

Deep Learning for Computer Vision

GAN Improvements

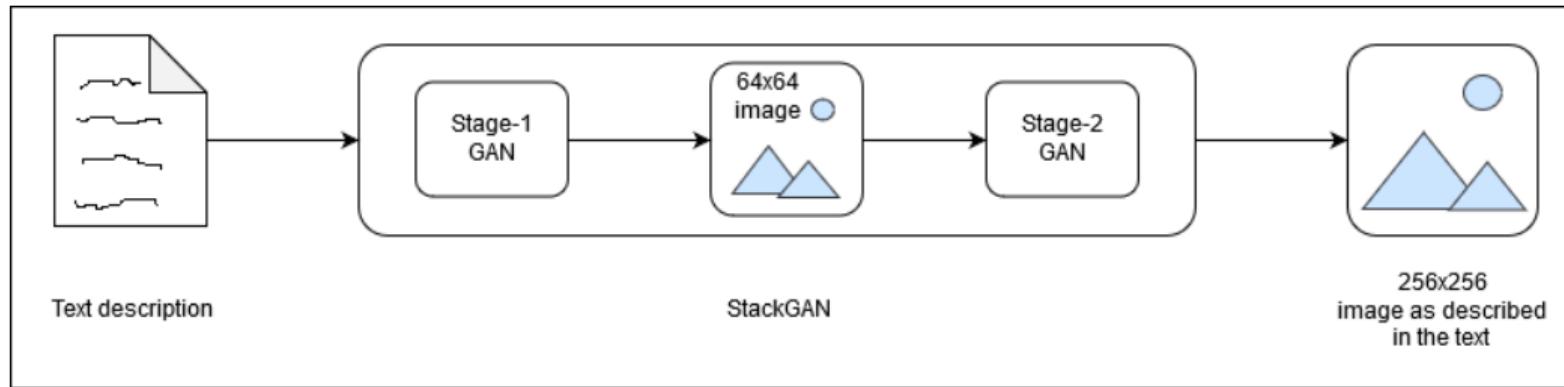
Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad



StackGAN¹

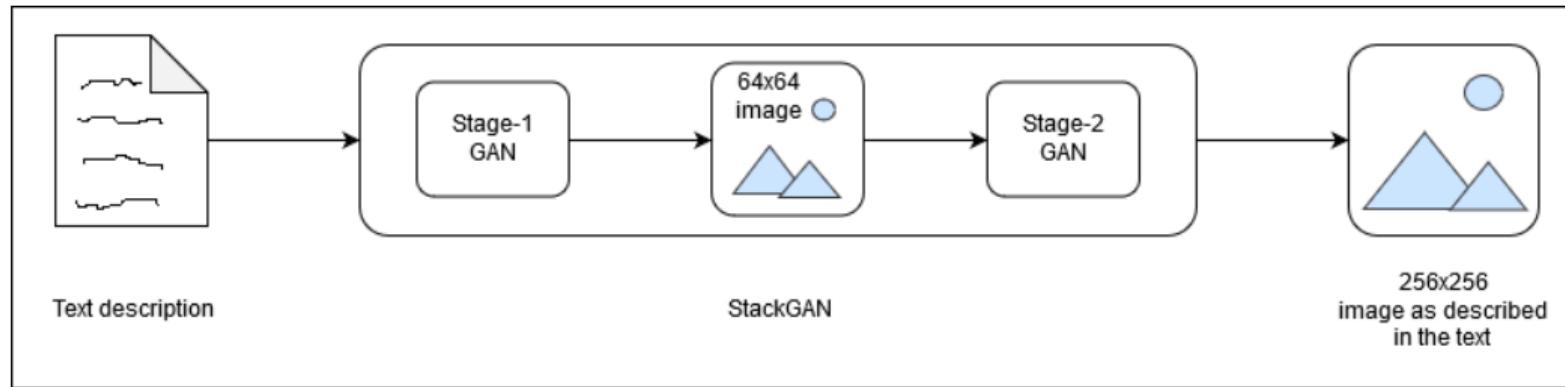
Generate 256 × 256 photo-realistic images conditioned on text descriptions



¹Zhang et al, StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, ICCV 2017

StackGAN¹

Generate 256×256 photo-realistic images conditioned on text descriptions

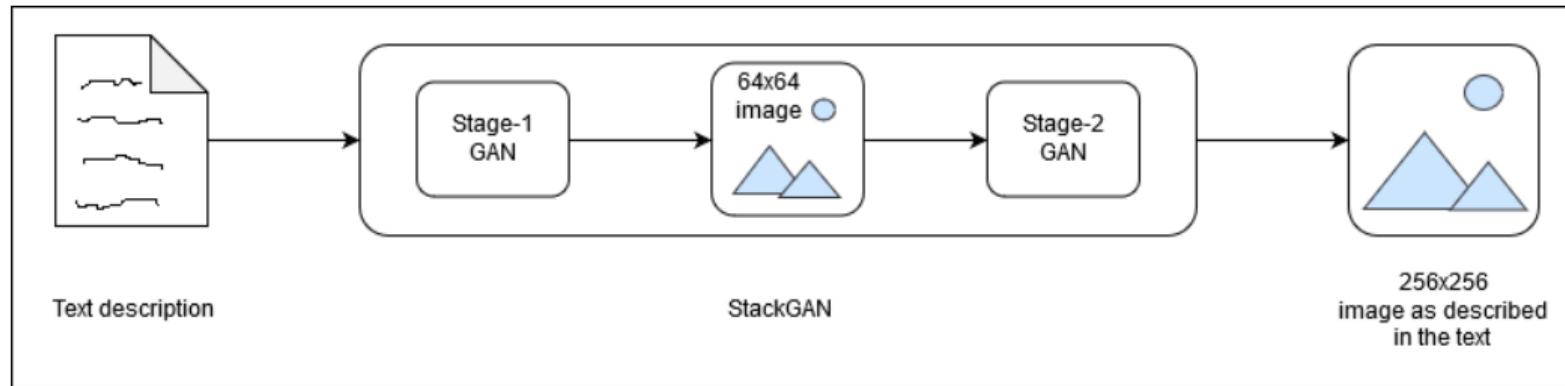


- **Stage 1:** Generate 64×64 images, low details

¹Zhang et al, StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, ICCV 2017

StackGAN¹

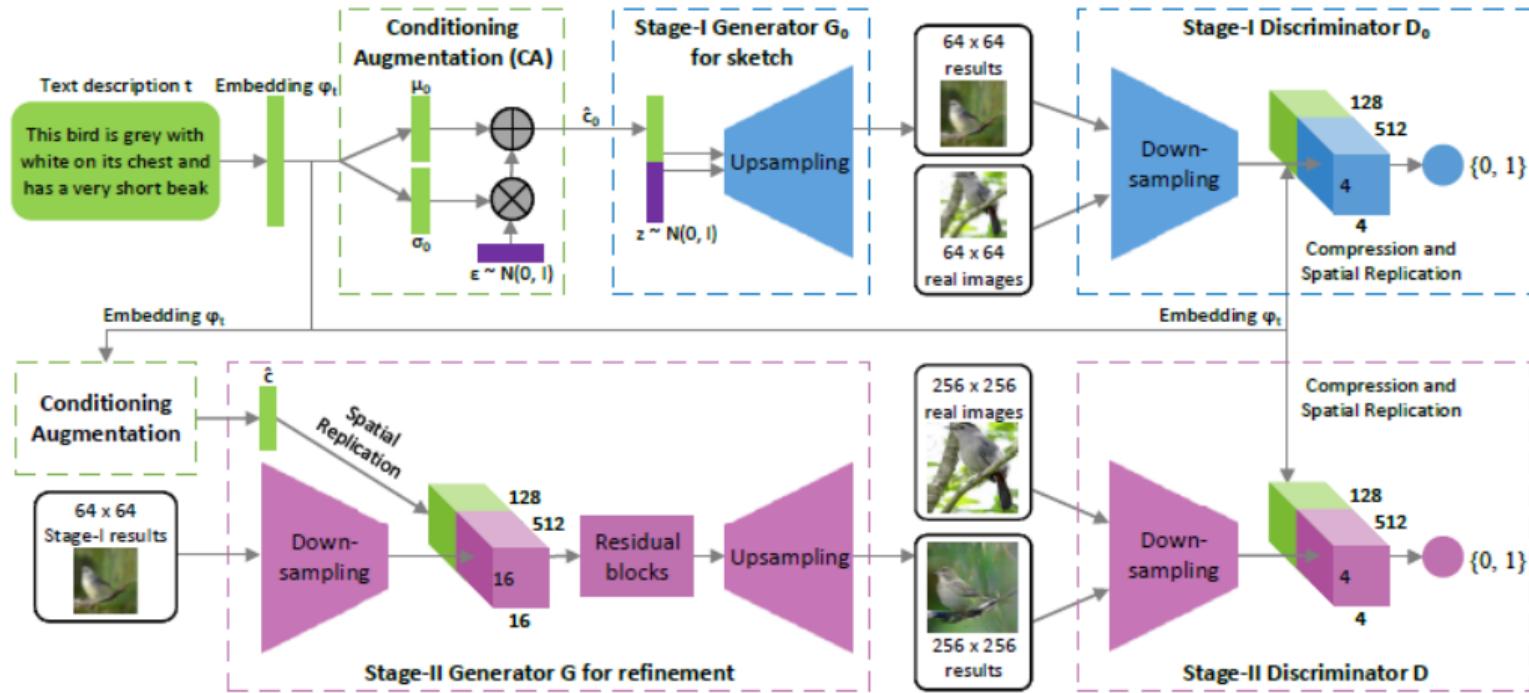
Generate 256×256 photo-realistic images conditioned on text descriptions



- **Stage 1:** Generate 64×64 images, low details
- **Stage 2:** Take Stage 1 output, generate 256×256 , high detail and photo realistic, images
- Both stages conditioned on same textual input

¹Zhang et al, StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, ICCV 2017

StackGAN: Two-stage Network²



²Zhang et al, StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, ICCV 2017

Loss Functions

Scores from Discriminator:

$$s_r \leftarrow D(x, h)$$

{real image, correct text}

$$s_w \leftarrow D(x, \hat{h})$$

{real image, wrong text}

$$s_f \leftarrow D(\hat{x}, h)$$

{fake image, correct text}

Loss Functions

Scores from Discriminator:

$$\begin{aligned}s_r &\leftarrow D(x, h) && \{\text{real image, correct text}\} \\ s_w &\leftarrow D(x, \hat{h}) && \{\text{real image, wrong text}\} \\ s_f &\leftarrow D(\hat{x}, h) && \{\text{fake image, correct text}\}\end{aligned}$$

Then alternate maximizing:

$$\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$$

and minimizing:

$$\mathcal{L}_G \leftarrow \log(1 - s_f) + \lambda D_{KL}(\mathcal{N}(\mu_0(\phi_t), \Sigma_0(\phi_t)) || \mathcal{N}(0, I))$$

StackGAN: Sample Results³

Text description	This flower has petals that are white and has pink shading	This flower has a lot of small purple petals in a dome-like configuration	This flower has long thin yellow petals and a lot of yellow anthers in the center	This flower is pink, white, and yellow in color, and has petals that are striped	This flower is white and yellow in color, with petals that are wavy and smooth	This flower has upturned petals which are thin and orange with rounded edges	This flower has petals that are dark pink with white edges and pink stamen	
64x64 GAN-INT-CLS								
256x256 StackGAN								

³Zhang et al, StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, ICCV 2017; Reed et al, GAN-INT-CLS: Generative Adversarial Text to Image Synthesis, ICML 2016

Progressive GAN⁴



- Generates high-resolution images at 1024×1024 resolution
- **Key idea:** Grow both generator and discriminator progressively

⁴Karras et al, Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018

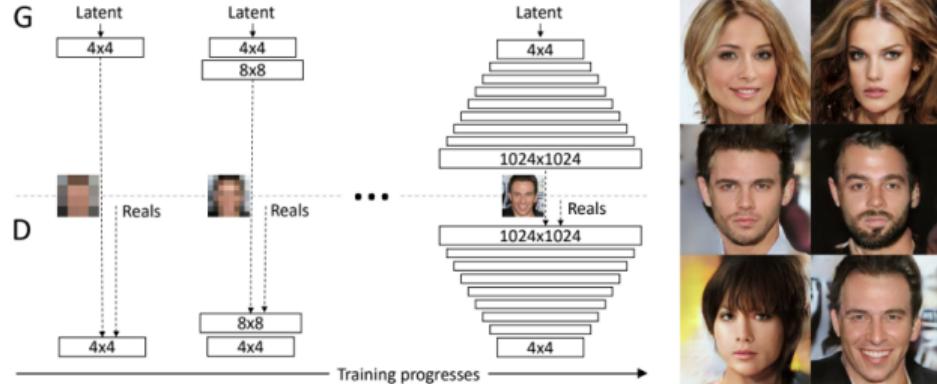
Progressive GAN⁴



- Generates high-resolution images at 1024×1024 resolution
- **Key idea:** Grow both generator and discriminator progressively
- **Other contributions:**
Minibatch standard deviation, equalized learning rate and pixel-wise feature vector normalization in generator

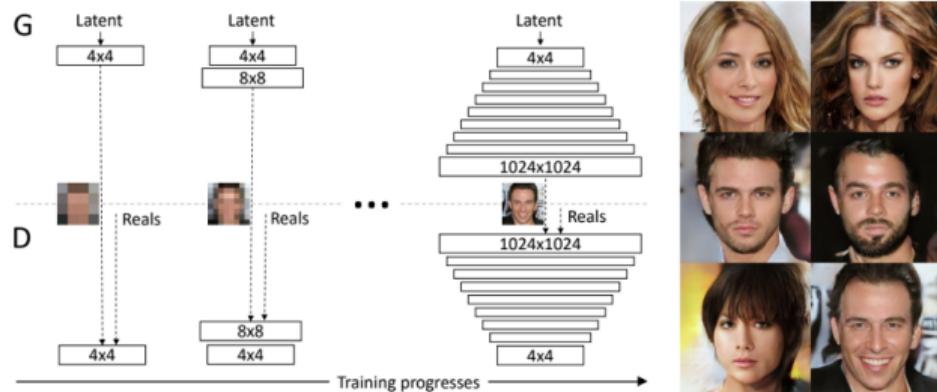
⁴Karras et al, Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018

Progressive GAN: Multi-scale Architecture



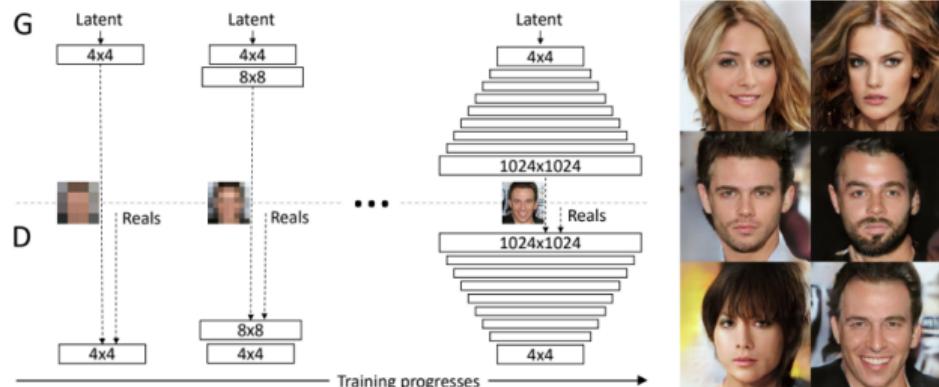
- Generator first produces 4×4 images until this reaches some kind of convergence

Progressive GAN: Multi-scale Architecture



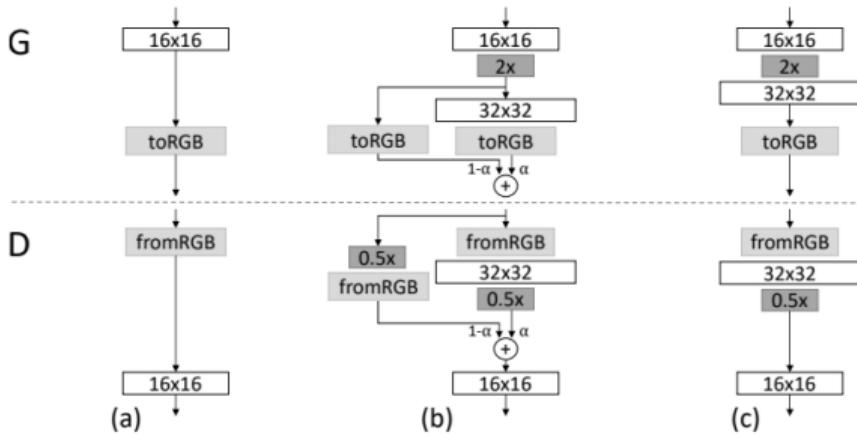
- Generator first produces 4×4 images until this reaches some kind of convergence
- Then task increases to 8×8 images, and so on until 1024×1024

Progressive GAN: Multi-scale Architecture



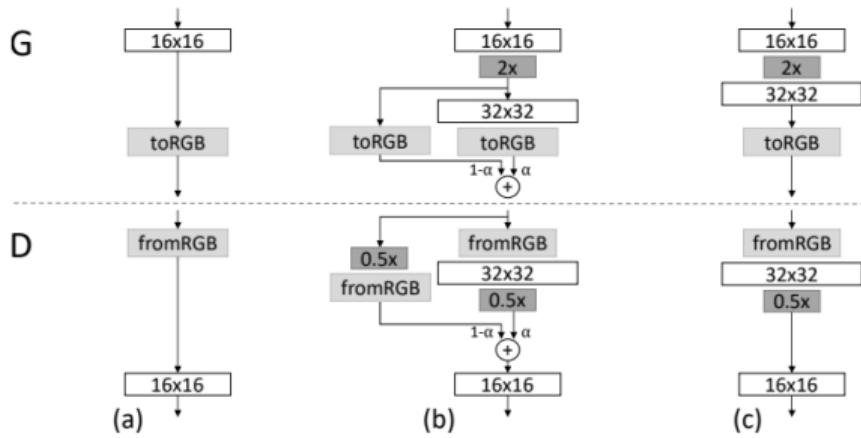
- Generator first produces 4×4 images until this reaches some kind of convergence
- Then task increases to 8×8 images, and so on until 1024×1024
- Allows for stable training of high-resolution images

Progressive GAN: Fading in New Layers



- Methodology similar to ResNets

Progressive GAN: Fading in New Layers



- Methodology similar to ResNets
- In figure (b), generator G:
 - Nearest neighbor interpolation** of upsampled 16×16 layer's output, i.e. 32×32 is added to a 32×32 output layer
 - $\alpha \times \text{new output layer} + (1 - \alpha) \times \text{projected layer}; \alpha \in \{0, 1\}$

Progressive GAN: Other Contributions

- **Minibatch standard deviation:** Standard deviation for each feature in each spatial location over a minibatch computed and averaged; this is concatenated to all spatial locations at a later layer of discriminator. Why?

Progressive GAN: Other Contributions

- **Minibatch standard deviation:** Standard deviation for each feature in each spatial location over a minibatch computed and averaged; this is concatenated to all spatial locations at a later layer of discriminator. Why? [Homework!](#)
- **Equalized learning rate:** $\tilde{w}_i = \frac{w_i}{c}$, where w_i are weights and c is per-layer normalization constant; helps keep weights at similar scale during training

Progressive GAN: Other Contributions

- **Minibatch standard deviation:** Standard deviation for each feature in each spatial location over a minibatch computed and averaged; this is concatenated to all spatial locations at a later layer of discriminator. Why? [Homework!](#)
- **Equalized learning rate:** $\tilde{w}_i = \frac{w_i}{c}$, where w_i are weights and c is per-layer normalization constant; helps keep weights at similar scale during training
- **Pixelwise feature vector normalization in generator G :** Normalize feature vector in each pixel of G after each convolutional layer using:

$$b_{x,y} = \frac{a_{x,y}}{\sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (a_{x,y}^j)^2 + \epsilon}} \quad (1)$$

where $\epsilon = 10^{-8}$, N is number of feature maps, $a_{x,y}$ and $b_{x,y}$ are original and normalized feature vectors in pixel (x, y) respectively

Progressive GAN: Results⁵



Mao et al. (2016b) (128 × 128)

Gulrajani et al. (2017) (128 × 128)

Our (256 × 256)

⁵Karras et al, Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018

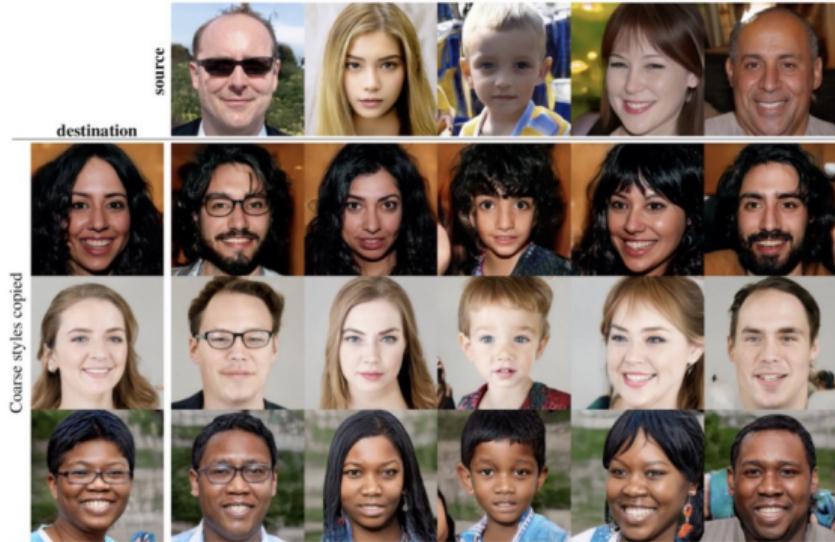
StyleGAN⁶



- ProGAN generates high-quality images, but control of specific features is very limited

⁶Karras et al, A Style-Based Generator Architecture for Generative Adversarial Networks, CVPR 2019

StyleGAN⁶



- ProGAN generates high-quality images, but control of specific features is very limited
- **StyleGAN:** Automatically learned, unsupervised separation of high-level attributes (pose and identity), stochastic variation (hair) and scale-specific control attributes

⁶Karras et al, A Style-Based Generator Architecture for Generative Adversarial Networks, CVPR 2019

How StyleGAN works: Intuition

- **Coarse** resolution of up to 82 - affects pose, general hair style, face shape, etc

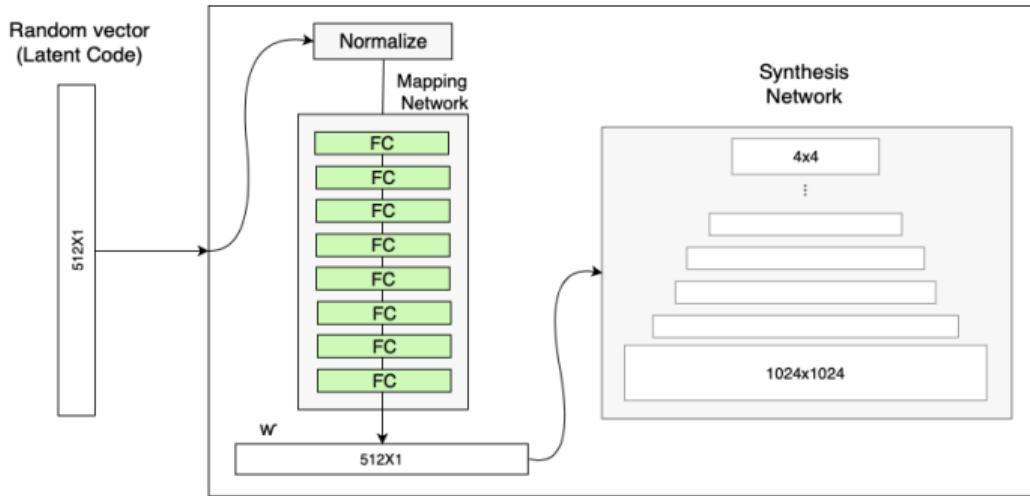
How StyleGAN works: Intuition

- **Coarse** resolution of up to 82 - affects pose, general hair style, face shape, etc
- **Middle** resolution of 162 to 322 - affects finer facial features, hair style, eyes open/closed, etc

How StyleGAN works: Intuition

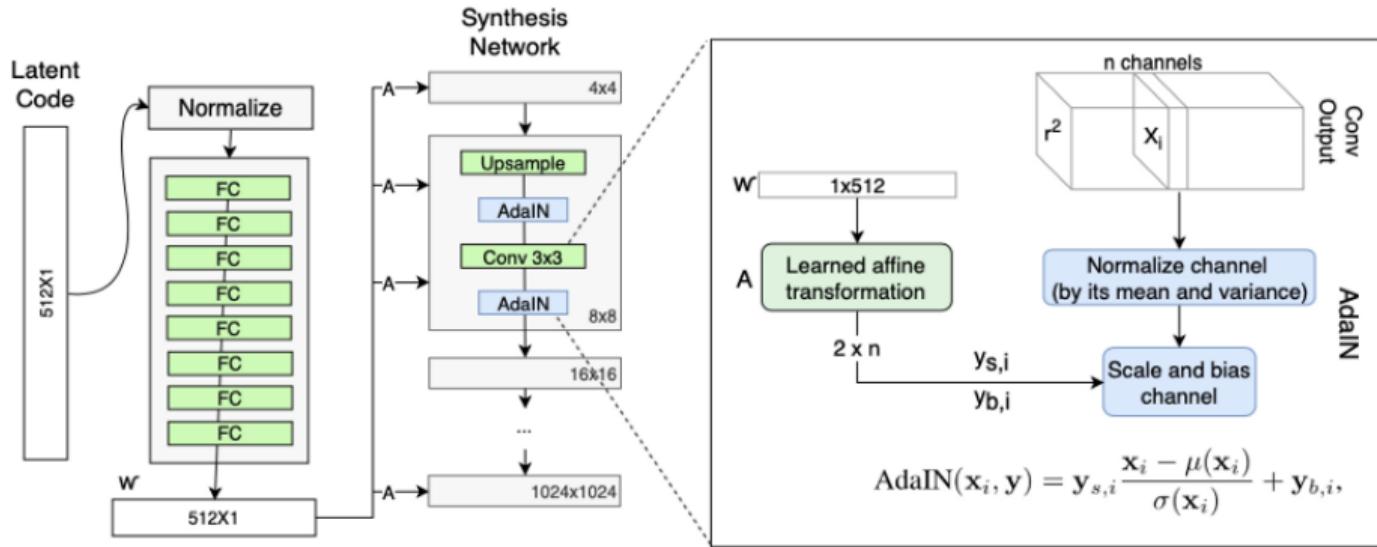
- **Coarse** resolution of up to 82 - affects pose, general hair style, face shape, etc
- **Middle** resolution of 162 to 322 - affects finer facial features, hair style, eyes open/closed, etc
- **Fine** resolution of 642 to 10242 - affects color scheme (eye, hair and skin) and micro features

StyleGAN: Mapping Network



- Encodes input vector into an intermediate vector to control different visual features
- 8 fully connected (FC) layers, output w is of same size as input layer, i.e. 512×1

StyleGAN: Adaptive Instance Normalization



- Transfers w (from mapping net) to generated image
- Module is added to each resolution level of synthesis network, defines the visual expression of features in that level

SPADE⁷

Key Idea

- Previous methods directly feed semantic layout as input to network

⁷Park et al, Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR 2019

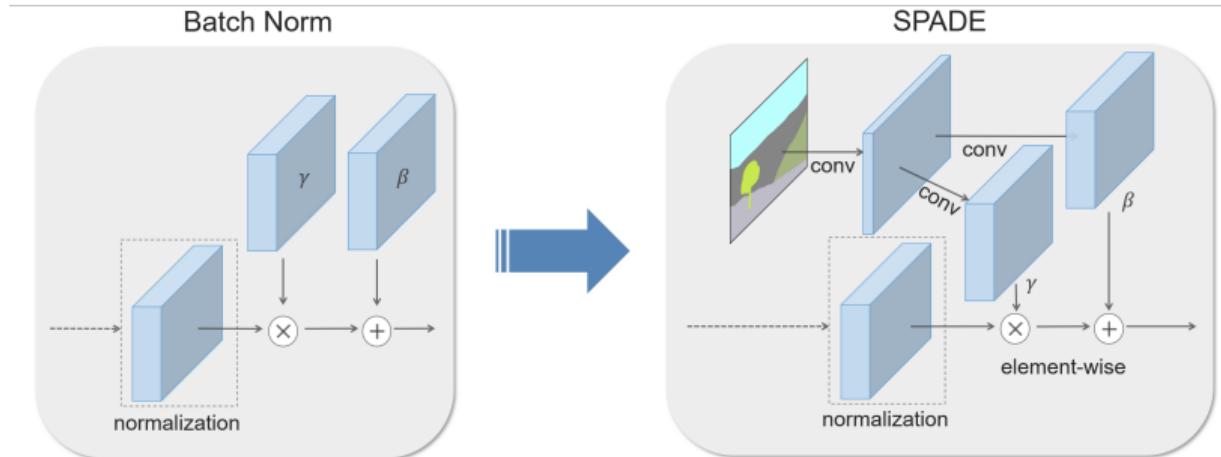
SPADE⁷

Key Idea

- Previous methods directly feed semantic layout as input to network
- **Spatially-adaptive normalization:** Input layout for modulating activations in normalization layers through a spatially-adaptive, learned transformation

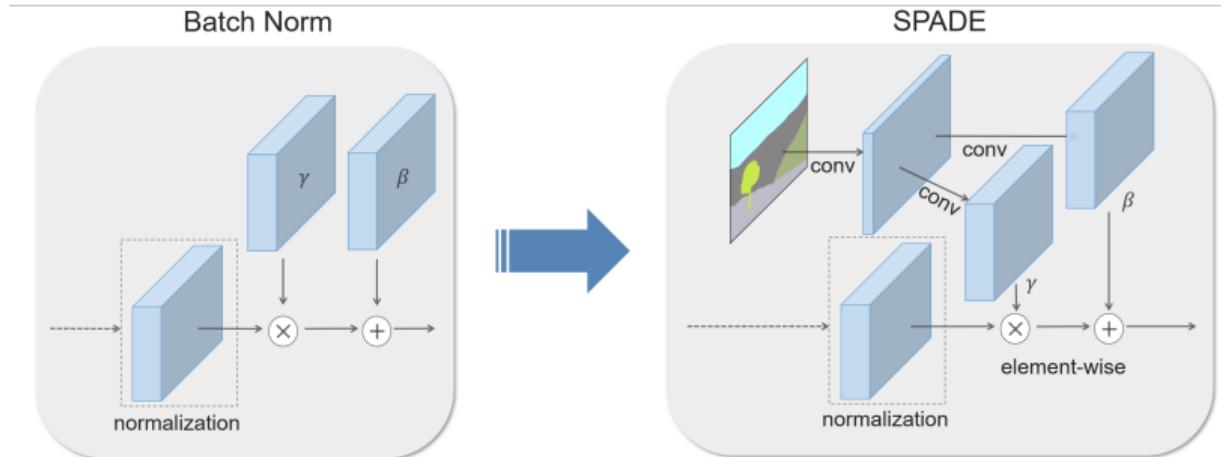
⁷Park et al, Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR 2019

SPADE: Methodology



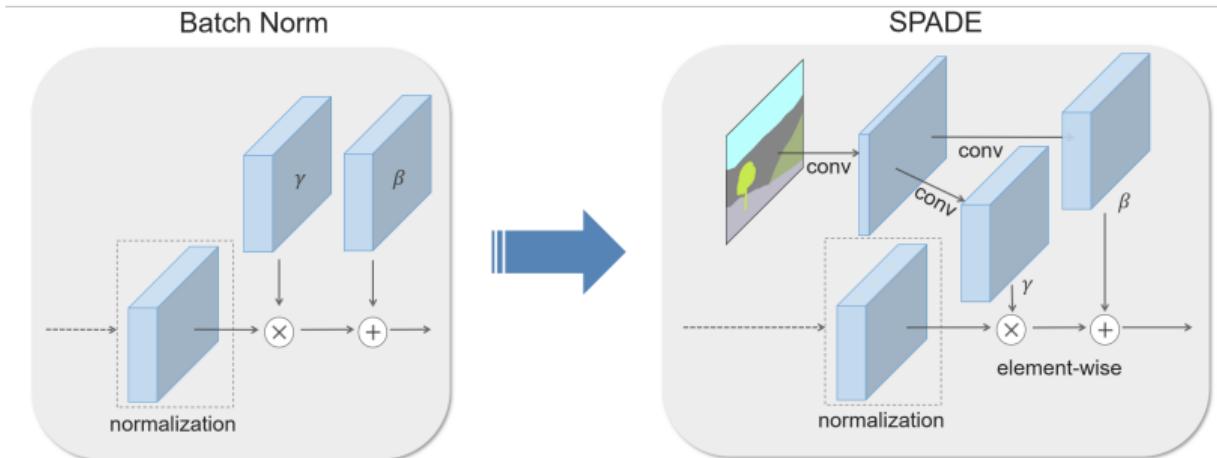
- Batch Normalization gives us affine layers

SPADE: Methodology



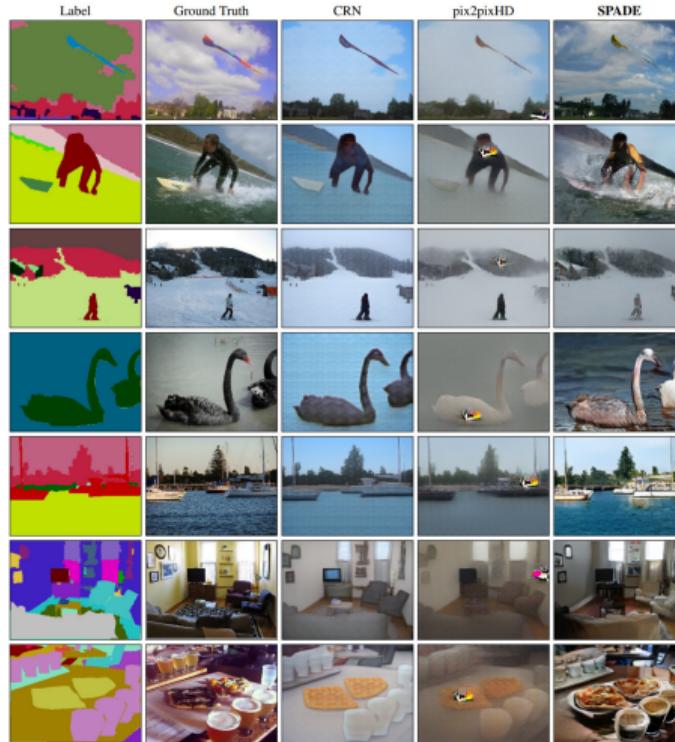
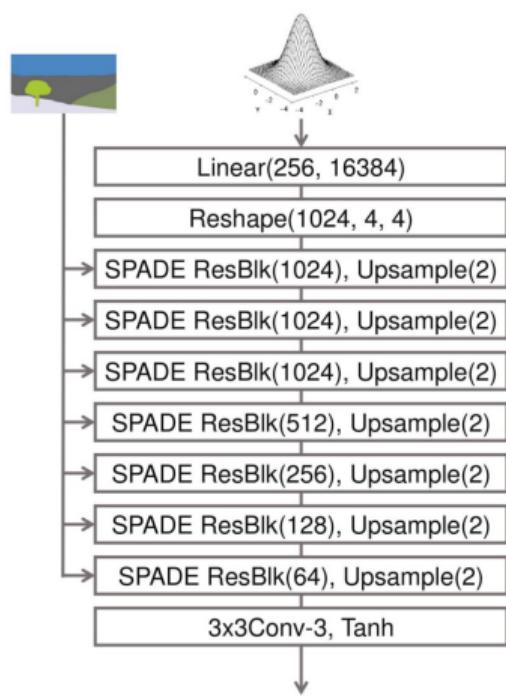
- Batch Normalization gives us affine layers
- In SPADE, affine layer is learned from semantic segmentation map (or any other computer vision task)

SPADE: Methodology



- Batch Normalization gives us affine layers
- In SPADE, affine layer is learned from semantic segmentation map (or any other computer vision task)
- Semantic information is provided via SPADE layers; random latent vector may still be used as input to network, used to manipulate style of generated images

SPADE: Architecture and Results⁸



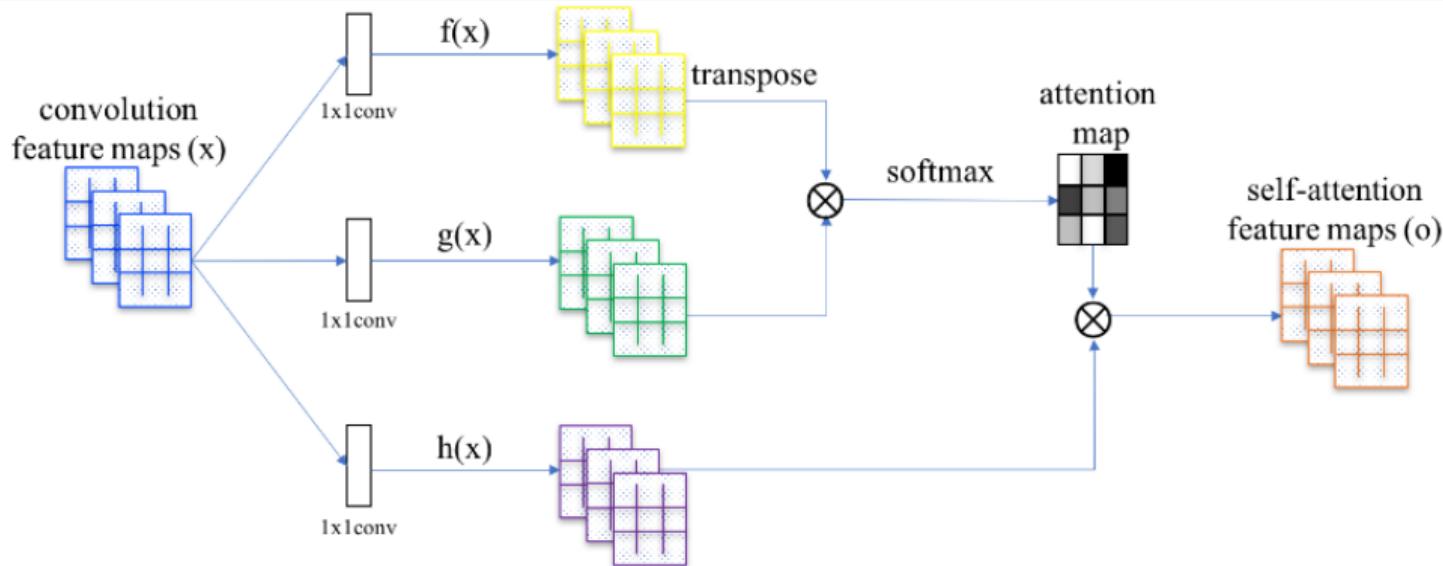
⁸Park et al, Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR 2019

BigGAN⁹

- Intended to scale up GANs for better high-resolution generation
- Designed for class-conditional image generation (generation of images using both a noise vector and class information as input)
- Multiple design decisions to improve generation quality

⁹Brock et al, Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019

BigGAN

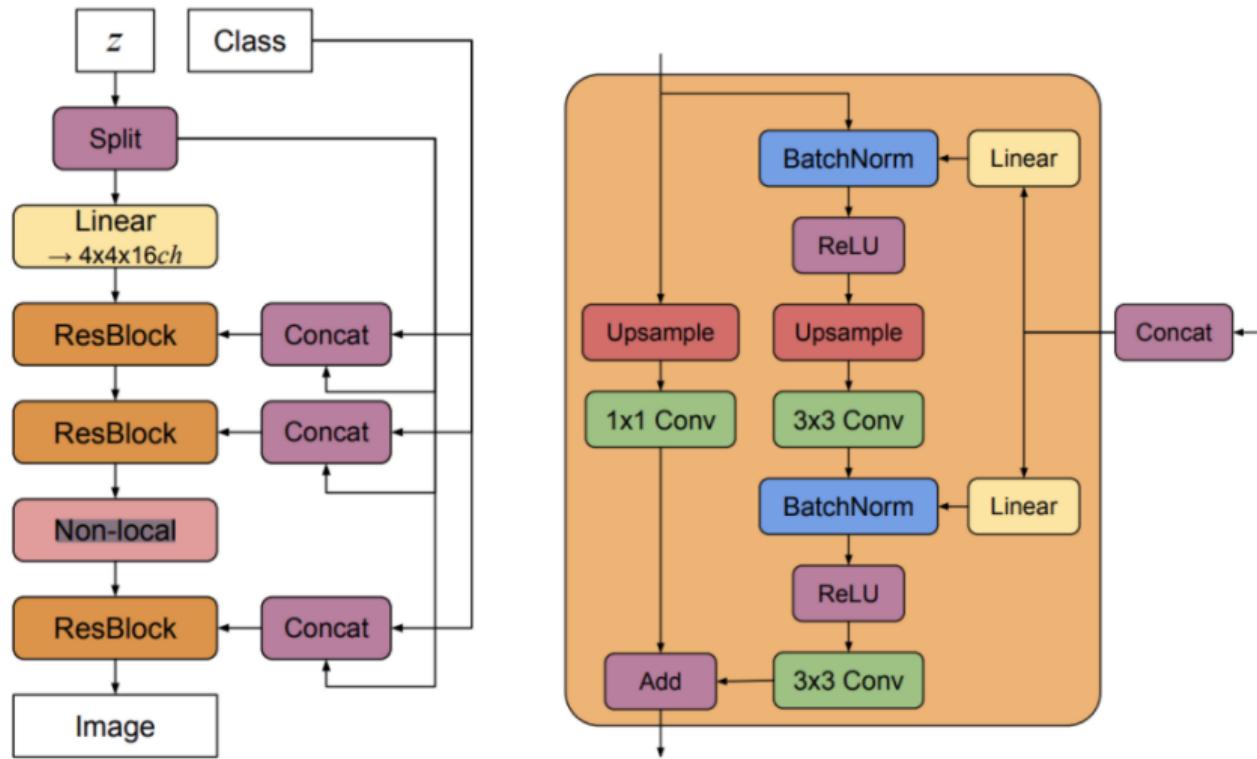


Base model is **Self-Attention GAN (SAGAN)**

Trained using **Hinge Loss**: $\max(0, 1 - t \cdot y)$, where t is target output and y is predicted output

Credit: Zhang et al, Self-Attention Generative Adversarial Networks, ICML 2019

BigGAN: Class-conditional Latents



BigGAN: Other Design Decisions

- **Spectral Normalization:** Normalizes weight matrix W using spectral norm so that it satisfies Lipschitz constraint $\sigma(W) = 1$ (See Miyato et al, Spectral Normalization for Generative Adversarial Networks, ICLR 2018 for details)
- **Orthogonal Weight Initialization:** Initialize weights in each layer to be a random orthogonal matrix (satisfying $W^T W = I$)
- **Skip-z Connections:** Directly connect input latent z to specific layers deep in the network
- **Orthogonal Regularization:** Encourages weights to be orthogonal:
 $R_\beta(W) = \beta \|W^T W - I\|_F^2$. Why? **Homework!**

Credit: Jason Brownlee, MachineLearningMastery.com

BigGAN: Other Tricks/Hacks

- Updates discriminator model twice before updating generator model in each training iteration
- Model weights are averaged across prior training iterations using a moving average (similar to Progressive GAN)
- Large batch sizes of 256, 512, 1024 and 2048 images (best performance at 2048)
- More model parameters: doubled number of channels or feature maps (filters) in each layer
- **Truncation Trick:** Sample from truncated Gaussian (values above a threshold) as input at inference alone

Credit: Jason Brownlee, MachineLearningMastery.com

Homework

Readings

- Wang et al, [Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy](#), arXiv 2020
- Chapter 20 (Deep Generative Models), Deep Learning book
- Code links:
 - Progressive GAN: https://github.com/tkarras/progressive_growing_of_gans
 - StackGAN: <https://github.com/hanzhanggit/StackGAN>
 - StyleGAN: <https://github.com/NVlabs/stylegan>
 - BigGAN: <https://github.com/ajbrock/BigGAN-PyTorch>

Questions

- Minibatch Standard Deviation is used in ProgressiveGAN. Why is this useful?
- Orthogonal Regularization of weights is used in BigGAN. Why is this useful?

References

-  Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).
-  Han Zhang et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5907–5915.
-  Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *International Conference on Learning Representations*. 2018.
-  Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.
-  Taesung Park et al. "Semantic image synthesis with spatially-adaptive normalization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2337–2346.