# Your Local GAN: Designing Two Dimensional Local Attention Mechanisms for Generative Models

Giannis Daras
National Technical University of Athens
daras.giannhs@gmail.com

Augustus Odena
Google Brain
augustusodena@google.com

Han Zhang
Google Brain
zhanghan@google.com

Alexandros G. Dimakis
UT Austin
dimakis@austin.utexas.edu

## Abstract

*We introduce a new local sparse attention layer that preserves two-dimensional geometry and locality. We show that by just replacing the dense attention layer of SAGAN with our construction, we obtain very significant FID, Inception score and pure visual improvements. FID score is improved from 18.65 to **15.94** on ImageNet, keeping all other parameters the same. The sparse attention patterns that we propose for our new layer are designed using a novel information theoretic criterion that uses information flow graphs.*

*We also present a novel way to invert Generative Adversarial Networks with attention. Our method uses the attention layer of the discriminator to create an innovative loss function. This allows us to visualize the newly introduced attention heads and show that they indeed capture interesting aspects of two-dimensional geometry of real images.*

## 1. Introduction

Generative Adversarial Networks [10] are making significant progress on modeling and generating natural images [25, 3]. Transposed convolutional layers are a fundmamental architectural component since they capture spatial invariance, a key property of natural images [18, 15, 26]. The central limitation (e.g. as argued in [25]) is that convolutions fail to model complex geometries and long-distance dependencies – the canonical example is generating dogs with fewer or more than four legs.

To compensate for this limitation, attention layers [24] have been introduced in deep generative models [25, 3]. Attention enables the modeling of long range spatial dependencies in a single layer which automatically finds correlated parts of the image even if they are far apart. First introduced in SAGAN [25] and further improved in BigGAN [3], attention layers have led to some of the best known GANs currently available.

Attention layers have a few limitations. The first is that they are computationally inefficient: Standard dense attention requires memory and time complexity that scales quadratically in the size of the input. Second, dense attention layers are statistically inefficient: A significant number of training samples is required to train attention layers, a problem that becomes more pronounced when multiple attention heads or layers are introduced [5]. Statistical inefficiency also stems from the fact that dense attention does not benefit from locality, since most dependencies in images relate to nearby neighborhoods of pixels. Recent work indicates that most attention layer heads learn to attend mainly to local neighborhoods [23].

To mitigate these limitations, sparse attention layers were recently introduced in Sparse Transformers [5]. In that paper, different types of sparse attention kernels were introduced and used to obtain excellent results for images, text and audio data. They key observation we make is that the patterns that were introduced in Sparse Transformers are actually designed for one-dimensional data, such as text-sequences. Sparse Transformers [5] were applied to images by reshaping tensors in a way that significantly distorts distances of the two-dimensional grid of image pixels. Therefore, local sparse attention kernels introduced in Sparse Transformers fail to capture image locality.

**Our Contributions:**

- We introduce a new *local sparse attention layer* that preserves two-dimensional image locality and can support good information flow through attention steps.
- To design our attention patterns we use the information theoretic framework of Information Flow 3 Graphs [8]. This quantifies how information can flow through mul-

Figure 1: Samples generated by our model YLG-SAGAN after training on ImageNet. The images are visually significantly better compared to the SAGAN baseline, as also supported by FID and Inception score metrics.

tiple steps and preserve two-dimensional locality. We visualize learned attention maps and show that different heads indeed learn different aspects of the geometry of generated images.

- We modify SAGAN [25] using our new two-dimensional sparse attention layers to introduce Your Local GAN (YLG) - SAGAN. We empirically show that this change yields significant benefits. We train on ImageNet-128 and we achieve **14.53%** improvement to the FID score of SAGAN and **8.95%** improvement in Inception score, by only changing the attention layer while maintaining all other parameters of the architecture. Our ablation study shows that indeed the benefits come from two dimensional inductive bias and not from introducing multiple heads. Furthermore, YLG-SAGAN achieves this performance in $800k$ training steps as opposed to $1300k$ for SAGAN and hence reduces the training time by approximately $40\%$.

- To visualize our attention maps on natural images, we came across the problem of inverting a generator: given an image $x$, how to find a latent code $z$ so that $G(z)$ is as close as possible to $x$. The natural inversion process of performing gradient descent on this loss works in small GANs [2, 20, 19, 14] but has been notoriously failing in bigger models with attention like SAGAN[1]. We present a solution to the GAN inversion problem: We use the attention layer of the discriminator to obtain a weighting on the loss function that subsequently we use to invert with gradient descent. We empirically show excellent inversion results for cases where standard gradient descent inversion fails.

We open-source our code and pre-trained models to encourage further research in this area: https://github.com/giannisdaras/ylg[2].

[1]This fact is folklore, known among researchers who try to solve inverse problems with GANs. There are numerous other ways to invert (i.e. training an encoder) but also show poor results on modern attention GANs.

[2]Code based on tensorflow-gan library.

## 2. Background

**Dense Attention** Given matrices $X \in \mathbb{R}^{N_X \times E_X}$, $Y \in \mathbb{R}^{N_Y \times E_Y}$, attention of $X$ to $Y$, updates the vector representation of $X$ by integrating the vector representation of $Y$. In this paper, $X, Y$ are intermediate image representations. More specifically, attention of $X$ to $Y$ associates the following matrices with the inputs: The key matrix $Y_K = Y \cdot W_K$, the query matrix $X_Q = X \cdot W_Q$ and the value matrix $Y_V = Y \cdot W_V$ where $W_K \in \mathbb{R}^{E_Y \times E}, W_Q \in \mathbb{R}^{E_X \times E}, W_V \in \mathbb{R}^{E_Y \times E_V}$ are learnable weight matrices. Intuitively, queries are compared with keys and values translate the result of this comparison to a new vector representation of $X$ that integrates information from $Y$. Mathematically, the output of the attention is the matrix: $O = \sigma\left(X_Q \cdot Y_K^T\right) \cdot Y_V$.

**Sparsified Attention**. The quadratic complexity of attention to the size of the input is due to the calculation of the matrix $A_{X,Y} = X_Q \cdot Y_K^T, \in \mathbb{R}^{N_X \times N_Y}$. Instead of performing this calculation jointly, we can split attention in multiple steps. At each step $i$, we attend to a subset of input positions, specified by a binary mask $M_i \in \{0,1\}^{N_X \times N_Y}$. Mathematically, at step $i$ we calculate matrix $A^i_{X,Y}$, where: $A^i_{X,Y}[a,b] = \begin{cases} A_{X,Y}[a,b], & M^i[a,b] = 1 \\ -\infty, & M^i[a,b] = 0 \end{cases}$. In this expression, $-\infty$ means that after the softmax, this position will be zeroed and thus not contribute to the calculation of the output matrix. The design of the masks $\{M^i\}$ is key in reducing the number of positions attended.

There are several ways to perform multi-step attention [5] in practice. The simplest is to have separate heads [24] calculating matrices $\{A^i_{X,Y}\}$ in parallel and then concatenate along the feature dimension. We follow this approach (more details included in the Supplementary Material).

## 3. Your Local GAN

### 3.1. Full Information Attention Sparsification

As explained, an attention sparsification in $p$ steps is described by binary masks $\{M^1, ..., M^p\}$. The question is how to design a good set of masks for these steps. We introduce a tool from information theory to guide this design.

Information Flow Graphs are directed acyclic graphs introduced in [8] to model distributed storage systems through network information flow [1]. For our problem, this graph models how information flows across attention steps. For a given set of masks $\{M^1, ..., M^p\}$, we create a multi-partite graph $G(V = \{V^0, V^1, ..., V^p\}, E)$ where directed connections between $V^i, V^{i+1}$ are determined by mask $M^i$. Each group of vertices in partition $V^i$ corresponds to attention tokens of step $i$.

We say that an attention sparsification has **Full Information** if its corresponding Information Flow Graph has a directed path from every node $a \in V^0$ to every node $b \in V^p$. Please note that the Fixed pattern [5] shown in sub-figure 2a does not have Full Information: there is no path from node 2 of $V^0$ to node 1 of $V^2$.

Sparse attention is usually considered as a way to reduce the computational overhead of dense attention at a hopefully small performance loss. However, we show that attention masks chosen with a bias toward two-dimensional locality, can surprisingly *outperform* dense attention layers (compare the second and the third row of Table 1). This is an example of what we call the statistical inefficiency of dense attention. Sparse attention layers with locality create better inductive bias and hence can perform better in the finite sample regime. In the limit of infinite data, dense attention can always simulate sparse attention or perform better, in the same way that a fully connected layer can simulate a convolutional layer for a possible selection of weights.

We design the sparse patterns of YLG as the natural extensions of the patterns of [5] while ensuring that the corresponding Information Flow Graph supports Full Information. The first pattern, which we call Left to Right (LTR), extends the pattern of [5] to a bi-directional context. The second pattern, which we call Right to Left (RTL), is a transposed version of LTR. The corresponding $9 \times 9$ masks and associated Information Flow Graphs are presented in sub-figures 2b, 2e (LTR) and 2c, 2f (RTL). These patterns allow attention only to $n\sqrt{n}$ positions, significantly reducing the quadratic complexity of dense attention. It is possible to create very sparse Full Information graphs using multiple attention steps, but designing them and training them remains open for future work; in this paper we focus on two-step factorizations. We include more details about information flow graphs in Supplementary Material.

### 3.2. Two-Dimensional Locality

The factorization patterns of Sparse Transformers [5] and their Full Information extensions illustrated in Figure 2 are fundamentally matched to one-dimensional data, such as text-sequences.

The standard way to apply these layers on images is to reshape the three dimensional image tensors (having three color channels) to a two-dimensional tensor $X \in R^{N \times C}$ that enters attention. This corresponds to $N$ tokens, each containing a $C$-dimensional representation of a region of the input image. This reshape arranges these $N$ tokens linearly, significantly distorting which parts of the image are nearby in two dimensions. This behavior is illustrated in the sub-figure at the left of Figure 3.

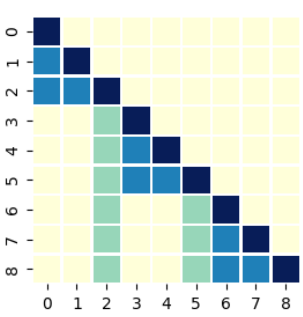
We argue that this is the reason that one-dimensional sparsifications are not ideal for images. In fact, the authors of [5] mention that the Fixed Pattern (Figure 2a) was designed for text-sequences and not for images. Our central finding is that these patterns *can* work very well for images, if their two dimensional structure is considered. The question is therefore how to take two-dimensional locality into account. We could create two-dimensional patterns directly on a grid but this would have significant computational overhead and also prevent us from extending one dimensional sparsifications that are known to work well [12, 5]. Instead, we modify one dimensional sparsifications to become aware of two-dimensional locality with the following trick: (i) we enumerate pixels of the image based on their Manhattan distance from location (0, 0) (breaking ties using row priority), (ii) shift the indices of any given one-dimensional sparsification to match the Manhattan distance enumeration instead of the reshape enumeration, and (iii) apply this new one dimensional sparsification pattern, that respects two-dimensional locality, to the one-dimensional reshaped version of the image. We call this procedure ESA (Enumerate, Shift, Apply) and illustrate it in Figure 3.

The ESA trick introduces some distortion compared to a true two-dimensional distance. We found however that this was not too limiting, at least for $128 \times 128$ resolution. Also, ESA offers an important implementation advantage: it theoretically allows the use of one-dimensional block-sparse kernels [11]. Currently these kernels exist only for GPUs, but making them work for TPUs is still under development.

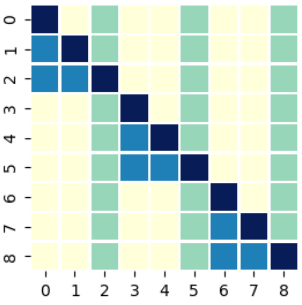
## 4. Experimental Validation

We conduct experiments on the ImageNet [21] dataset. We choose SAGAN [25] as the baseline for our models because, unlike BigGAN [3] it has official open-source Tensorflow code. BigGAN is not open-source and therefore training or modifying this architecture was not possible[3].

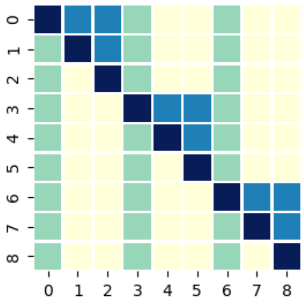
[3]There is an 'unofficial' PyTorch BigGAN implementation. However, it uses gradient checkpointing and requires 8 V100 GPUS for 15 days to



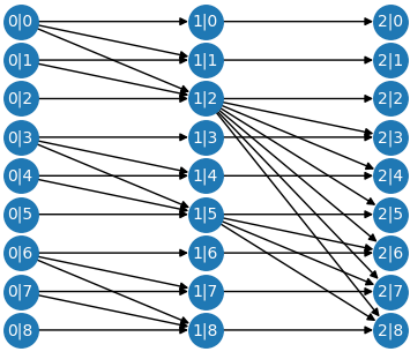
(a) Attention masks for Fixed Pattern [5].



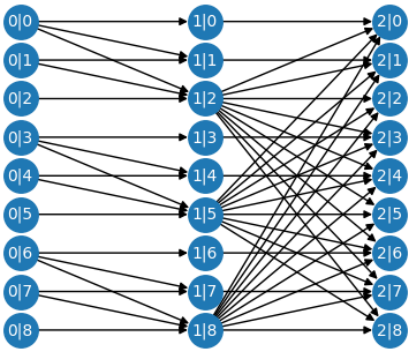
(b) Attention masks for Left To Right (LTR) pattern.



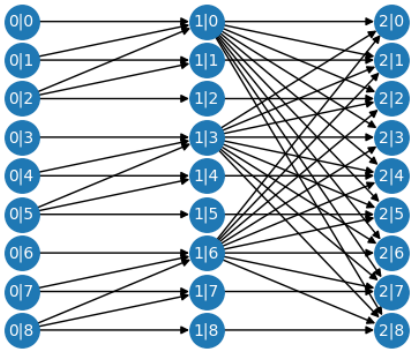
(c) Attention masks for Right To Left (RTL) pattern.



(d) Information Flow Graph associated with Fixed Pattern. This pattern *does not have Full Information*, i.e. there are dependencies between nodes that the attention layer cannot model. For example, there is no path from node 2 of $V^0$ to node 1 of $V^2$.



(e) Information Flow Graph associated with LTR. This pattern has **Full Information**, i.e. there is a path between any node of $V^0$ and any node of $V^2$. Note that the number of edges is only increased by a constant compared to the Fixed Attention Pattern [5], illustrated in 2d.



(f) Information Flow Graph associated with RTL. This pattern also has **Full Information**. RTL is a "transposed" version of LTR, so that local context at the right of each node is attended at the first step.

Figure 2: This Figure illustrates the different 2-step sparsifications of the attention layer we examine in this paper. First row demonstrates the different boolean masks that we apply to each of the two steps. Color of cell [i. j] indicates whether node i can attend to node j. With dark blue we indicate the attended positions in both steps. With light blue the positions of the first mask and with green the positions of the second mask. The yellow cells correspond to positions that we do not attend to any step (sparsity). The second row illustrates Information Flow Graph associated with the aforementioned attention masks. An Information Flow Graph visualizes how information "flows" in the attention layer. Intuitively, it visualizes how our model can use the 2-step factorization to find dependencies between image pixels. At each multipartite graph, the nodes of the first vertex set correspond to the image pixels, just before the attention. An edge from a node of the first vertex set, $V^0$, to a node of the second vertex set, $V^1$, means that the node of $V^0$ can attend to node of $V^1$ at the first attention step. Edges between $V^1, V^2$ illustrate the second attention step.

In all our experiments, we change *only the attention layer* of SAGAN, keeping all the other hyper-parameters unchanged (the number of parameters is not affected). We trained all models for up to 1,500,000 steps on on individual Cloud TPU v3 devices (v3-8)[4], using a $1e^{-4}$ learning rate for generator and $4e^{-4}$ for the discriminator. For all the models we report the best performance obtained, even if it was obtained at an earlier point during training.

**Attention Mechanism** We start with the Fixed Pattern (Figure 2a) and modify it: First, we create Full Information extensions (Section 3.1), yielding the patterns Left-To-Right (LTR) and Right-To-Left (RTL) (Figures 2b and 2c respectively). We implement multi-step attention in parallel using different heads. Since each pattern is a two-step sparsification, this yields 4 attention heads. To encourage diversity of learned patterns, we use each pattern twice, so

train. We simply did not have such computing resources. We believe, however, that YLG can be easily combined with BigGAN (by simply replacing its dense attention layer) and will yield an even better model.

[4]This research has been supported by NSF Grants 1618689, DMS 1723052, CCF 1763702, AF 1901292 and research gifts by Google, Western Digital and NVIDIA. The TPUs were offered by the TFRC program.

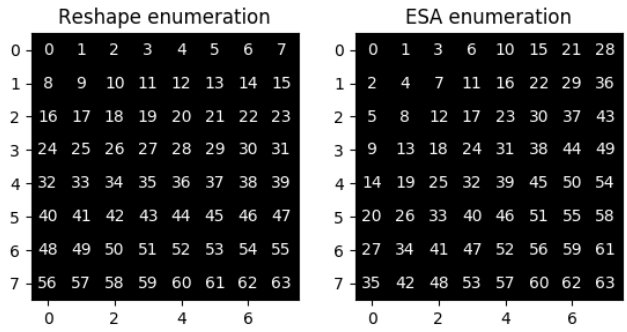



Figure 3: Reshape and ESA enumerations of the cells of an image grid that show how image grid is projected into a line. (Left) Enumeration of pixels of an $8 \times 8$ image using a standard reshape. This projection maintains locality only in rows. (Right) Enumeration of pixels of an $8 \times 8$ image, using the ESA framework. We use the Manhattan distance from the start $(0, 0)$ as a criterion for enumeration. Although there is some distortion due to the projection into 1-D, locality is mostly maintained.

the total number of heads in our new attention layer is 8. We use our ESA procedure (Section 3.2) to render these patterns aware of two dimensional geometry.

**Non-Square Attention** In SAGAN, the query image and the key image in the attention layer have different dimensions. This complicates things, because the sparsification patterns we discuss are designed for self-attention, where the number of query and key nodes is the same. Specifically, for SAGAN the query image is $32 \times 32$ and the key image is $16 \times 16$. We deal with this in the simplest possible way: we create masks for the $16 \times 16$ image and we shift these masks to cover the area of the $32 \times 32$ image. Thus every $16 \times 16$ block of the $32 \times 32$ query image attends with full information to the $16 \times 16$ key image.

| | # Heads | FID | Inception |
|---|---|---|---|
| SAGAN | 1 | 18.65 | 52.52 |
| SAGAN | 8 | 20.09 | 46.01 |
| **YLG-SAGAN** | 8 | **15.94** | **57.22** |
| YLG - No ESA | 8 | 17.47 | 51.09 |
| YLG - Strided | 8 | 16.64 | 55.21 |

Table 1: Table of results after training SAGAN and YLG-SAGAN on ImageNet. Table also includes Ablation Studies (SAGAN 8 heads, YLG - No ESA, YLG - Strided). Our best model, **YLG**, achieves **15.94** FID and **57.22** Inception score. Our scores correspond to **14.53%** and **8.95%** improvement to FID and Inception respectively. We emphasize that these benefits are obtained by only one layer change to SAGAN, replacing dense attention with the local sparse attention layer that we introduce.

**Results:** As shown in Table 1, YLG-SAGAN (3rd row) outperforms SAGAN by a large margin measured by both FID and Inception score. Specifically, YLG-SAGAN increases Inception score to **57.22** ($8.95\%$ improvement) and improves FID to **15.94** ($14.53\%$ improvement). Qualitatively, we observe really good-looking samples for categories with simple geometries and homogeneity. Intuitively, a two-dimensional locality can benefit importantly categories such as valleys or mountains, because usually the image transitions for these categories are smoother compared to others and thus the dependencies are mostly local.

Additionally to the significantly improved scores, one important benefit of using YLG sparse layer instead of a dense attention layer, is that we observe significant reduction of the training time needed for the model to reach it's optimal performance. SAGAN reached it's best FID score after more that 1.3 million training steps while YLG-SAGAN reaches its' optimal score after **only 865,000 steps** ($\approx 40\%$ reduction to the training time). Figure 4 illustrates SAGAN and YLG-SAGAN FID and Inception score as a function of the training time.

We create two collages to display samples from our YLG version of SAGAN. At the Upper Panel of Figure 7, we show dogs of different breeds generated by our YLG-SAN. At the Lower Panel, we use YLG-SAGAN to generate samples from randomly chosen classes of the ImageNet dataset.

### 4.1. Ablation Studies

**Number of Attention Heads** The original SAGAN implementation used a single-headed attention mechanism. In YLG, we use multiple heads to perform parallel multi-step sparse attention. Previous work has shown that multiple heads increased performance for Natural Language Processing tasks [24]. To understand how multiple heads affect SAGAN performance, we train an 8 head version of SAGAN. The results are reported in the second row of Table 1. Multiple heads actually **worsen** significantly the performance of the original SAGAN, reducing Inception score from 52.52 to 46.01. We provide a post-hoc interpretation of this result. The image embedding of the query vector of SAGAN has only 32 vector positions. By using 8 heads, each head gets only 4 positions for its' vector representation. Our intuition is that a 4-positions vector representation is not sufficient for effective encoding of the image information for a dense head and that accounts for the decrease in performance. It is important to note that YLG-SAGAN does not suffer from this problem. The reason is that each head is sparse, which means that only attends to a percentage of the positions that dense head attends to. Thus, a smaller vector representation does not worsen performance. Having multiple divergent sparse heads allows YLG layer to discover complex dependencies in the image space throughout the multi-step attention.

**Two-Dimensional Locality** As described in Section 3.2 YLG uses the ESA procedure, to adapt 1-D sparse patterns to data with 2-D structure. Our motivation was that grid-

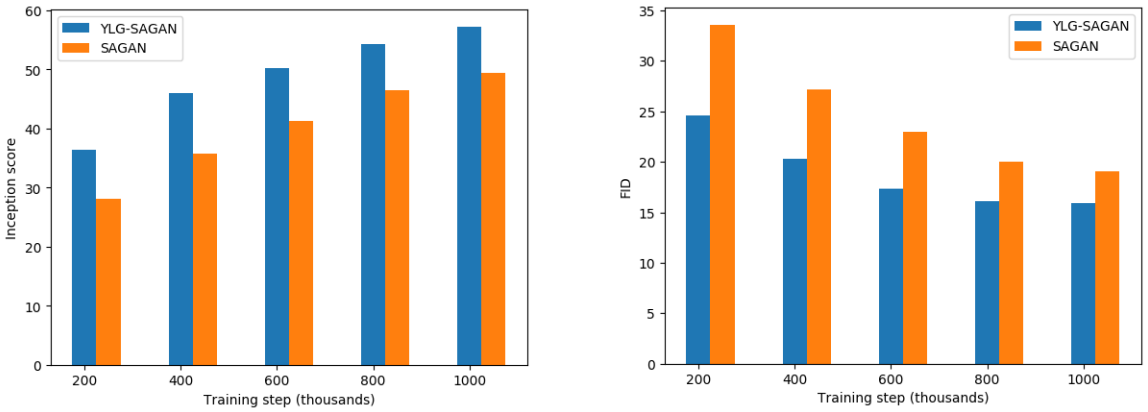



Figure 4: Training comparison for YLG-SAGAN and SAGAN. We plot every 200k steps the Inception score (a) and the FID (b) of both YLG-SAGAN and SAGAN, up to 1M training steps on ImageNet. As it can be seen, YLG-SAGAN converges much faster compared to the baseline. Specifically, we obtain our best FID at step 865k, while SAGAN requires over 1.3M steps to reach its FID performance peak. Comparing peak performance for both models, we obtain an improvement from 18.65 to **15.94** FID, by only changing the attention layer.

locality could help our sparse attention layer to better model local regions. In order to validate this experimentally, we trained a version of YLG *without* the ESA procedure. We call this model YLG - No ESA. The results are shown in 4th row of Table 1: without the ESA procedure, the performance of YLG is about the same with the original SAGAN. This experiment indicates that ESA trick is essential for using 1-D sparse patterns for grid-structured data. With ESA, FID improves from 17.47 to 15.94 and Inception score from 51.09 to 57.22, without any other difference in the architecture. Thus, ESA is a plug-and-play framework that achieves great performance boosts to both FID and Inception score metrics. ESA allows the utilization of fast sparse 1-D patterns that were found to work well for text-sequences to be adapted to images, with great performance benefits. In section 5.1, we visualize attention maps to showcase how our model utilizes ESA framework in practice.

**Sparse Patterns** Our YLG layer uses the LTR and RTL patterns (Figures 2b and 2c respectively). Our intuition is that using multiple patterns at the same time increases performance because the model will be able to discover dependencies using multiple different paths. To test this intuition, we ran an experiment using the Full Information extension of the Strided [5] pattern. We choose this pattern because it was found to be effective for modeling images [5] due to its' periodic structure. As with LTR and RTL patterns, we extend the Strided pattern so that it has Full Information. We refer to the YLG model that instead of LTR and RTL patterns, has 8 heads implementing the Strided pattern as YLG - Strided. For our experiment, we use again the ESA trick. We report the results on the 5th row of Table 1. YLG - Strided importantly surpasses SAGAN both in FID and Inception score, however, it is still behind YLG. Although in the Sparse Transformers [5] it has been claimed that strided pattern is more suitable for images than the patterns we use in YLG, this experiment strongly suggests that it is the grid-locality which makes the difference, as both models are far better than SAGAN. Also, this experiment indicates that multiple sparse patterns can boost performance compared to using a single sparse pattern. To be noted, using multiple different patterns at the same attention layer requires scaling the number of heads as well. Although YLG variations of SAGAN were not impacted negatively by the increase of attention heads, more severe up-scaling of the number of heads could potentially harm performance, similarly to how 8 heads harmed performance of SAGAN.

## 5. Inverting Generative Models with Attention

We are interested in visualizing our sparse attention on real images, not just generated ones. This leads naturally to the problem of *projecting an image on the range of a generator*, also called inversion. Given a real image $x \in \mathbb{R}^n$ and a generator $G(z)$, inversion corresponds to finding a latent variable $z^* \in \mathbb{R}^k$, so that $G(z^*) \in \mathbb{R}^n$ approximates the given image $x$ as well as possible. One approach is to try to solve the following non-convex optimization problem:

$$\underset{z^*}{\text{argmin}}\{\|G(z^*) - x\|^2\}. \quad (1)$$

To solve this optimization problem, we can perform gradient descent from a random initalization $z_0$ to minimize this projection distance in the latent space. This approach was introduced independently in several papers [16, 2, 20] and further generalized to solve inverse problems beyond inversion [2, 20, 19, 14]. Recent research [13, 22] demonstrated that for fully connected generators with random weights and sufficient layer expansion, gradient descent will provably converge to the correct optimal inversion.

Unfortunately, this theory does not apply for generators that have attention layers. Even empirically, inversion by gradient descent fails for bigger generative models like SAGAN and YLG-SAGAN. As we show in our experiments the optimizer gets trapped in local minimima producing reconstructions that only vaguely resemble the target image. Other approaches for inversion have been tried in the literature, like training jointly an encoder [9] but none of these methods have been known to successfully invert complex generative models with attention layers.

We propose a novel inversion method that uses the discriminator to solve the minimization problem in an different representation space. Interestingly, the discriminator yields representations with a smoother loss landscape, especially if we use the attention layer in a special way. In more detail: We begin with a random latent variable $z$ and a given real image $x$. We denote with $D^0$ the Discriminator network up to, but not including, the attention layer and obtain the representations $D^0(G(z))$ and $D^0(x)$. We could perform gradient descent to minimize the distance of these discriminator representations:

$$\|D^0(G(z)) - D^0(x)\|^2.$$

We found, however, that we can use the attention map of the real image to further enhance inversion. We will use the example of the SAGAN architecture to illustrate this. Inside the SAGAN Discriminator's attention, a map $M \in \mathbb{R}^{32\times32\times16\times16}$ is calculated. For each pixel of the $32 \times 32$ image, this attention map is a distribution over the pixels of the $16 \times 16$ image. We can use this attention map to extract a *saliency map*. For each pixel of the $16 \times 16$ image, we can average the probabilities from all the pixels of the $32 \times 32$ image and create a distribution $S$ of shape $16 \times 16$. Intuitively, this distribution represents how important each pixel of the image is to the discriminator.

Our proposed inversion algorithm is to perform gradient descent to minimize the discriminator embedding distance, weighted by these saliency maps:

$$\|\big(D^0(G(z)) - D^0(x)\big) \cdot S'\|^2, \tag{2}$$

where $S'$ is a projected version of saliency map $S$ to the dimensions of $D^0(x)$. We actually calculate one saliency map $S'$ per head and use their sum as the final loss function that we optimize for inversion. More details are included in the Supplementary Material.

### 5.1. Inversion as lens to attention

Given an arbitrary real image, we can now solve for a $z$ yielding a similar generated image from the generator, and visualize the attention maps.

We explain our approach using an example of a real image of a redshank (Figure 5a). Figure 5b shows how the standard method for inverting generators [2] fails: the beak, legs, and rocks are missing. Figure 5c shows the result of our method. Using the $z$ that we found using inversion, we can project maps of the attention layer back to the original image to get valuable insight into how the YLG layers work.

First, we analyze the differences between the YLG-SAGAN attention heads. For each attention head of the generator, we create a saliency map as described above and use these maps to analyze the attention mechanism. As shown in Figure 5d, the head-7 in the generator is mostly ignoring background focusing on the bird. Other heads function differently: The saliency map of head-2 (Figure 5e) shows that this head attends globally. We also find that there are heads that that attend quite sparsely, for example, head-5 attends only to 5-6 background pixels.

We present a second inversion, this time an indigo bird (Figure 6a). Figure 6b shows how the standard method [2] for inverting fails: the head of the bird and the branch are not reconstructed. We also illustrate where specific query points attend to. We first illustrate that the the model exploited the local bias of ESA: We plot the attention map for query point $(0, 0)$ for generator-head-0. This point, indicated with a blue dot, is part of the background. We clearly see a local bias in the positions this point attends to. Another example of *two-dimensional* local attention is shown in Figure 6e. This figure illustrates the attention map of generator-head-4 for a query point on the body of the bird (blue dot). This point attends to the edges of the bird body and to the bird head.

Finally, Figure 6f shows that there are query points that attend to long-distance, demonstrating that the attention mechanism is capable of exploiting both locality and long-distance relationships when these appear in the image.

## 6. Related Work

There has been a flourishing of novel ideas on making attention mechanisms more efficient. Dai et al. [7] separate inputs into chunks and associate a state vector with previous chunks of the input. Attention is performed per chunk, but information exchange between chunks is possible via the state vector. Guo et al. [12] show that a star-shaped topology can reduce attention cost from $O(n^2)$ to $O(n)$ in text sequences. Interestingly, this topology does have full information, under our framework. Sukhbaatar et al. [23] introduced the idea of a learnable adaptive span for each attention layer. The idea of learnable patters is also explored by Correia et. al [6]. Calian et al. [4] proposed a fast randomized algorithm that exploits spatial coherence and sparsity to design sparse approximations. We believe that all these methods can be possibly combined with YLG, but so far nothing has been demonstrated to improve generative models in a plug-and-play way that this work shows.

There is also prior work on using attention mechanisms

(a) (b) (c) (d) (e)

Figure 5: Inversion and Saliency maps for different heads of the Generator network. We emphasize that this image of a redshank bird was not in the training set, it is rather obtained by a Google image search. Saliency is extracted by averaging the attention each pixel of the key image gets from the query image. We use the same trick to enhance inversion. (a) A real image of a redshank. (b) A demonstration of how the standard inversion method [2] fails. (c) The inverted image for this redshank, using our technique. (d) Saliency map for head 7. Attention is mostly applied to the bird body. (e) Saliency map for head 2. This head attends almost everywhere in the image.

(a) (b) (c) (d) (e) (f)

Figure 6: Inverted image of an indigo bird and visualization of the attention maps for specific query points. (a) The original image. Again, this was obtained with a Google image search and was not in the training set. (b) Shows how previous inversion methods fail to reconstruct the head of the bird and the branch. (c) A successful inversion using our method. (d) Specifically, 6d shows how attention uses our ESA trick to model background, homogeneous areas. (e) Attention applied to the bird. (f) Attention applied with a query on the branch. Notice how attention is non-local and captures the full branch.

to model images: One notable example is Zhang et al. [25], which we have discussed extensively and which adds a self-attention mechanism to GANs. See also Parmar et al. [17], which uses local-attention that is not multi-step.

## 7. Conclusions and Future Work

We introduced a new type of local sparse attention layer designed for two-dimensional data. We believe that our layer will be widely applicable for any model with attention that works on two-dimensional data. An interesting future direction is the design of attention layers, thought of as multi-step networks with connections that can either be manually chosen or learned. The two conflicting objectives are to make these networks as sparse as possible (for computational and statistical efficiency) but also support good information flow. We introduced information flow graphs as a mathematical abstraction and proposed full information as a desired criterion for such networks.

Finally, we presented a novel way to solve the inversion problem for GANs. Our technique uses the discriminator in two ways: First, using its attention to obtain pixel importance and second, as a smoothing representation of the inversion loss landscape. This new inversion method allowed us to visualize our network on approximations of real images and also to test how good a generative model is in this important coverage task. We believe that this is the first key step towards using generative models for inverse problems.

Figure 7: Upper Panel: YLG conditional image generation on different dog breeds from ImageNet dataset. From up to down: eskimo husky, siberian husky, saint bernard, maltese. Lower Panel: Random generated samples from YLG-SAGAN.

## References


[1] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. Network information flow. *IEEE Transactions on information theory*, 46(4):1204–1216, 2000. 3

[2] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 537–546. JMLR. org, 2017. 2, 6, 7, 8

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv e-prints*, page arXiv:1809.11096, Sep 2018. 1, 3

[4] Dan A. Calian, Peter Roelants, Jacques Cali, Ben Carr, Krishna Dubba, John E. Reid, and Dell Zhang. SCRAM: Spatially Coherent Randomized Attention Maps. *arXiv e-prints*, page arXiv:1905.10308, May 2019. 7

[5] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 1, 2, 3, 4, 6

[6] Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively sparse transformers. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. 7

[7] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv e-prints*, page arXiv:1901.02860, Jan 2019. 7

[8] Alexandros G Dimakis, P Brighten Godfrey, Yunnan Wu, Martin J Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *IEEE transactions on information theory*, 56(9):4539–4551, 2010. 1, 3

[9] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. 7

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, Jun 2014. 1

[11] Scott Gray, Alec Radford, and Diederik P Kingma. Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 2017. 3

[12] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-Transformer. *arXiv e-prints*, page arXiv:1902.09113, Feb 2019. 3, 7

[13] Paul Hand and Vladislav Voroninski. Global guarantees for enforcing deep generative priors by empirical risk. *IEEE Transactions on Information Theory*, 2019. 6

[14] Maya Kabkab, Pouya Samangouei, and Rama Chellappa. Task-aware compressed sensing with generative adversarial networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2, 6

[15] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1812.04948, Dec 2018. 1

[16] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017. 6

[17] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018. 8

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1511.06434, Nov 2015. 1

[19] Ankit Raj, Yuqi Li, and Yoram Bresler. Gan-based projector for faster recovery with convergence guarantees in linear inverse problems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5602–5611, 2019. 2, 6

[20] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017. 2, 6

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3

[22] Ganlin Song, Zhou Fan, and John Lafferty. Surfing: Iterative optimization over incrementally trained deep networks. *arXiv preprint arXiv:1907.08653*, 2019. 6

[23] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive Attention Span in Transformers. *arXiv e-prints*, page arXiv:1905.07799, May 2019. 1, 7

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv e-prints*, page arXiv:1706.03762, Jun 2017. 1, 2, 5

[25] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 1, 2, 3, 8

[26] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stack-GAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1612.03242, Dec 2016. 1