SAGAN

(Self- attention GANS)

Problems with Convolutional Based Gans

- Difficult to train on multi-class dataset like imagenet
- They learn to generate simple images like ocean, sky
- Often fails to generate complex images like dog(only able to generate furs but fails on details like seperate legs)

Why is this problem arising?

- Because convolutions have local receptive field
- Can’t capture long range dependency

How to solve?

Traditional approaches using convolution

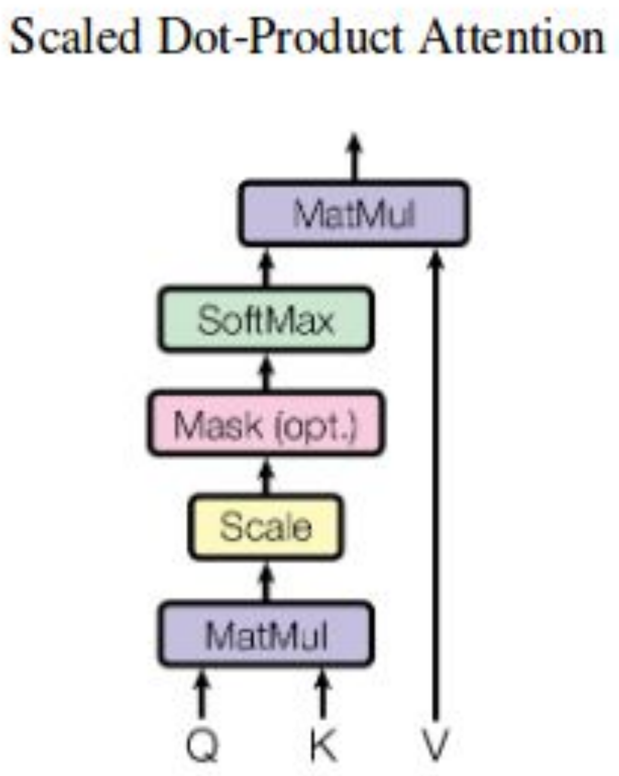
- Let’s increase the spatial size of the kernel?
- :(Will reduce efficiency

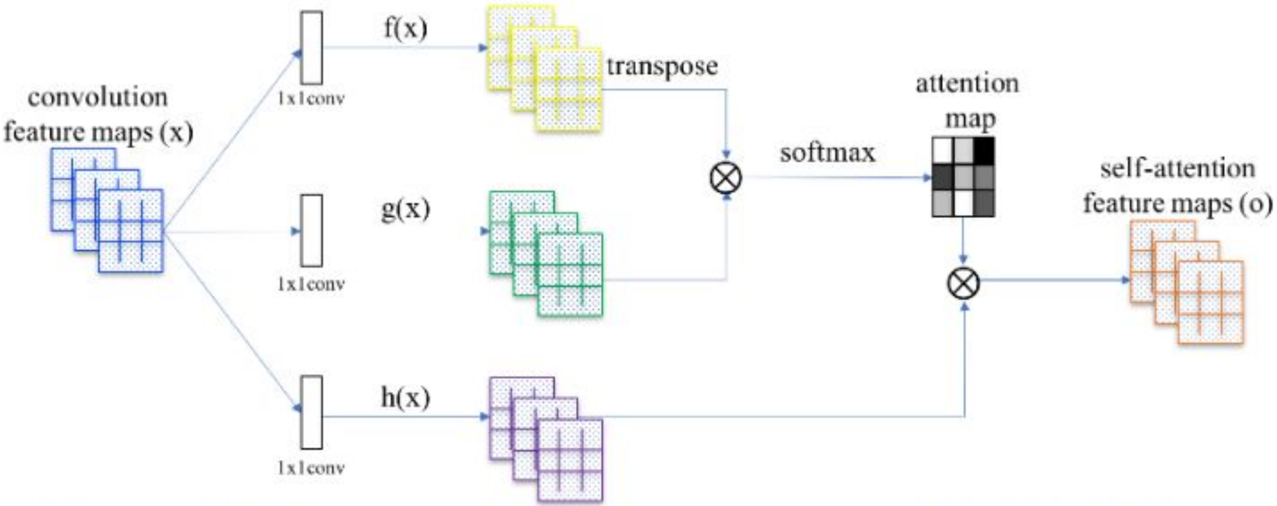
- To have large receptive field we can increase number of layers it takes large amount of layers for which training becomes unstable
- Unable to converge

Solution: SAGAN(Self- attention GANS)

- Idea derived from Vaswani et. al (AIAYN)
- Creates a balance b/w efficiency and long term dependency
- A few more tricks
    - Spectral Norm(on weights)
    - TTUR(update rule)
- It does not replace convolution but is complementary
- Captures fine details from distant locations in the image

What is Attention?





gure 2: The proposed self-attention mechanism. The ⊗ denotes matrix multiplication. The softmax
›eration is performed on each row.

The image features from the previous hidden layer $\boldsymbol{x} \in \mathbb{R}^{C \times N}$ are first transformed into two feature spaces $\boldsymbol{f}, \boldsymbol{g}$ to calculate the attention, where $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W_f x}$, $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{W_g x}$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}, \text{ where } s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T \boldsymbol{g}(\boldsymbol{x_j}), \tag{1}$$

and $\beta_{j,i}$ indicates the extent to which the model attends to the i^{th} location when synthesizing the j^{th} region. Then the output of the attention layer is $\boldsymbol{o} = (\boldsymbol{o_1}, \boldsymbol{o_2}, ..., \boldsymbol{o_j}, ..., \boldsymbol{o_N}) \in \mathbb{R}^{C \times N}$, where,

$$\boldsymbol{o_j} = \sum_{i=1}^{N} \beta_{j,i} \boldsymbol{h}(\boldsymbol{x_i}), \text{ where } \boldsymbol{h}(\boldsymbol{x_i}) = \boldsymbol{W_h x_i}. \tag{2}$$

In the above formulation, $\boldsymbol{W_g} \in \mathbb{R}^{\bar{C} \times C}$, $\boldsymbol{W_f} \in \mathbb{R}^{\bar{C} \times C}$, $\boldsymbol{W_h} \in \mathbb{R}^{C \times C}$ are the learned weight matrices, which are implemented as 1×1 convolutions. We use $\bar{C} = C/8$ in all our experiments.

In addition, we further multiply the output of the attention layer by a scale parameter and add back the input feature map. Therefore, the final output is given by,

$$\boldsymbol{y_i} = \gamma \boldsymbol{o_i} + \boldsymbol{x_i}, \tag{3}$$

where γ is initialized as 0. This allows the network to first rely on the cues in the local neighborhood – since this is easier – and then gradually learn to assign more weight to the non-local evidence. The intuition for why we do this is straightforward: we want to learn the easy task first and then

Techniques to improve training

Spectral Normalisation on Weights

Constraints the Lipschitz constant of the weights(controls gradients)

Applied to both G & D (in previous paper it is applied only to D)

TTUR:

Two time sclae update rule

Different learning rate for G and D

Metrics

IS - Incepttion score (Higher the better)

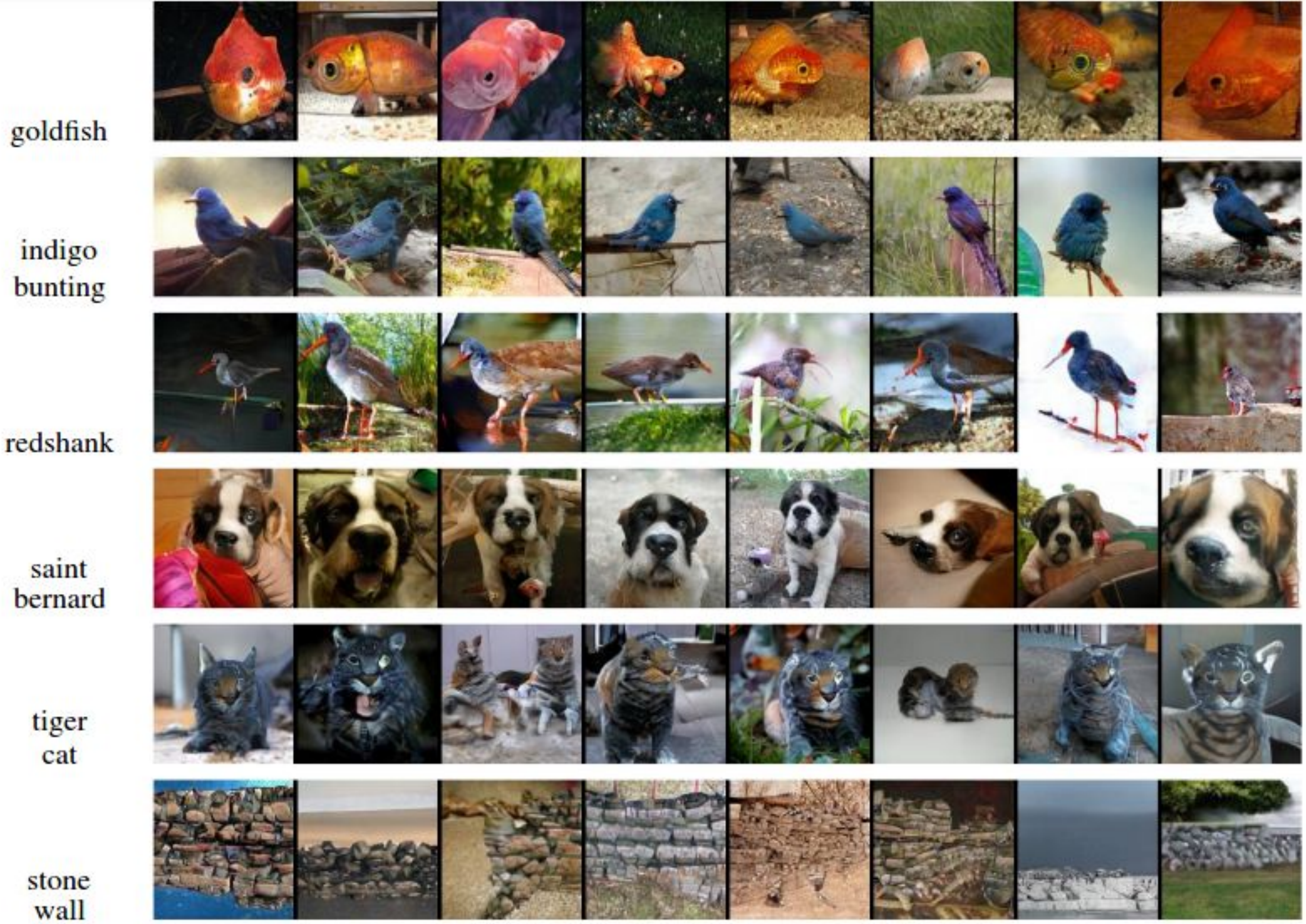
FID - (Frechet Inception Distance)

Wasserstein 2 distance of feature layer Inception-V3 network

Effect of SN and TTUR (From paper)

Results and State of the art

Model	Inception Score	FID
AC-GAN [31]	28.5	/
SNGAN-projection [17]	36.8	27.62*
SAGAN	**52.52**	**18.65**


goldfish
indigo bunting
redshank
saint bernard
tiger cat
stone wall