning to their normal level of zoom. We were unable to identify a single image within the curated collection that did not exhibit similar artifacts. We were able to identify similar artifacts for all images we looked at, irrespective of the resolution or dataset, including non-human images. Thus we believe that this can serve as a reliable "fingerprint" for identifying StyleGAN3 based models.

E.1.2. StyleSwin

For StyleSwin [71] we generated 50 samples from their official checkpoint and picked the best sample, discarding any samples where there were obvious large scale artifacts (colloquially refereed to as "GAN monsters"). These samples can be generated from our codebase using the StyleSwin checkpoints. Our selection is show in Figure A4, where artifacts are more apparent than in StyleGAN3. We notice clear "block" like shapes throughout the face, as seen in Figure A4b. These are significantly different from those noted by Zhang *et al*. [71], which are more similar to pixelization. Interestingly we do not observe the same pixelization visible in the samples shown in Figures 3 and 5 of their paper, but ours look more similar to those shown in their header or in Figures 7, 10, 11, or 12.

In addition we notice more continuous patterns most easily seen by the ear, Figure A4c, but also observable in the chin and near the eyes (not to be confused with "crow's feet"). We also notice a large discrepancy between the eyes. The image exhibits clear heterochromia (the iris are different colors), as well as significantly different sizes. Upon close inspection, it can be seen that the reflection within the eyes would suggest the person is looking at two different scenes, with different lighting conditions. Additionally, we notice a high rate of speckling in faces, with the easiest to view one being the light yellow spot on the cheek. Such artifacts are less obvious and may be confused with common skin blemishes (e.g. sun spots).

These artifacts show a clear demonstration where the transformer does not provide long-range coherence. We also observe these patterns within the images shown in the paper, including the aforementioned figures. We provide further explination of these artifacts in Appendix F.

E.1.3. StyleNAT

For our image selection we follow the same procedure as with StyleSwin Appendix E.1.2. Our selection is shown in Figure A5, and other examples can be seen by zooming in on our header (Figure 1) or in Figure 5 for samples from the lower resolution network. Similar artifacts are visible within these images and some maybe more easily seen in those examples. For example, the subject's iris color is dark and may make investigation of the iris and pupils more dif-

(a) 1024 FFHQ Sample from StyleSwin

(b) Forehead squares

(c) Right ear texture

Figure A4. Artifacts from StyleSwin FFHQ 1024. (We generated these.)

(a) 1024 FFHQ Sample from StyleNAT

(b) Forehead lines

(c) Right eye spotting

Figure A5. Artifacts from StyleNAT FFHQ 1024. (We generated these.)

ficult for some. We find heterochromia and distorted pupils less common among our samples but observe that when it happens it is more likely to be Sectoral Heterochromia, where the iris has multiple colors (as is referenced by the popular YouTuber 3Blue1Brown). This can be seen in the central image of the header, in the eyes if the blond hair girl. Her eyes are predominantly blue, but show bits of brown, and there is an unrealistic color closer to cerulean blue in the bottom corners, similar to those of the fictional Fremen in Dune. We find full heterochromia is quite rare, but an instance can be observed in the header in the right most column of the FFHQ-256 samples. Several other images have slightly distorted pupils. Despite these artifacts, we believe that our samples perform better than others within these areas and demonstrates our model's ability to learn long range coherence.

We do identify other artifacts, which we believe can be used to visually fingerprint our model. We also observe a banding like pattern, but that these are smoother lines and may be easily confused with strands of hair or indentations of the skin. We show a zoomed in section of the forehead in Figure A5b. Similar artifacts are more apparent on the left eye and near the smile lines of the cheek (clearer on the right side). We also observe some spotting artifacts, that are predominantly blue in color. We illustrate this in Figure A5c, showing the right eye, but this is also visible between the eyebrows. Similar artifacts appear in the man's beard, but may be easily confused with gray hairs. We also notice this chromatic aberrations within the strands of hair in the forehead. This can also be seen a bit in Figure A5b but may require zooming in depending on the screen used for viewing. The speckling artifacts being similar to StyleSwin may be a result of the underlying architecture, but further investigation is necessary.

We believe that these artifacts can serve as means to visually fingerprint our model and distinguish it from others. Notably, are lines appear to be an artifact of the method, and we are able to view these within the attention maps Appendix F.

F. Attention Maps

To help explain the observations we see throughout this work, we visualized the attention maps for both StyleSwin and StyleNAT. We note that neither of these networks can have attention maps generated in the usual manner. Our code for this analysis will also be included in our GitHub. For both versions we can't extract the attention map from the forward network, as would be usually done, but instead extract both the query and key values. Exact methods are explained in the respective FFHQ sections. We believe that these maps demonstrate the inherent biases of the network and specifically demonstrate why StyleNAT, and critically the Hydra attention, result in superior performance. Swin's shifted windows demonstrate a clear pooling, which may be beneficial in classification tasks, but not as much for generative tasks, which are more sensitive and unstable. They also provide explanations upon where both networks may be improved within future works and we believe this tool will be valuable to other researchers in other domains.

We will look at both FFHQ and LSUN Church to try to determine the differences and biases of the networks and attention mechanisms. For all of these we will generate a random 50 samples and select by hand representative images for the give tasks. It is important to take care that there is a lot of subjectivity here and that these maps should only be used as guides into understanding our networks rather than explicit interpretations. Regardless, the attention maps are still a helpful tool in determining features and artifacts in generation, as we will see below. The patterns discussed were generally seen when looking at each of the sampled images during our curation.

Our attention maps suggest that these networks follow a fairly straight forward and logical method in building images. In general we see that lower resolutions focus on locating the region of the main objects within the scene while the higher resolutions have more focus on the details of the images. We see progressive generation of the images, that each resolution implicitly learns the final image in progressively detail. This suggests that the progressive training seen in StyleGAN-XL may also benefit both of these networks. The Style-based networks generally have two main feature layers (or blocks) per resolution level, which we similarly follow. Our maps also suggest that a logical generation method is performed at the resolution level. Where the first layer generating the structure, realigning the image after the previous up-sampling layer. The second layer generates more details at the resolution level. This may suggest that a simple means to increasing fidelity would be to make each resolution level deeper, which is also seen in StyleGAN-XL. In other words, fidelity directly correlates to the number of parameters, and thus it is necessary to incorporate that within our evaluation. The goals of this work is on architecture and the changes that they make, rather than overall fidelity. We leave that to larger labs with larger compute budgets.

To make reading easier we have placed all maps at the end of the document and provide detailed descriptions in each caption.

F.1. FFHQ

For FFHQ we will look at specifically the 1024 dataset and we will select our best sample. We are doing this to help determine the differences in artifacts that we saw in Appendix B. Since many of these features are fine points we will want to see the high resolution attention maps to understand what the transformers are concentrating on and how the finer details are generated. Specifically, we use the same images that were used within the previous section to help us identify the specific issues we discussed.

F.1.1. (FFHQ) StyleSwin

For StyleSwin we extract the query and key values from each forward layer (note that there are two attentions per resolution level for StyleGAN based networks). We perform this for each split window which has shape $[B\frac{H}{w_s}\frac{W}{w_s}, n_h, w_s^2, C']$, where B is the batch, W, H are the height and width, w_s is the window size, n_h is the number of heads, and C' is the number of channels. We concatenate along the split heads and then reverse the windowing operation by re-associating the windows with the height and width. Once this is done we can mean the pixel dimensions for the query and flatten them for the key (q is unsqueezed for proper shaping). We then can obtain a normal attention map where we have an image of dimensionality B, n_h, H, W.

We will look at this attention map for the same sample as in Figure A4. We break these into multiple figures so that they fit properly with Figure A13 representing the 1024×1024 resolution, Figure A14 the 512×512, Figure A15 representing both the 256×256 and 128×128, Figure A16 the 64×64 and 32×32, and finally Figure A17 representing the 16×16 and 8×8 resolutions. Note that the second half of the heads represents a shifted window, per the design specified in their paper. In these feature maps we see consistent blocking happening, which is indicative of the issues with the Swin Transformer [41]. This also confirms the artifacts and texture issues we saw in the previous section. These artifacts can even be traced down to the 64 resolution level, Figure A16. We believe that this is a particularly difficult resolution for this network as it has more blocking in the second layer than others.

We also notice that in the earlier feature maps that StyleSwin has difficulties in picking up long range features, such as ears and eyes. This likely confirms the authors' observations of frequent heterochromism (common in GANs), mismatched pupil sizes, and differing ear shapes.

At higher resolutions (128 and above) we find that the network struggles with texture along the face despite establishing the general features. This is seen by half the heads being dark and the other half being bright, as is seen in Figures A14 and A15. This does suggest some underperformance from the network, with one set of heads doing significantly more work when compared to the others. We also see higher blocking, especially in the first layer, at lower resolutions, indicating difficulties in acquiring the general scene structure. This warrants more flexibility, such as that offered by Hydra.

F.1.2. (FFHQ) StyleNAT

For StyleNAT we perform a similar operation as to StyleSwin. We similarly extract the queries and keys, mean over the query's pixel dimensions (unsqueezing), and flattening the key's pixel dimensions. We similarly get back an image of shape $[B, n_h, H, W]$, with similar dimension definitions. We break these into multiple figures so that they fit properly with Figure A8 representing the 1024×1024 resolution, Figure A9 the 512×512, Figure A10 representing both the 256×256 and 128×128, Figure A11 the 64×64 and 32×32, and finally Figure A12 representing the 16×16 and 8×8 resolutions. The first half of the heads has no dilation and the second half has dilations corresponding with the architecture specified in Table A1.

We see that in the high resolution images that NA is learning textures and long range features across the face. This supports the claim that transformer mechanisms are adequately learning these long range features, as should be expected, and would support more facial symmetry that would be seen in human faces.

In the first layer we notice that more local features are being learned, which explains the better textures seen in our samples. Particularly we notice in Figures A8 and A9 that the first 2 heads learn feature maps on the main part of the face. Noting that the first two heads represent 7×7 kernels that are not dilated. We also noticed some swirling patterns in the second half of heads, which correspond to dilated neighborhood attention mechanisms. These correspond to the soft lines we saw above, and act like edge detectors. The second layer does a better job at finer detail and removes many of these, tough they are still visible on the chin. We notice that these particularly appear around hair and may be reasoning that the hair quality of our samples perform well. The long range, dilated, features all do tend to learn long range features and aspects like backgrounds, as we would expect.

In the lower resolutions we see that these attention maps learn more basic features such as noses, ears, and eyes, which helps resolve many of the issues faced by CNN based GANs. Details such as eyes and mouth can be identified even at the 16 resolution image Figure A12! The authors noticed that while generating they observed lower rate of heterochromia (different eye colors), which are common mistakes of GANs. This is difficult to quantify as it would unlikely be caught by metrics such as FID but we can see from the attention maps that the early focus on eyes suggests that this observation may not be purely speculation.

We believe that these attention maps demonstrate a strong case for StyleNAT and more specifically our Hydra Neighborhood Attention. That small kernels can perform well on localized features, like CNNs, but that our long range kernels can incorporate long range features that we'd want from transformers. We can also see from the feature maps that the mixture of heads does support our desire for added flexibility. This is done in a way that is still efficient computationally, having high throughputs, training speed, and a low requirement on memory.

F.2. Church Attention Maps

To help us understand the differences in performances specifically in the LSUN Church dataset we also wish to look at the attention maps to help give us some clues. We know that FID has limitations being that Inception V3 is trained on ImageNet-1k and uses a CNN based architecture. ImageNet-1k is primarily composed of biological figures and so does not have many objects that have hard corners like LSUN Church. Additionally, CNNs have a biased towards texture [15, 23], which can potentially make the metric less meaningful, especially on datasets like this. Since we had noticed that the Swin FFHQ attention maps had a bias to create blocky shapes and StyleNAT had a bias to create rounder shapes, we may wish to look into more detail to determine if these are biases of the architecture or that of the dataset. We find that this is true for StyleSwin but not of StyleNAT.

To understand why this dataset provides larger difficulties for these networks we not only select a good sample, but also a bad sample, hoping to find where the model loses coherence. We find that in general this happens fairly early on, with the networks having difficulties placing the "subjects" within the scene. We see higher fidelity maps in the better samples but find that overall these struggle far more than on the FFHQ task.

We believe that our results here show that FID is not reliable for the LSUN Church dataset, as well as demonstrates that Church is a significantly harder generation problem for these models than FFHQ is. This is claim is consistent with many of the aforementioned works, which present stronger cases and make similarly arguments for other metrics. These demonstrate the need to perform visual analysis

as well as feature analysis to ensure that the model is properly aligned with the goal of high quality synthesis rather than with the biases of our metrics. Evaluation unfortunately remains a difficult task, where great detail and care is warranted.

Specifically, at low resolutions both networks have difficulties in capturing the general concept of the scene. We believe that this is due to the increased variance and diversity of this dataset, compared to FFHQ. While human centered faces share a lot of general features, such as a large oval centered in the image, this generalization is not true for the Church dataset, which a wide variety of differing building shapes, many different background objects to include (which we say FFHQ prefers simple backgrounds), and that the images are taken from many different distances.

StyleSwin samples are shown in Figure A6 and the StyleNAT samples are show in Figure A7. We believe that both these samples look on par with the quality of that of ProjectedGAN [54], which currently maintains SOTA on LSUN Church with an FID of 1.59, and thus are sufficiently "good" samples. We will look at the blocks sized 32 to 256, as we believe this is sufficient to help us understand the problems, but we could generate smaller maps.

It is unclear at this point if the fidelity could be increased simply by increasing the number of training samples or if additional architecture changes need to be made in order to resolve this (as suggested above). We will specifically note that even SOTA generation on this dataset, ProjectedGAN [54], does not produce convincing fakes, while this task has been possible on FFHQ for some time, albeit not consistently. This is extra interesting considering that the SOTA FID on LSUN Church is 1.59, with 3 networks being below a 2.0 while SOTA FFHQ-256 (the same size) is 2.05 (this work) and scores as high as 3.8 [34] frequently produce convincing fakes. We also remind the reader that while ProjectedGAN performs well on Church (1.59), it does not do so on FFHQ (3.46), Table 3.

F.2.1. (Church) StyleSwin

For the StyleSwin generated images, Figure A6, we can see that the good image looks nearly like a Shutterstock image, almost reproducing a mirrored image and where the text is almost legible. But in this we also see large artifacts, like the floating telephone poll, the tree coming out of a small shed, or other distortions. We believe that this telephone poll may actually be part of a watermark, but are unsure. In the bad image, we see that there was a mode failure and specifically that the generation lost track of the global landscape. Even with this failure, we still do see church like structures, such as a large distorted window, making this image more of a surrealist interpretation of a church than a photograph. These images will thus provide good representations for understanding these two modes, of why good church images are still hard to generate and why they completely fail.

In short, we find that at all levels, there is higher blockiness and less detail captured by the attention mechanisms when compared to FFHQ. We see either very high or very low activations with the inability to focus on singular tasks. At low resolutions we see difficulties capturing structure and that this error propagates through the model.

Figure A22 shows our full resolution images, and in the attention maps we can again see the same blocky/pixelated structures that we found in the FFHQ investigation, but at a higher rate. For the good images, in the first layer we see that the first head is performing an outline detection on the scene, almost like a Sobel filter. We also see that the last head is delineating the boundary between the foreground and background, particularly the sky. Interestingly this appears almost like Pointillism, which we see in many following maps as well. This is likely bias from the shifted windows. In the second layer we see more fine grained structure, but interestingly we do not see as much as we saw in the FFHQ version at the same resolution, Figure A15, which suggests that this is a more difficult task for this network. We can also see that this level is concentrating on the text at the bottom of the image, which is not as clearly visible in the smaller resolution maps. Notably, we do not clearly see the floating telephone pole or the wires in the sky. These could be formed from another part of the network, such as the RGB or MLP layers, but we have not investigated this. Further investigation is needed to understand the contributions of these layers.

Moving down to the 128 resolution images in Figure A23 we see that the attention maps overall get much messier. For the good image we can see that the roof of the church is picked up by many of the heads. In the second layer, on the second head, we also see a clear filter looking at the tree and roof of the church, which we can also see a less clear selection in head 6 and the first layer at head 5. These same heads provide decent filters for the bad images, the layer 1 head 5 and layer 2 head 2 seeming to do the best at object filtering.

Looking at the 64 resolution Figure A24 and 32 resolutions Figure A21 we can more clearly see where the problems are happening. In FFHQ the 64 resolution maps, Figure A16, is where we start to first see our main object with relative details and the 32 resolution has a decent depiction of broad shape. We do not have as good of an indication within these maps, where the 64 resolution images do not have clearly identifiable building textures, let alone building shapes. This is even worse at the 32 resolution image level.

Interestingly, at this resolution it is difficult to distinguish which version would generate the good or bad image, which doesn't seem distinguishable till at least the 128 resolution. These aspects suggest that the generation of this data is substantially harder for this model. This is extra interest-

ing considering that the FID scores are fairly close for both of these datasets, with FFHQ being 2.81 and Church being 2.95. With more difficulties in capturing general structure the network then struggles to increase detail and this systematic issue cannot be resolved. This network saw trained on 1.5M iterations.

F.2.2. (Church) StyleNAT

StyleNAT performs significantly worse at LSUN Church, and it isn't clear why this is. Determining if this is a limitations of the metric, which may be biased to these features [40], we must explore a bit deeper. For these attention maps we will use the model that generated visible text and what the authors thought were higher quality. This network uses smaller kernel sizes of 3 and has a max dilation rate of 8. Thus the dilations are [[1],[1][1,2],[1,2,4],[1,2,4],[1,2,4],[1,2,4,8]]. This is the same configuration as when higher overfitting was observed. Since higher overfitting tends to correspond to higher fidelity we want to investigate what went right, to improve the work. The good image here is on par with that of Swin, and SOTA works, but the bad image is again a surrealist work wherein we see a agglomeration of a "Church." The good image appears pixelated, has scan lines, and some other distortions such as the car being reflected and turned into a bush. The bad image seems to incorporate nearly every feature within the dataset, including churches, towers, temples, cathedrals, as well as many different trees all smashed together Cronenberg style.

In short, we find that StyleNAT is in fact able to generate hard lines, as this dataset is biased towards, but does tend to prefer smoother features. We also see that at even the early stages that the scene has difficulties capturing global coherence. This likely explains the instabilities we faced and why training often diverged fairly early on, with nearly a fifth of the number of iterations as FFHQ and nearly 10% of StyleSwin.

The 256 resolution attention maps, Figure A18, images we immediately see that some of the attention heads to not have rounder features, indicating that our network does not have a significant bias towards biological shapes. In the first level, the first three attention heads have what appear to be scan lines, which we do see manifest in the full image. We also see traces of this in the next three heads, as well as most of the heads in the second layer. It appears that in this case, this level is looking a lot at texture, similar to that in FFHQ Figure A10. An interesting feature here, clearer in the first layer in heads 4-8, is that the we see what looks like the skeleton of a tree with branches coming out, almost centered at where the actual tree is in the main picture (left). What is notable here is that neither the trunk nor the branches are visible in the generated image, and that the "imagined" trunk is a bit translated from where we may predict it would be on the "actual" tree.

In other maps that we generated, that aren't shown, we noticed this pattern is extremely frequent when trees exist in the scene and there exists identical structure when the tree does not have foliage. This includes the circular shapes adjacent to the trunks. We did not notice this feature when only the foliage is visible, where the tree may look more like a bush, such as in the bad sample. We did not notice such skeletons as prevalent in the Swin version, although the best example can be seen in head 4 of the 256 layer in Figure A22, but this appeared to be an exception rather than the norm. We are careful to make a conclusion that the network has classified trees and understands their skeletal structure and note that a reasonable alternative explanation is that this trunk looking figure can just as easily be a guide for distinguishing the location of the tree.

There is also a notable difference in the attention maps between levels. In general we believe these show that the first layer is working more on the general structure of the scene while the second layer is improving detail. We also saw such correlations within the FFHQ analysis. We believe that this is a reasonable guess because the first level follows an upsampling layer and thus the network needs to first re-establish structure of the scene before it can provide detail. We also believe that this happens within the Swin based generator as well. This can mean that potentially higher fidelity generators can add additional layers, and that this is more necessary at higher resolutions.

As for the reasons for the low quality generations, we notice that the scenes in the 32 and 64 resolution, Figures A20 and A21, levels have potentially suggestive attention maps. Particularly we notice that the bad quality image has much more chaotic attention maps. Interestingly, we also see the scan lines

(a) Good (b) Bad

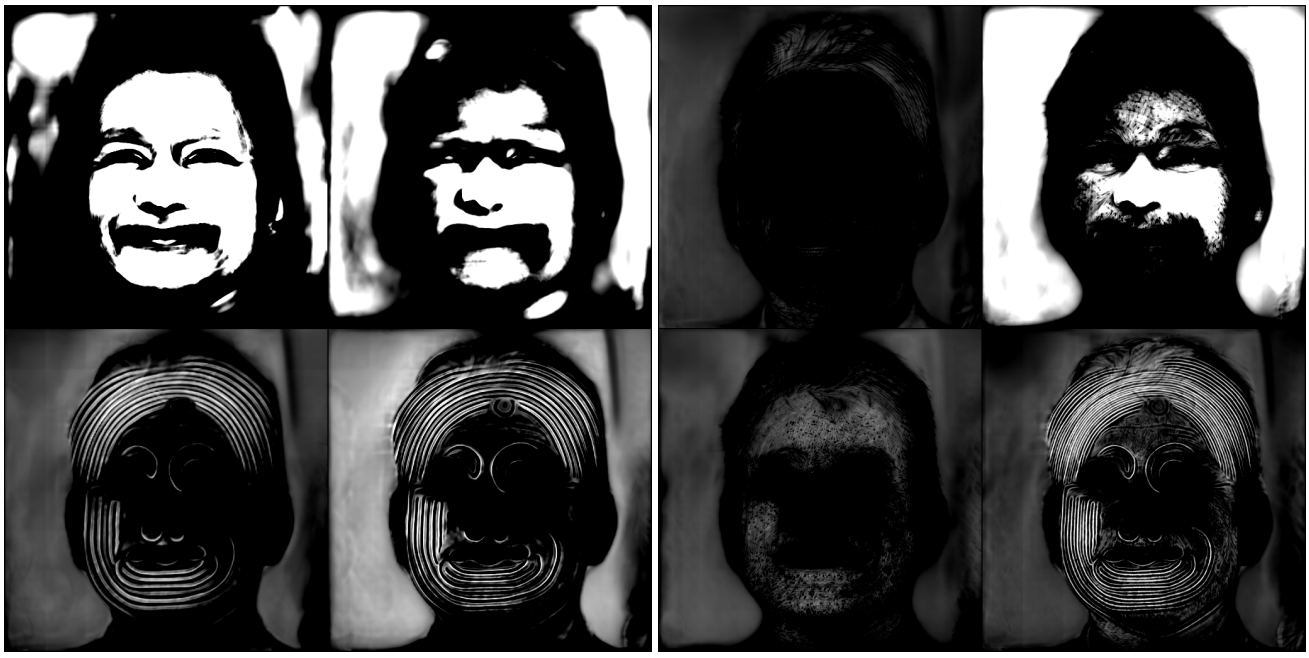
Figure A6. Church good and bad samples from StyleSwin. Good example has a clearly visible church and tree with a good distinction of foreground and background. Good example has a fairly legible citation but no other watermarks. Bad example has lost global structure but does maintain church like features.

(a) Good (b) Bad

Figure A7. Church good and bad samples from StyleNAT. Good sample has a clearly visible church, with cars out front and a clear distinction between foreground and background. While the bad sample distinguishes foreground and background, it is unable to correctly connect a coherent image of a church.

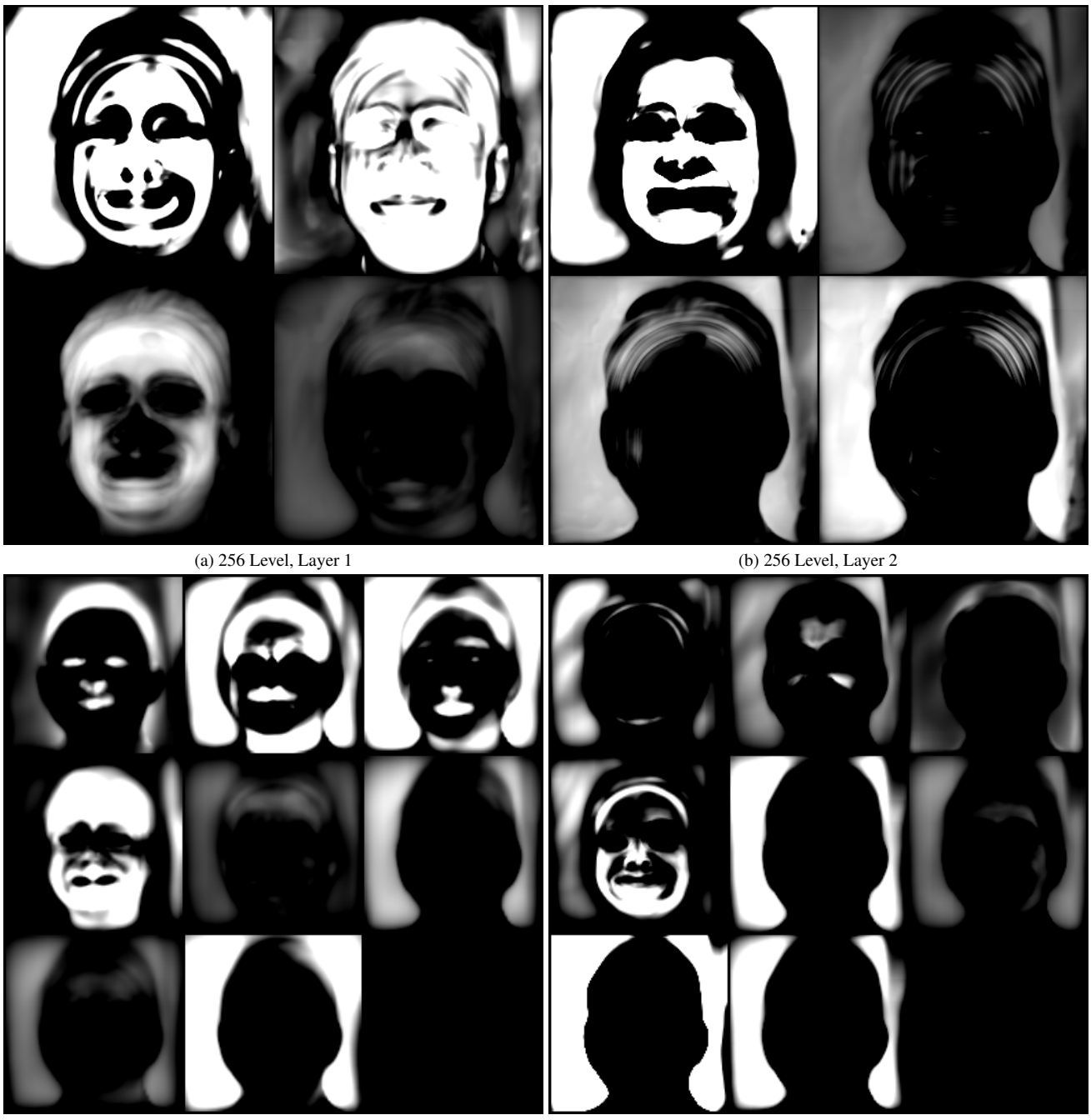
(a) 1024 Level, Layer 1 (b) 1024 Level, Layer 2

Figure A8. FFHQ StyleNAT attention maps at the level with 1024 resolution. Every layer has 4 heads with kernel size of 7, but the last 2 heads have a dilation of 128. First layer concentrates more on structure and second more on texture. Heads without dilations appear to focus more on texture and the face.



(a) 512 Level, Layer 1 (b) 512 Level, Layer 2

Figure A9. FFHQ StyleNAT attention maps at the level with 512 resolution. Every layer has 4 heads with kernel size of 7, but the last 2 heads have a dilation of 64. First layer appears to focus more on structure, with no dilations concentrating on the face. Dilated heads focus on head shape.



(a) 256 Level, Layer 1

(b) 256 Level, Layer 2

(c) 128 Level, Layer 1

(d) 128 Level, Layer 2

Figure A10. FFHQ StyleNAT attention maps at levels with 128 and 256 resolution. Every layer in each 32×32 level has 16 heads, and every layer in each 64×64 level has 8 heads, all with kernel size of 7. The second half of the heads have dilations 16 and 32 respectively. General structure is visible and it can be seen we capture long range features.

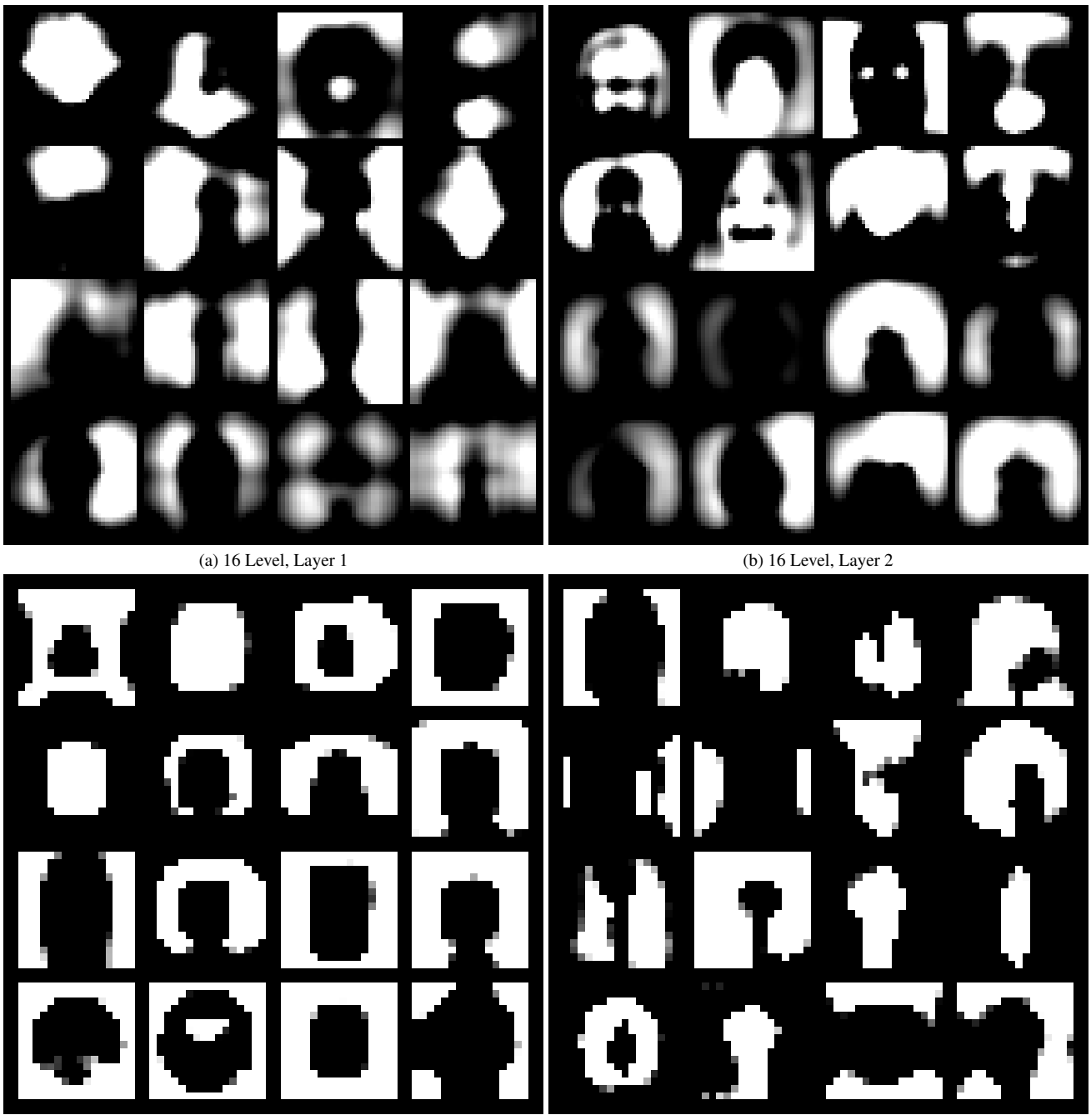
(a) 64 Level, Layer 1

(b) 64 Level, Layer 2

(c) 32 Level, Layer 1

(d) 32 Level, Layer 2

Figure A11. FFHQ StyleNAT attention maps at levels with 32 and 64 resolution. Every layer in each level has 16 heads with kernel size of 7. The second half of the heads have dilations 4 and 8, respectively. Main structure visible in these resolutions, including eyes and the separation of face and hair.



(a) 16 Level, Layer 1

(b) 16 Level, Layer 2

(c) 8 Level, Layer 1

(d) 8 Level, Layer 2

Figure A12. FFHQ StyleNAT attention maps at levels with 8 and 16 resolution. Every layer in each level has 16 heads with kernel size of 7. The second half of the heads have dilations 1 and 2, respectively. The 8 resolution image looks to be focusing on placement of object within the scene, taking the general round shape and distinguishing subject from background.

(a) 1024 Level, Layer 1 (b) 1024 Level, Layer 2

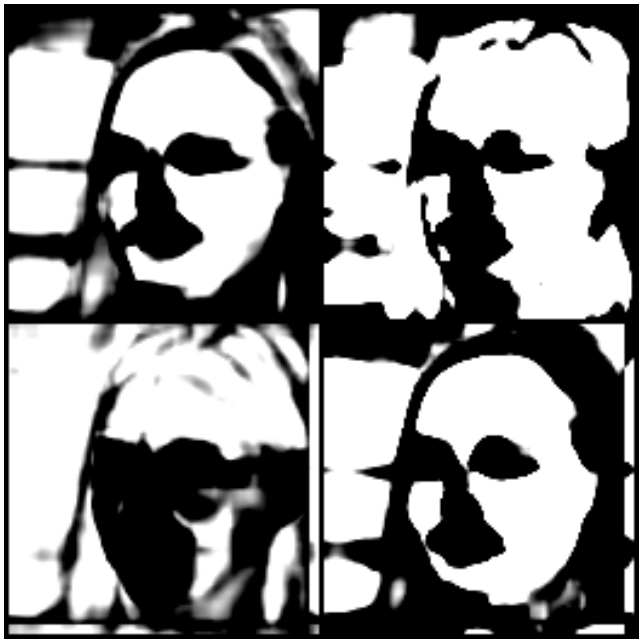
Figure A13. FFHQ StyleSwin attention maps at the level with 1024 resolution. Each layer has 4 heads with kernel size of 8 but half of them were trained with Shifted WSA. First level appears to concentrate on facial structure and texture. Second level appears to focus on symmetric features such as cheeks and eyes.

(a) 512 Level, Layer 1 (b) 512 Level, Layer 2

Figure A14. FFHQ StyleSwin attention maps at the level with 512 resolution. Each layer has 4 heads with kernel size of 8 but half of them were trained with Shifted WSA. Generative artifacts are clearly visible on forehead in most maps. Heads have vastly different concentration levels.

(a) 256 Level, Layer 1

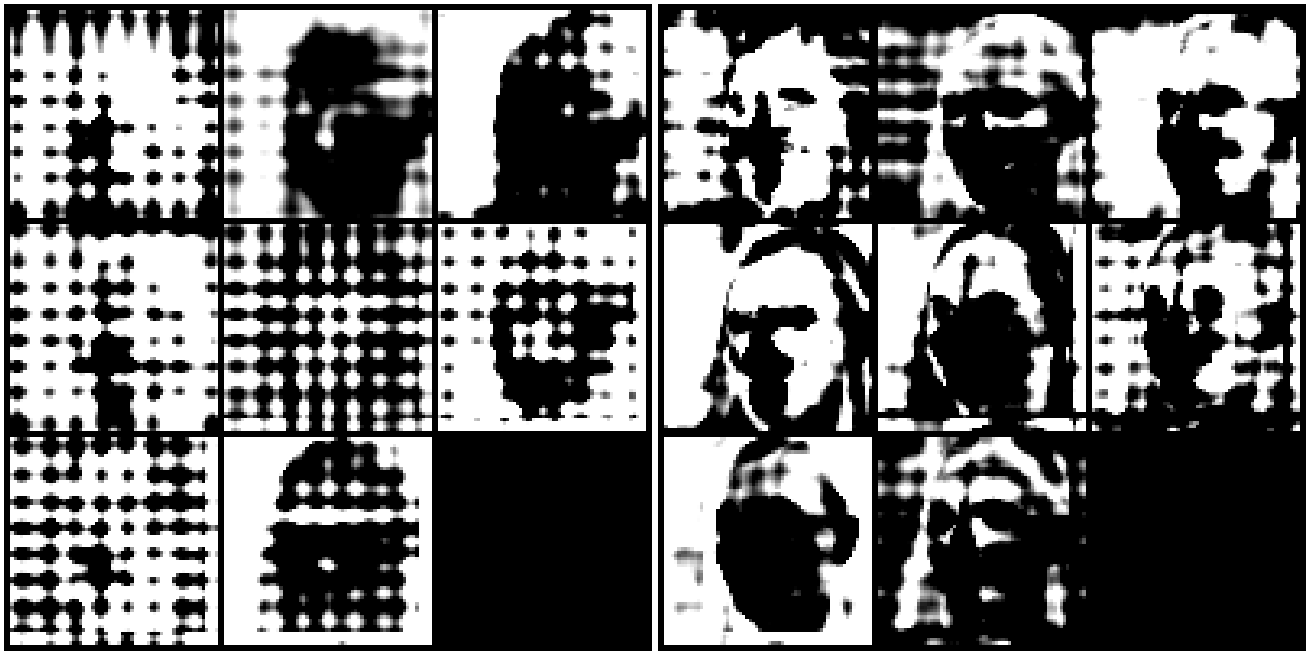
(b) 256 Level, Layer 2



(c) 128 Level, Layer 1

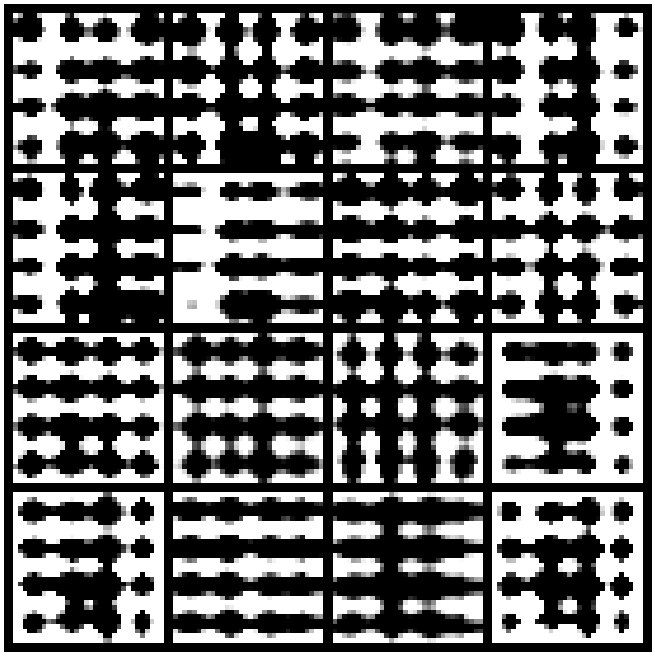
(d) 128 Level, Layer 2

Figure A15. FFHQ StyleSwin attention maps at levels with 128 and 256 resolution. Every layer in each level has 4 heads with kernel size of 8 but half of them were trained with Shifted WSA. 128 resolution shows beginning indications of generative artifact.

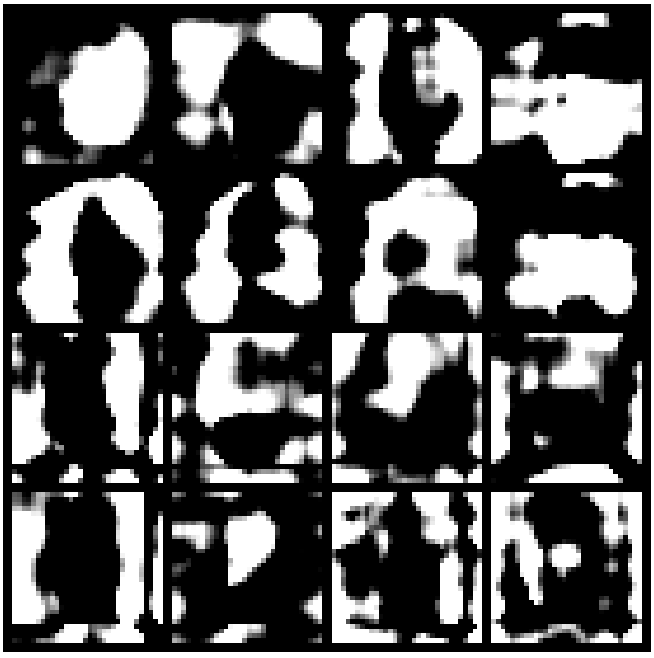


(a) 64 Level, Layer 1

(b) 64 Level, Layer 2

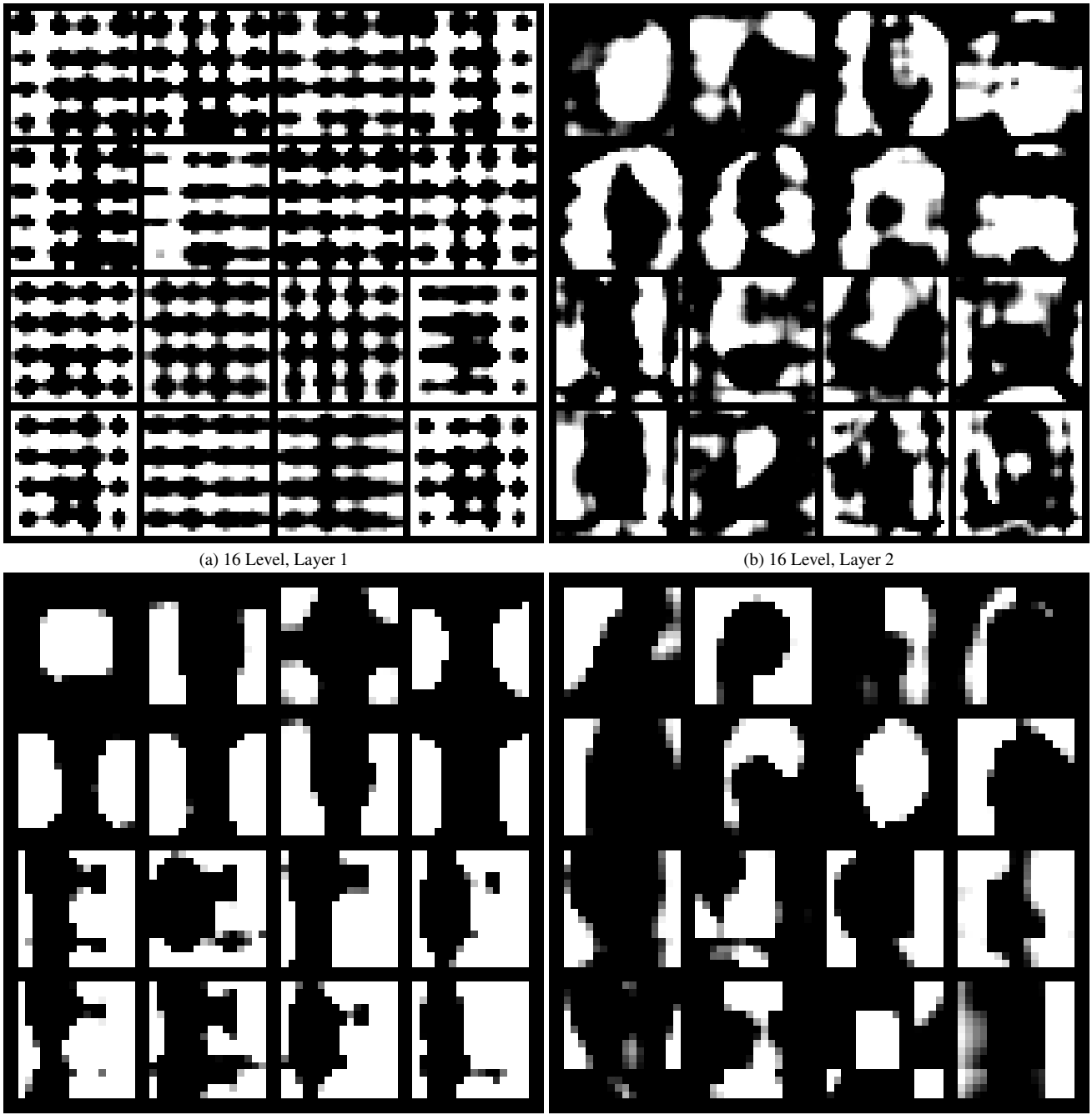


(c) 32 Level, Layer 1



(d) 32 Level, Layer 2

Figure A16. FFHQ StyleSwin attention maps at levels with 32 and 64 resolution. Every layer in each 32×32 level has 16 heads, and every layer in each 64×64 level has 8 heads, all with kernel size of 8, but half of them were trained with Shifted WSA.



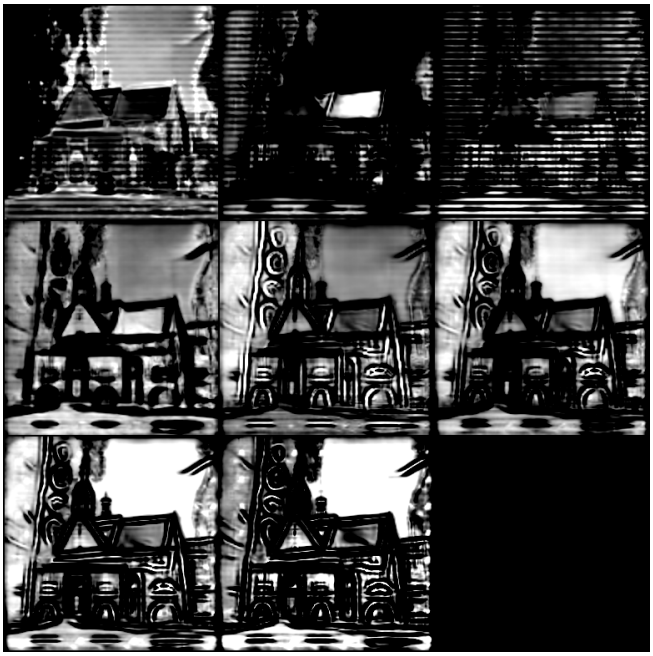
(a) 16 Level, Layer 1

(b) 16 Level, Layer 2

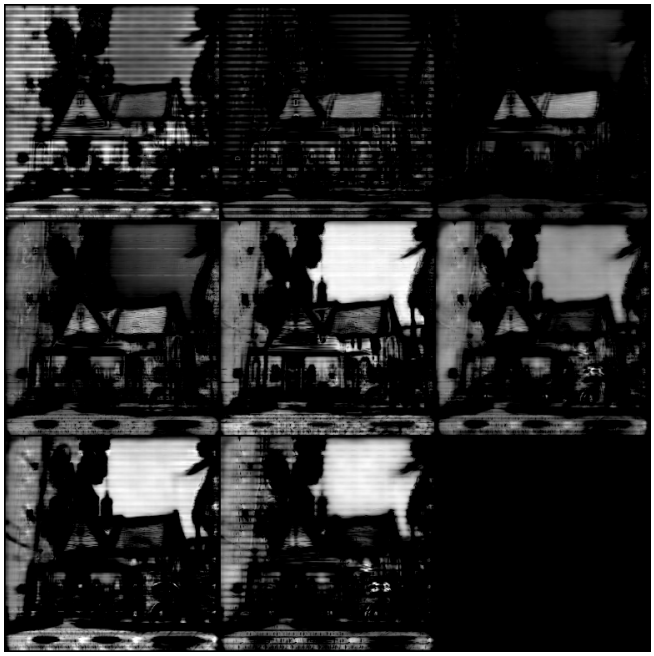
(c) 8 Level, Layer 1

(d) 8 Level, Layer 2

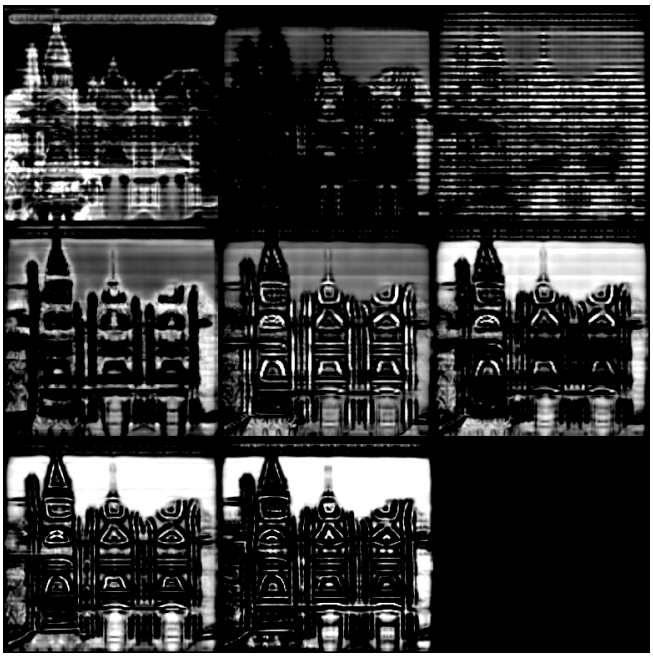
Figure A17. FFHQ StyleSwin attention maps at levels with 8 and 16 resolution. Every layer in each level has 16 heads with kernel size of 8 but half of them were trained with Shifted WSA.



(a) Good: 256 Level, Layer 1



(b) Good: 256 Level, Layer 2



(c) Bad: 256 Level, Layer 1

(d) Bad: 256 Level, Layer 2

Figure A18. Church StyleNAT 256 sized samples with bad and good samples. Generated images are highly predictable within both good and bad samples. Scanlines artifacts and hard lines are visible in both images, showing hard lines can be learned. "Tree trunk" like feature visible in good sample, with branches and swirls where foliage is located. Likely indicates a guide for the location of the tree in the scene rather than learning tree skeletal structures. Both maps can distinguish foreground and background. Bad sample looks more church like than the actual image. Notably the second layer, which we believe focuses on detail, has far lower activations in the bad sample. Despite progressive dilation, it is difficult to tell if heads are associated with local or global features, as was apparent in FFHQ.

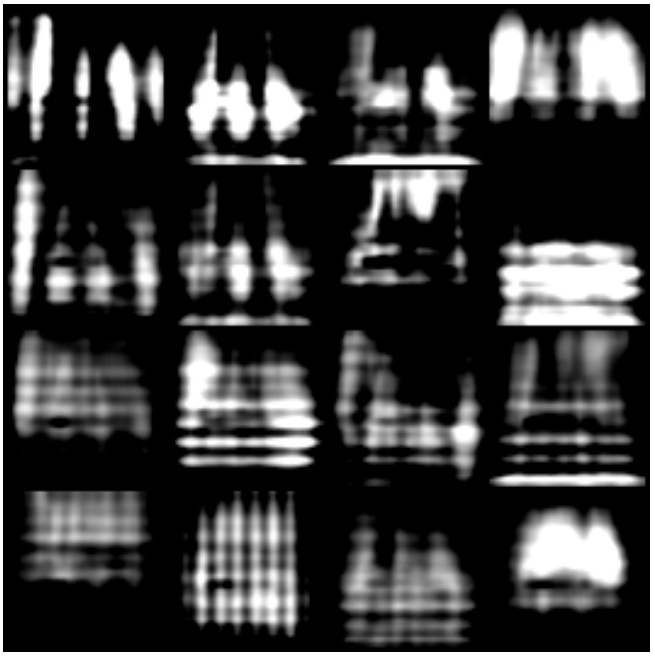
(a) Good: 128 Level, Layer 1

(b) Good: 128 Level, Layer 2

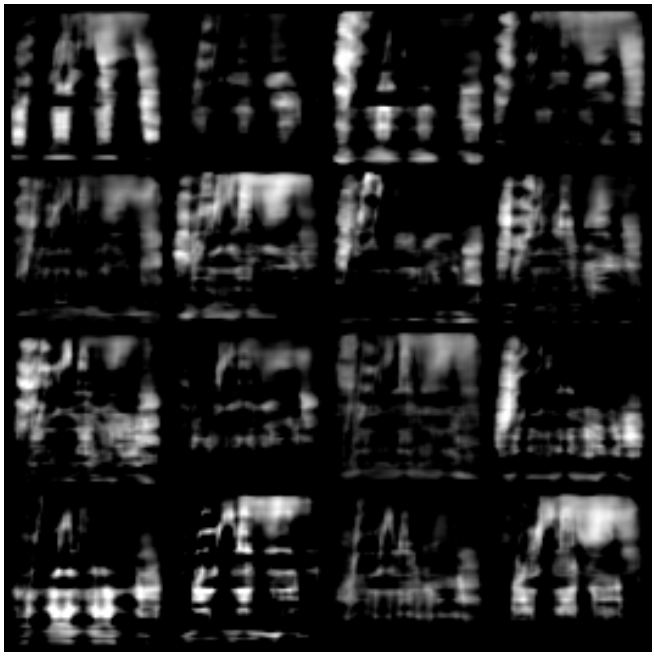
(c) Bad: 128 Level, Layer 1

(d) Bad: 128 Level, Layer 2

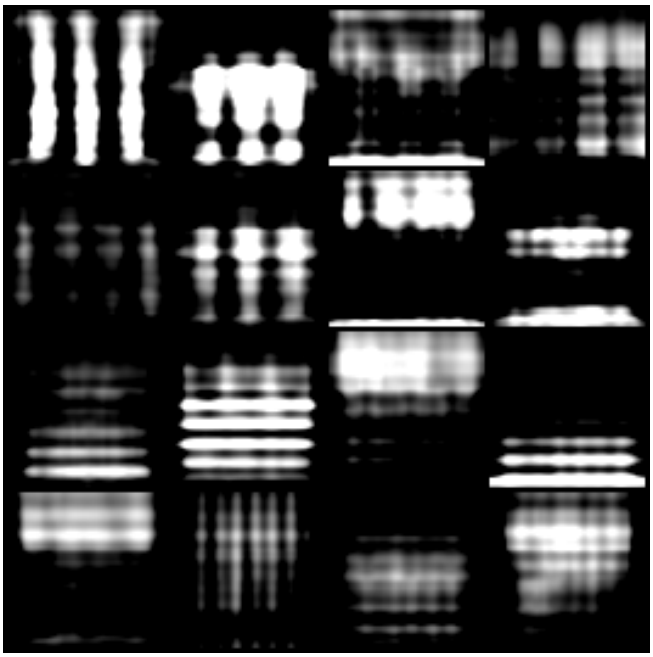
Figure A19. Church StyleNAT 128 sized samples with bad and good samples. Final image fairly predictable in the good sample but the bad sample looks more akin to stacked housing apartments. Scanlines weaker in the good sample and we can see loss of coherence in the bad sample. In both samples the detail layer has lower activations with one head appearing to dominate. We believe this decoherence propagates, preventing network from learning enough detail before scaling. Similar difficulties within StyleSwin indicate that this dataset may be more challenging and that detail is more important in lower resolutions. The early heads, which have no dilations, also clearly struggle to capture fine details. This is exceptionally apparent in the second layer which is more oriented towards this task.



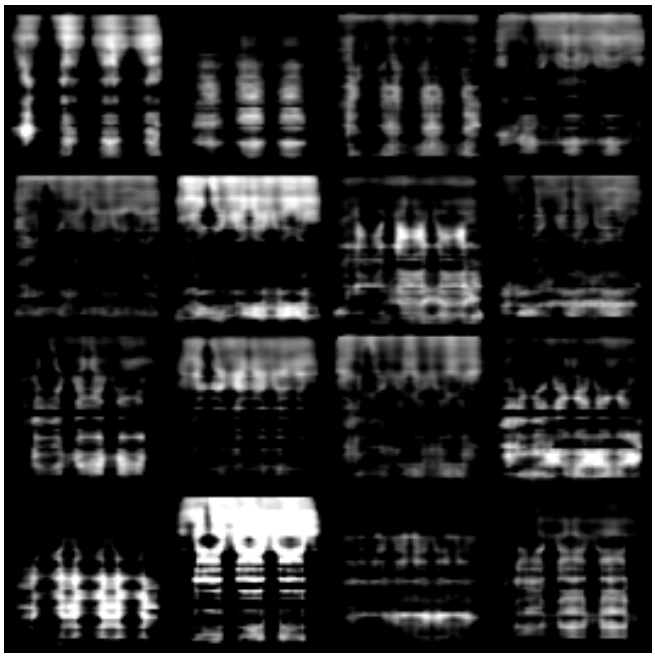
(a) Good: 64 Level, Layer 1



(b) Good: 64 Level, Layer 2

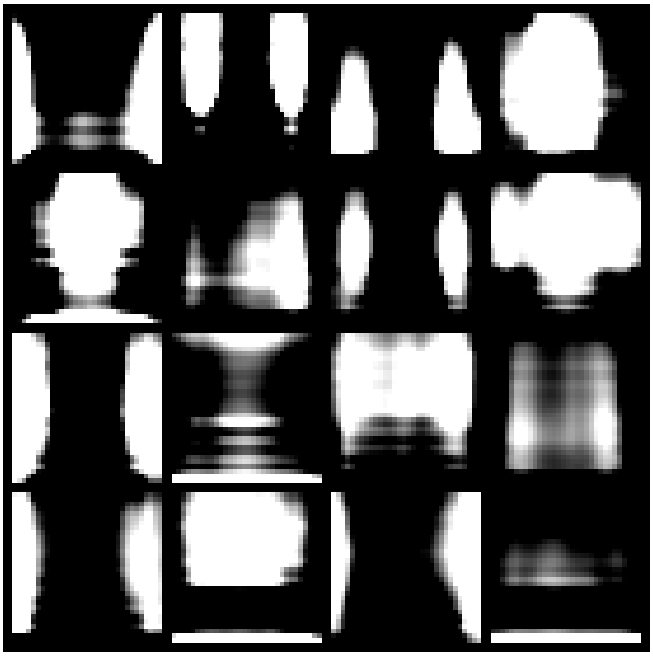


(c) Bad: 64 Level, Layer 1

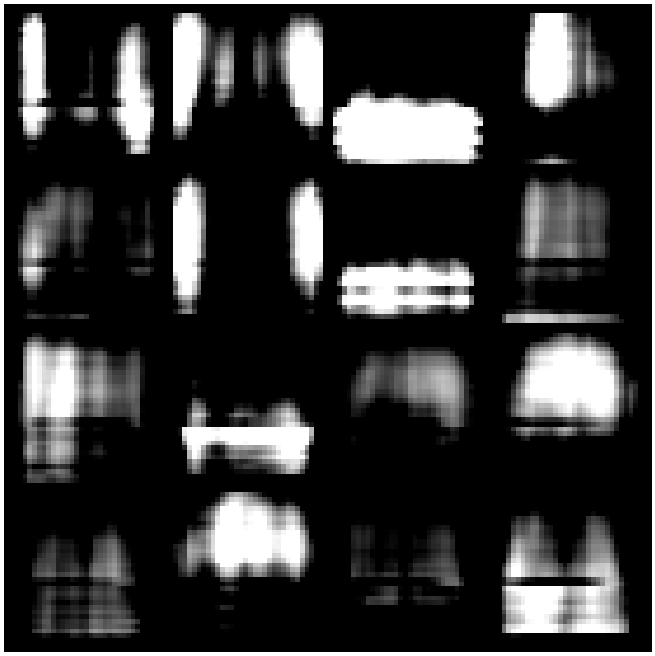


(d) Bad: 64 Level, Layer 2

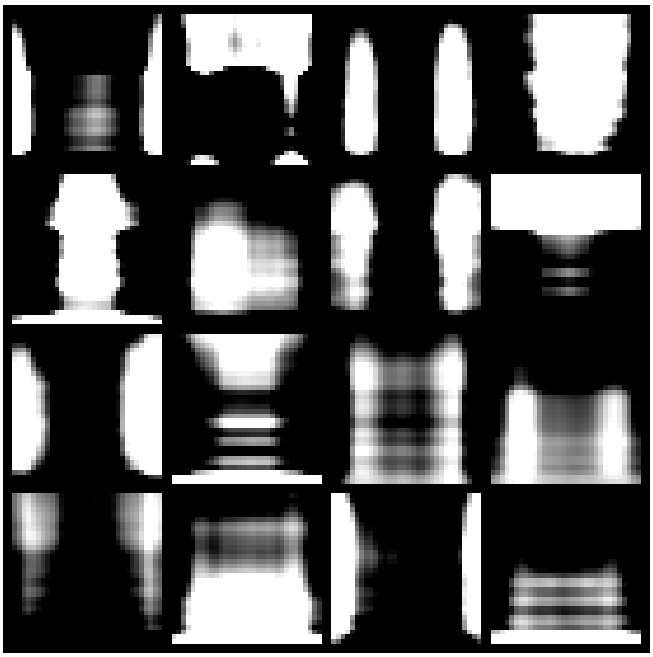
Figure A20. Church StyleNAT 64 sized samples with bad and good samples. The final images are not easily predictable at this resolution and we see little coherence. General shapes can be distinguished but this is not as strong as in FFHQ. First layer clearly focuses on general structure while the second on more detail. We continue to have difficulties associating head dilation with the corresponding receptive fields of the scene. The many bands suggest that there are difficulties in locating the object's placement within the scene. Both samples have tall tower like structures within the attention maps despite not being in final image or maps of the subsequent resolution. There are a lot of similarities between both samples, especially within the first layer. This could indicate overfitting and a strong preference to a strategy.



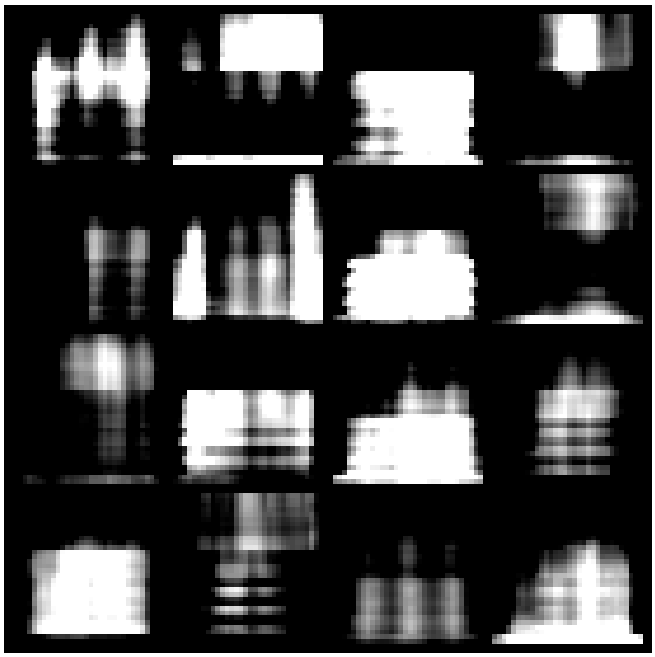
(a) Good: 32 Level, Layer 1



(b) Good: 32 Level, Layer 2



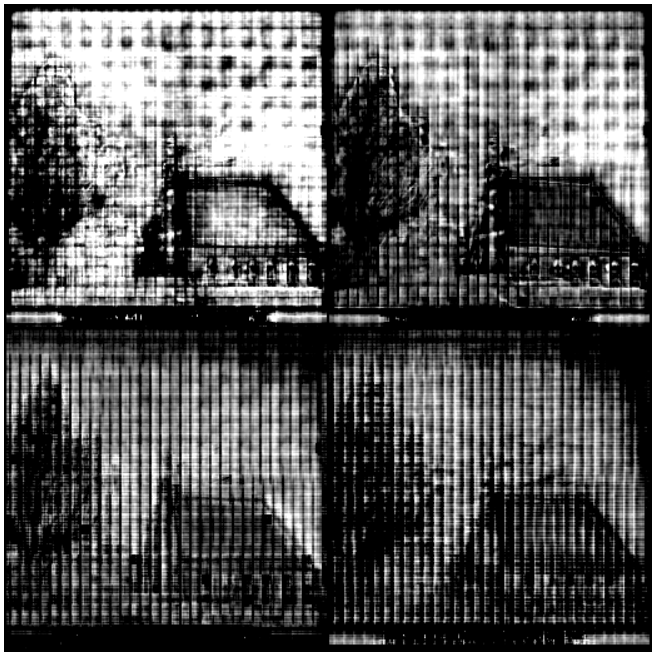
(c) Bad: 32 Level, Layer 1



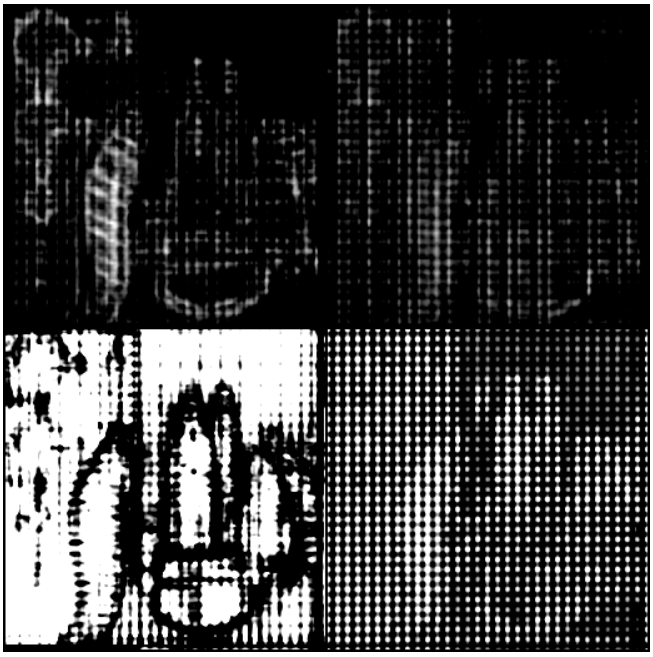
(d) Bad: 32 Level, Layer 2

Figure A21. Church StyleNAT 32 sized samples with bad and good samples. General structure is fairly coherent with blocky and tower like structures. Strong band at the bottom likely indicates attempt to generate shutterstock citation. In FFHQ we had clear placement of the subject within the scene at this level and even features like eyes and mouth. We see difficulties for this at this level, but do see towering structures. Unlike FFHQ this dataset has many differences in the general structure and location of main objects. This resolution has decent coherence for both samples but the lack of detail in the second layer may indicate how the lack of quality propagates within the network. Similar to StyleSwin these maps tend to put focus on the center of the image.

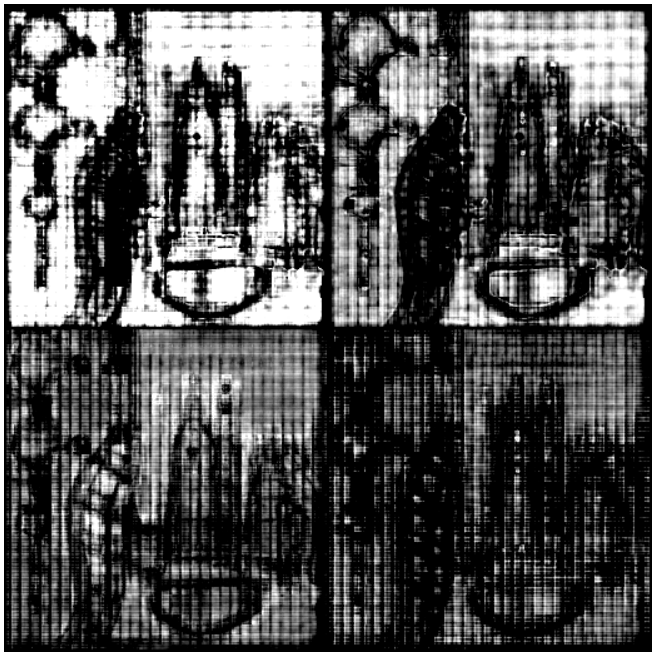
(a) Good: 256 Level, Layer 1



(b) Good: 256 Level, Layer 2



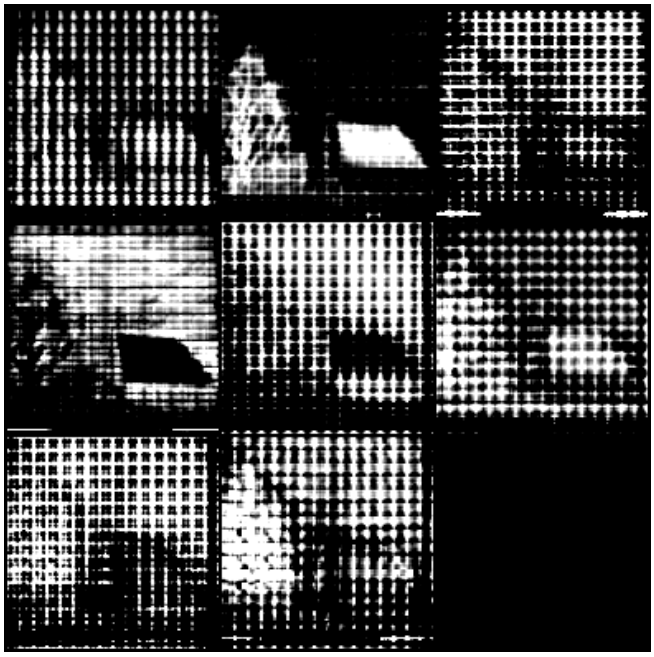
(c) Bad: 256 Level, Layer 1



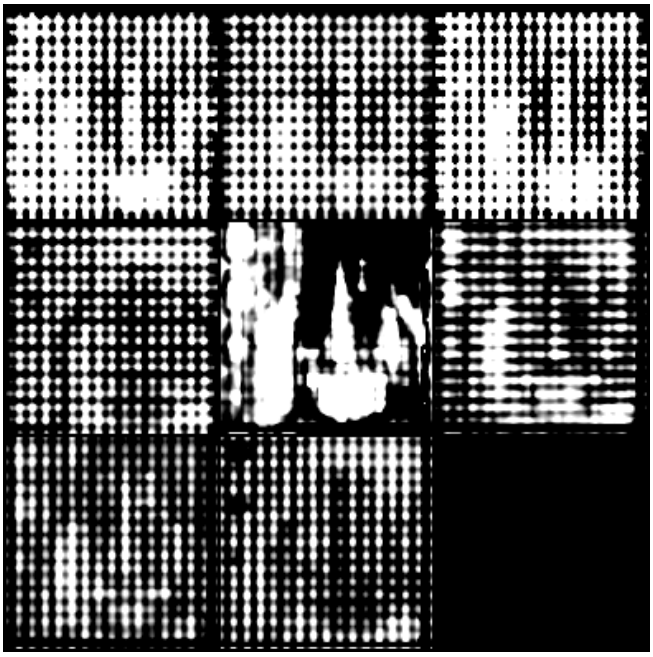
(d) Bad: 256 Level, Layer 2

Figure A22. Church StyleSwin 256 sized samples with bad and good samples. Blocky structure still exists akin to pointillism. Maps has filters reminiscent of edge filters, where the good sample can distinguish foreground and background. The tree and church are clearly visible and the good sample has a predictable final image. The floating telephone or watermark is not clearly identifiable here but we can see activations in the shutterstock citation at the bottom. Bad sample does not have as clear of an identification, and is more likely to have curved features. Similar to FFHQ the first 2 heads of the first layer have low activations while the other heads have disproportionately high.

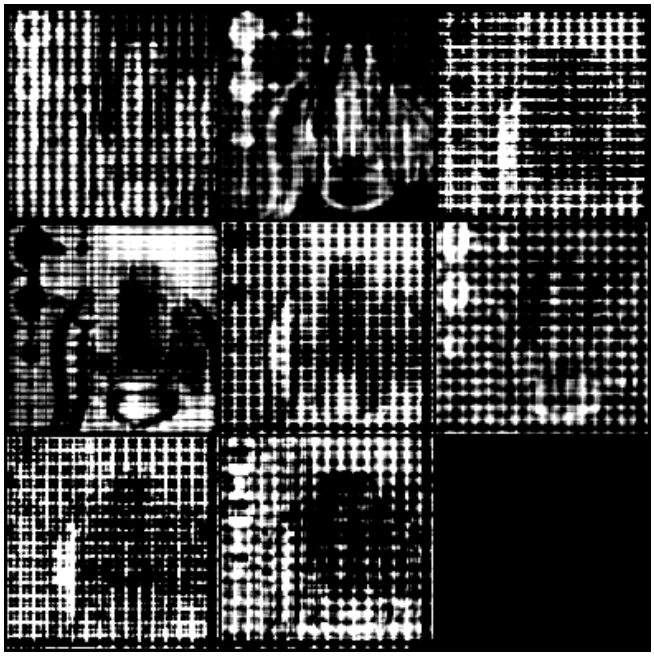
(a) Good: 128 Level, Layer 1



(b) Good: 128 Level, Layer 2

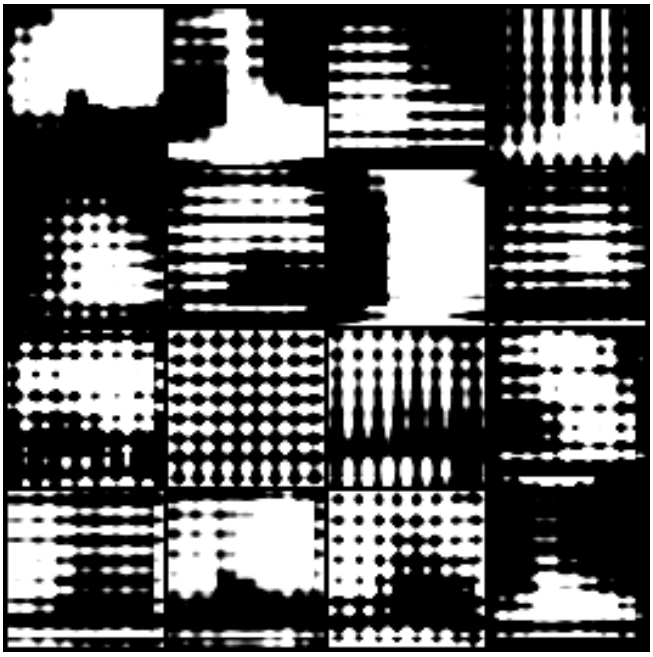


(c) Bad: 128 Level, Layer 1

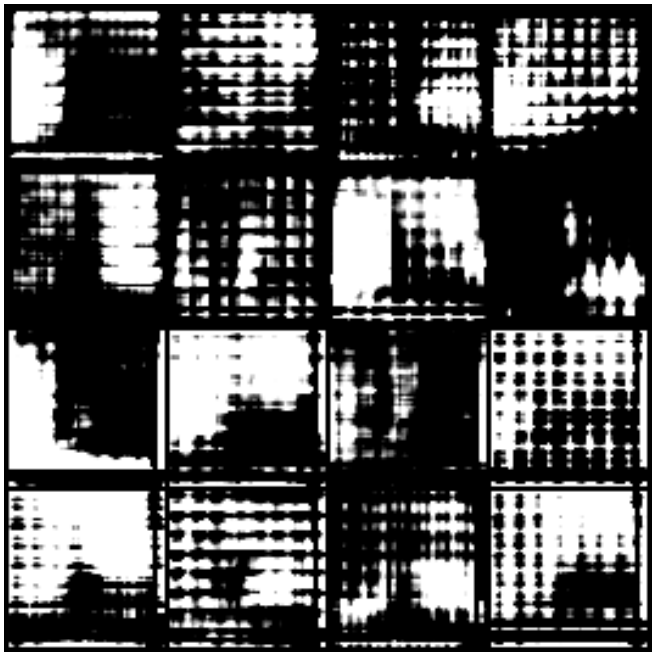


(d) Bad: 128 Level, Layer 2

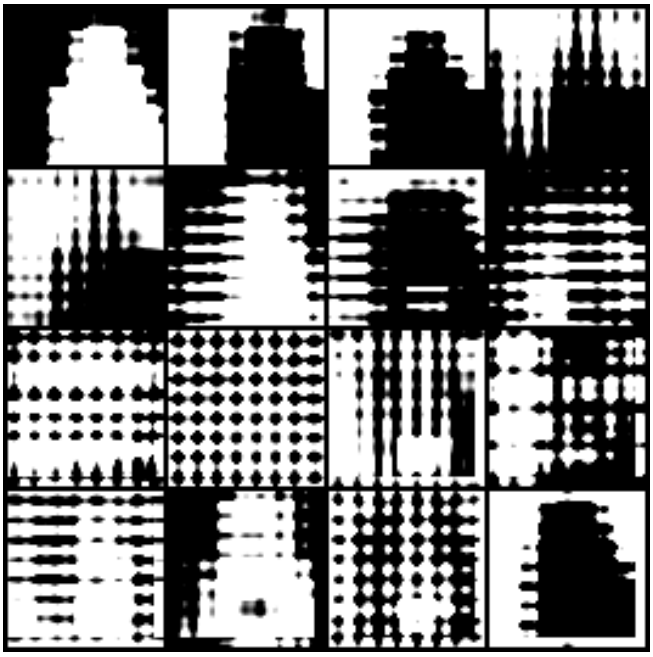
Figure A23. Church StyleSwin 128 sized samples with bad and good samples. Good sample has clear good filters and some heads have strong focus on the main objects in the scene. One head in the bad sample has this same clear filter. Maps have less detailed focus, activating on many different points within the scene. Maps have less structure and features at this resolution than we saw within the FFHQ examples. In FFHQ we had less pointillism, especially in the first layer, but this is extremely prominent here indicating a difficulty in attending to the scene. Attention activation is highly disproportionate at this resolution.



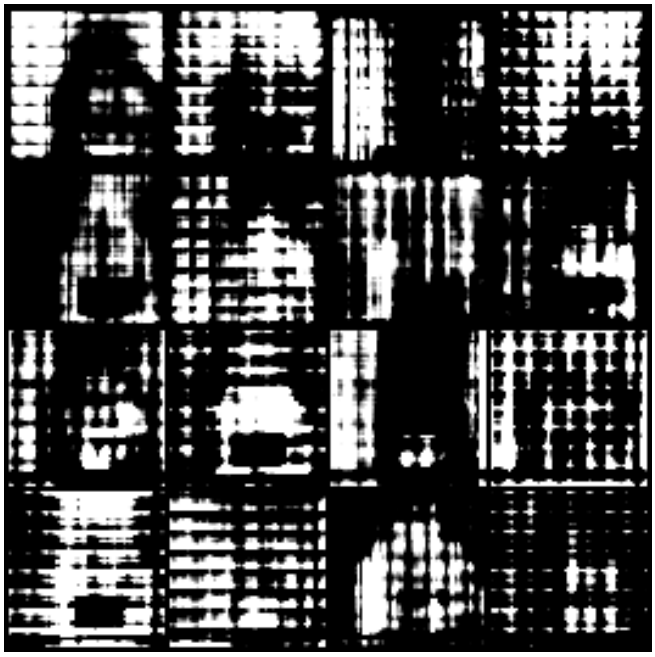
(a) Good: 64 Level, Layer 1



(b) Good: 64 Level, Layer 2

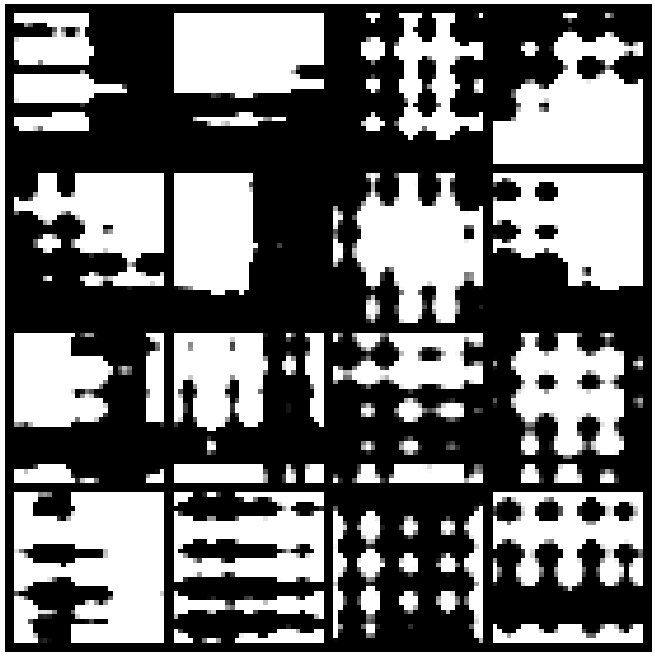


(c) Bad: 64 Level, Layer 1

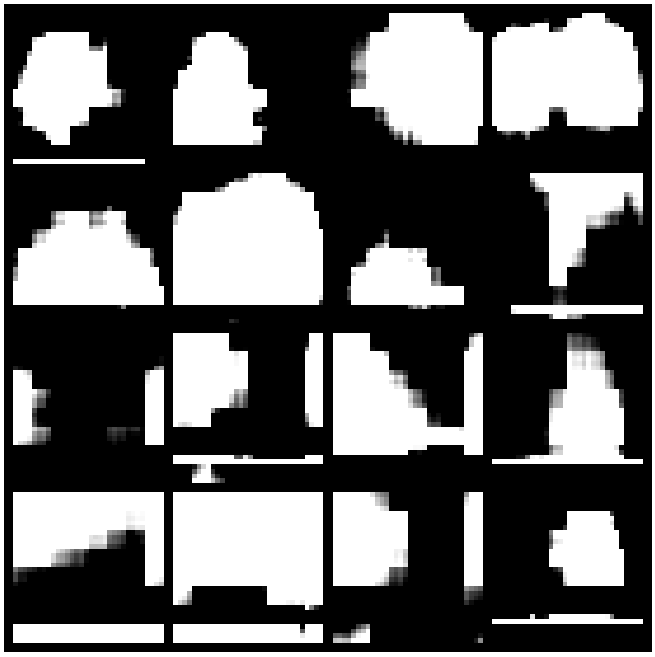


(d) Bad: 64 Level, Layer 2

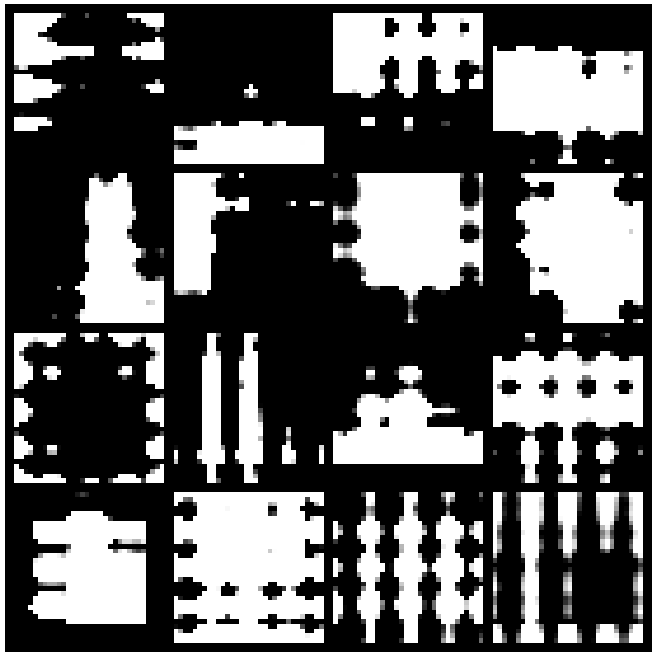
Figure A24. Church StyleSwin 64 sized samples with bad and good samples. Samples are difficult to differentiate at this level and we have lower interpretability. In FFHQ the face was locatable at this resolution and the second layer started to reduce the blocking. This resolution still appears to be concentrating on the main structure of the objects, but has large range contexts in both layers. In the good sample we have difficulties identifying the tree or church, but they are somewhat visible. Attentions are highly checkerboard, likely due to the shifting of windows. The bottom of the images indicates concentration on the shutterstock citation in both samples.



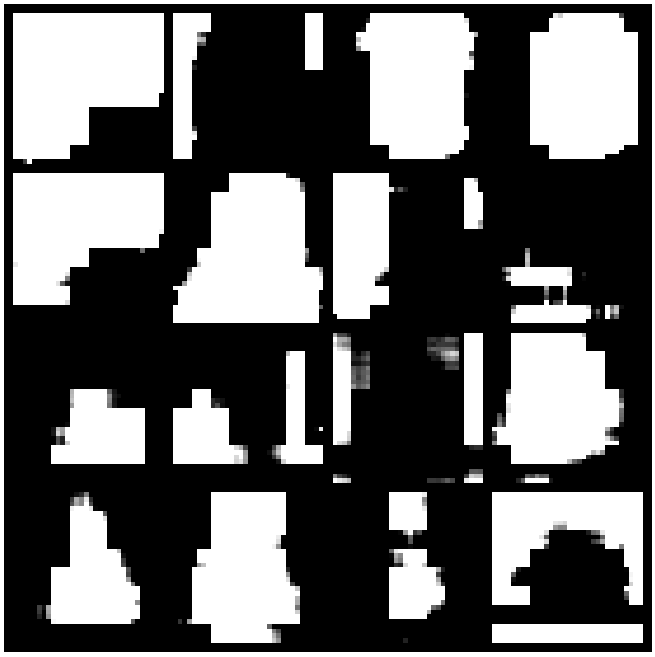
(a) Good: 32 Level, Layer 1



(b) Good: 32 Level, Layer 2



(c) Bad: 32 Level, Layer 1



(d) Bad: 32 Level, Layer 2

Figure A25. Church StyleSwin 32 sized samples with bad and good samples. Global structure is generally lost and would be difficult to predict produced sample from these maps. The church is identifiable in the second layer of the good sample, but attention is a bit scattered. First level is more sporadic compared to the second level, which is more connected. Similar to FFHQ the first layer has large checkerboard patterns and second layer is smoother, though less general structure is identifiable. This appears to indicate that the first layer is matching structure to the upscaling and the second layer concentrates on details. Coherence is likely lost after this resolution.