Jiin Kim

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Karras. T., Aila. T., Laine. S., Lehtinen. J., Progressive Growing of GANs for Improved Quality, Stability, and Variation, *arXiv:1710.10196*, 2017.

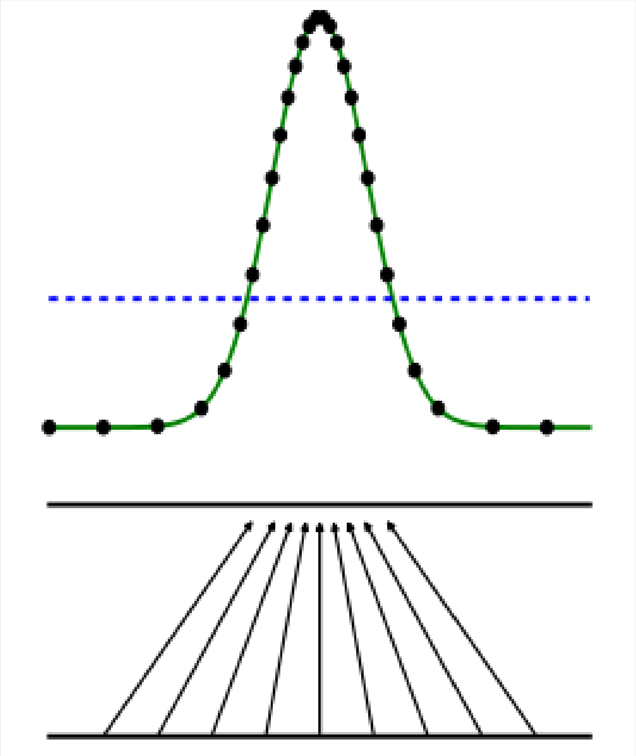
Contents

Introduction

Generative Models - overview

- Autoregressive model (e.g. PixelCNN)
  - Sharp images, slow to evaluate. No latent space
- VAEs
  - Fast to train, burry images
- GANs
  - Sharp images, low resolution, limited variation, unstable training

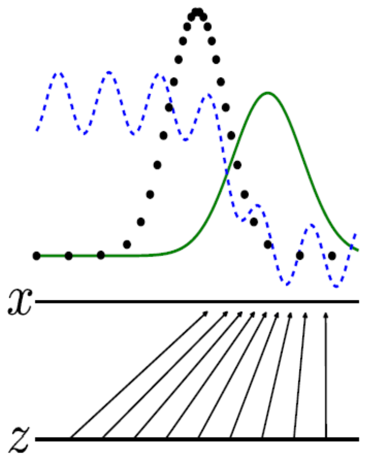
Generative Models - GANs

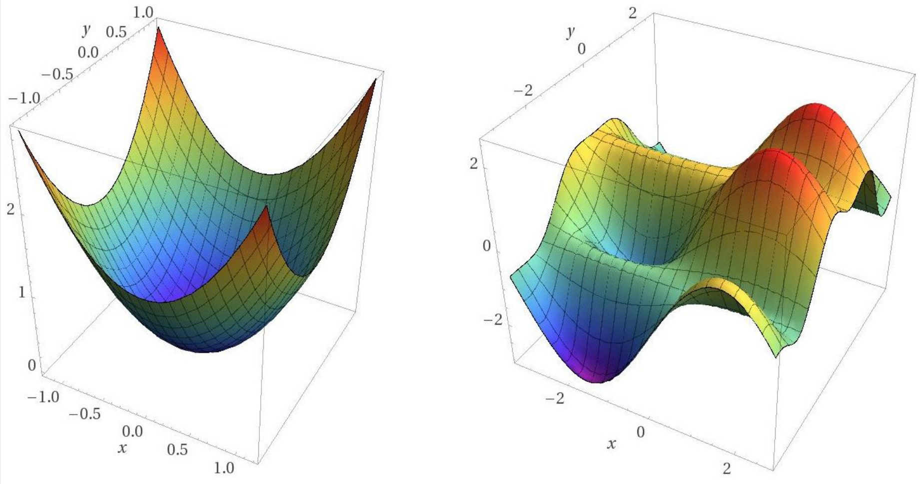


Blue, dashed line: discriminative distribution
Black, dotted line: data distribution (P_{data})
Green solid line: generative distribution (P_g)

Generative Models - GANs

- **Challenge 1**
  - If not much overlap between training and generated distributions then gradients can point in random directions.





Generative Models - GANs

- **Challenge 2**
  - Little variation in results

Plot of 100 GAN Generated MNIST Figures After 100 Epochs

Generative Models - GANs

- **Challenge 3**
  - High resolution harder because easier to tell apart

Low Resolution

High Resolution

Generative Models - GANs

- **Challenge 4**
  - High resolution requires smaller minibatches so training less stable

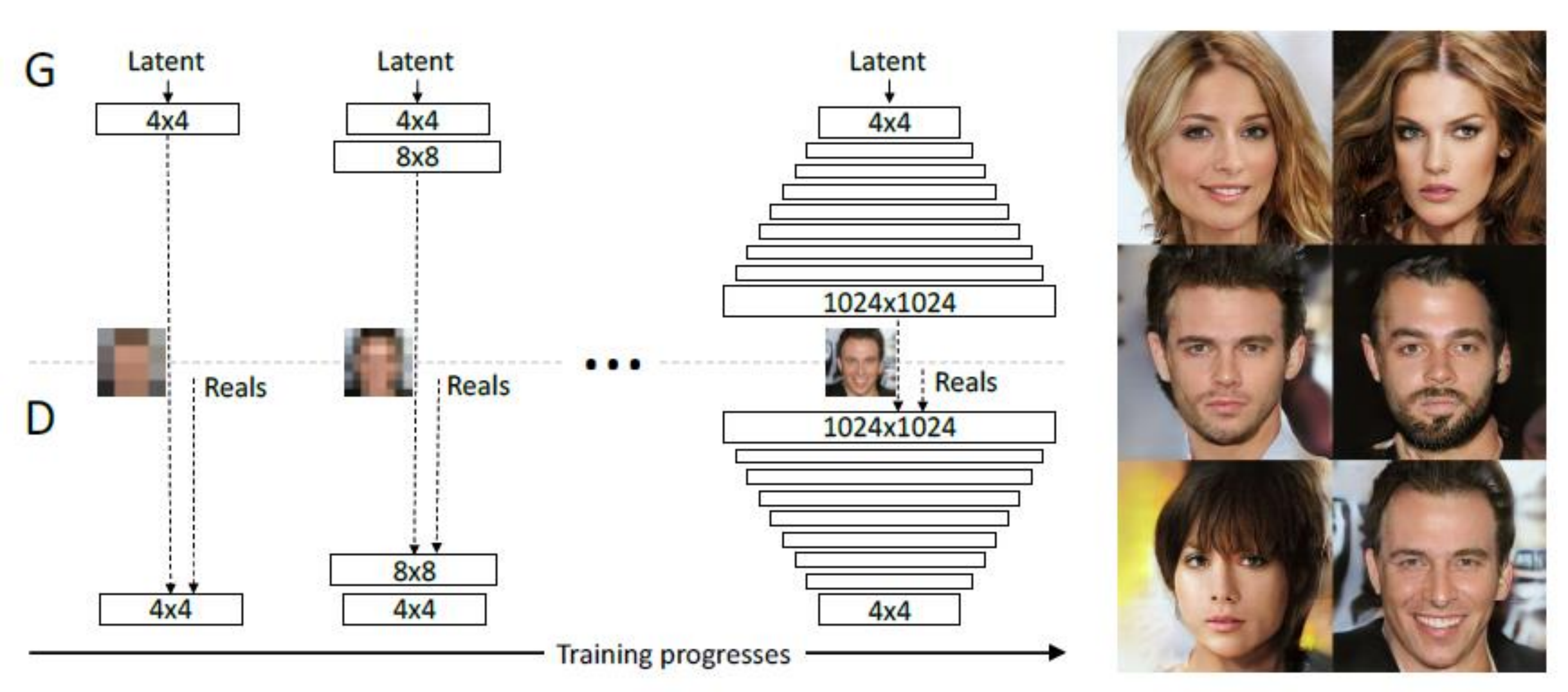
High Resolution

PGGAN - overview

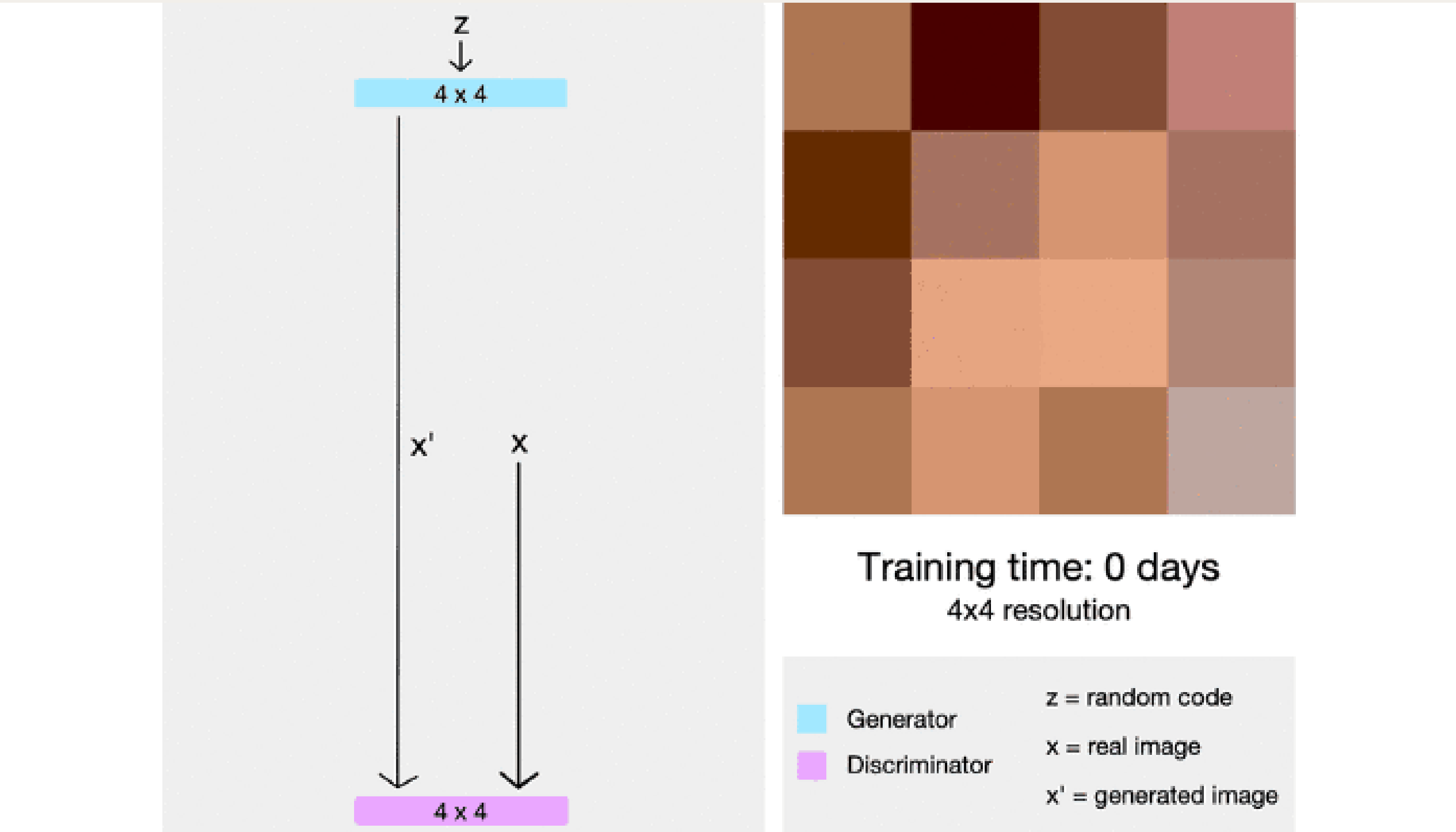
- Key insight
  - we can **grow both the generator and discriminator progressively**,
  - starting from easier **low-resolution images**,
  - and **add new layers** that introduce higher-resolution details as the training progress

Progressive Growing of GANs

PGGAN - growing the GAN

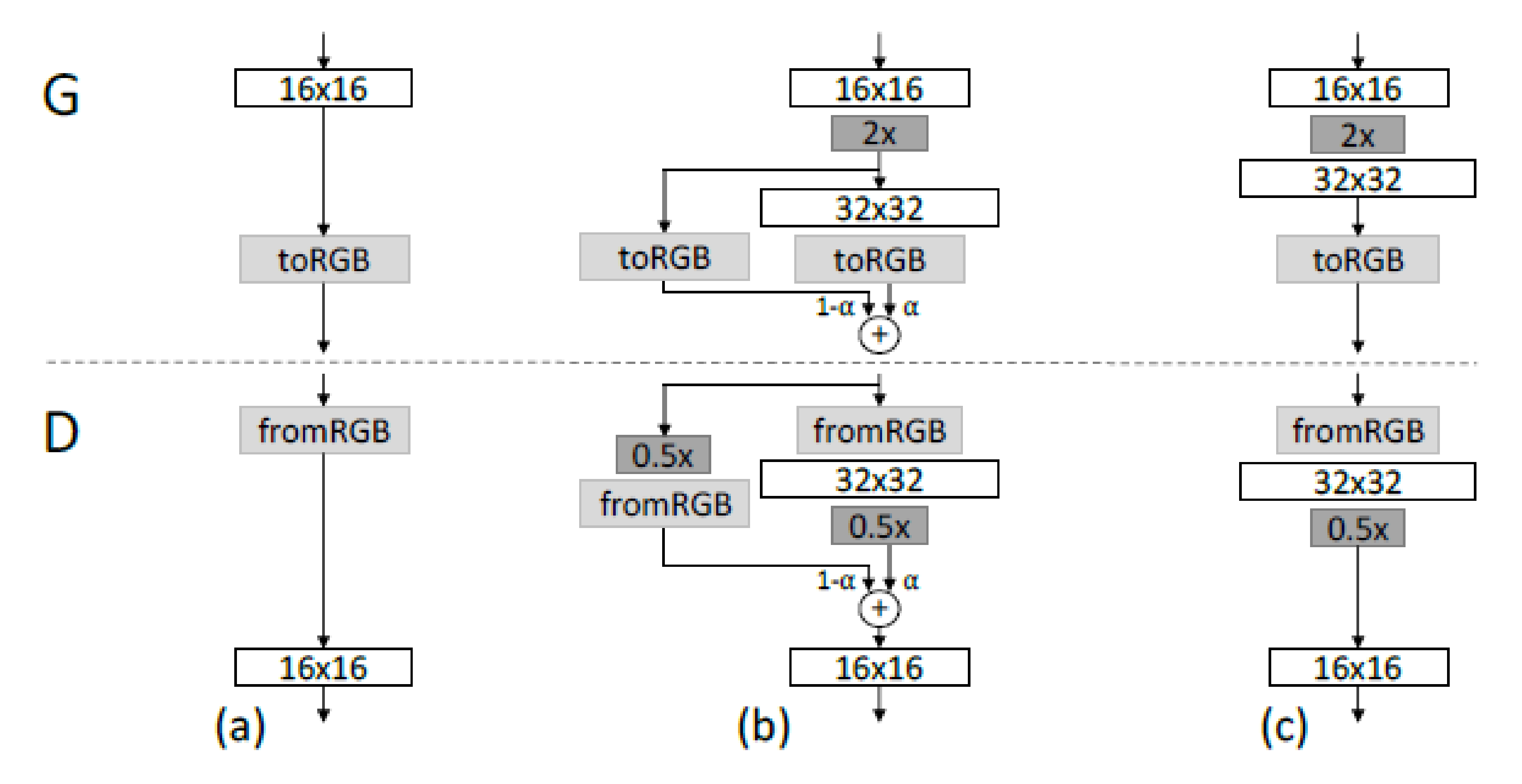


PGGAN - growing the GAN



PGGAN - fading in higher resolution layers

Fade in smoothly



Loss Function

- The author’s say their work is **independent of loss function**
- Do experiments with both WGAN-GP and LSGAN

PGGAN - benefits

- Training **avoids high resolution problem** of too much divergence early on
- **Faster** training, 2-6 x faster
- Only use a **single GAN** instead of a hierarchy of GANs
- More **stable training** - more steps done at lower resolution with larger minibatches

Increasing Variation using Minibatch Standard deviation

Minibatch standard deviation

- **Minibatch discrimination** (Salimans et al. 2016)
  - Compute feature statistics **across the minibatch**
  - Encourage the minibatches of generated and training images to show similar statistics
  - Add a **minibatch layer** towards the end of the discriminator

Minibatch standard deviation

- **Minibatch standard deviation**
  - **Simplifies** the minibatch discrimination and **improves variation**
  - How to compute ?
    - Compute standard deviation for each feature in each spatial location
    - Then average over all features and spatial locations to get a single value
    - Replicate the value and concatenate it to all spatial locations and over the minibatch, yields one additional feature map

Normalization in Generator and Discriminator

Equalized Learning Rate

- Use trivial **N(0, 1)** weight initialization and **scale weights** at runtime
- $\hat{w}_i = w_i / c$
  - w_i: weights, c: per-layer normalization constant from He's initializer
- Adaptive SGD methods normalize a gradient update by its estimated standard deviation
- If some parameters have a **larger dynamic range** than others, they will take **longer to adjust**
- Our approach ensures that the **dynamic range**, and thus the **learning speed**, is the **same for all weights**

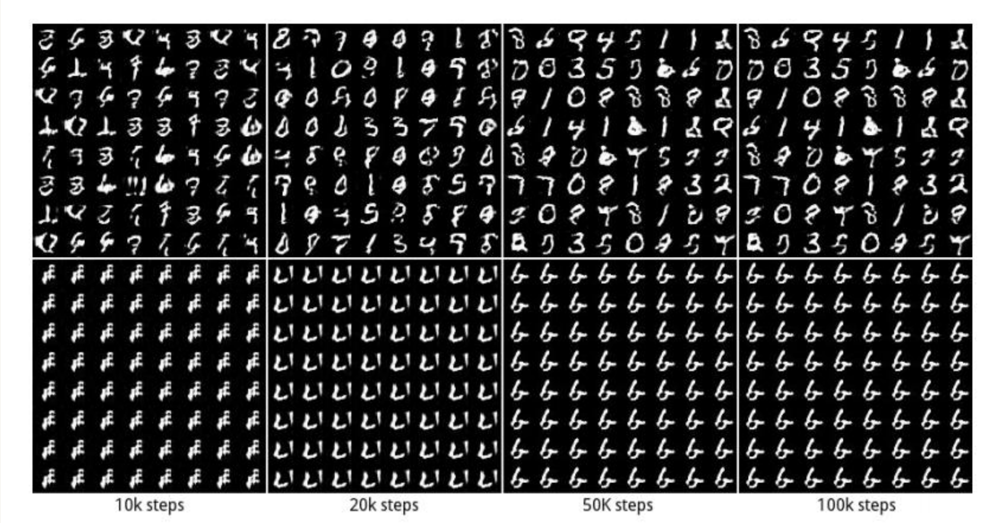
Pixelwise Feature Vector Normalization in Generator

- Normalize the feature vector in each pixel to unit length in the generator after each convolutional layer
- Prevent feature map magnitudes from getting too large

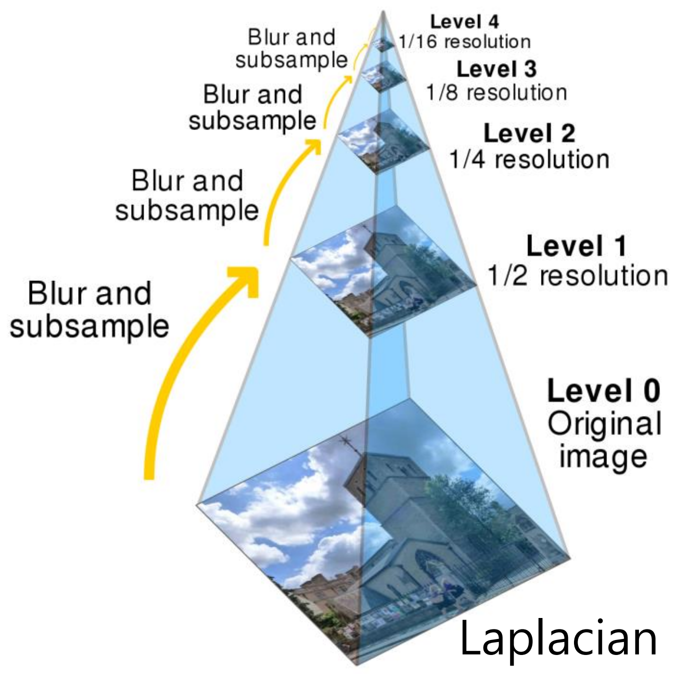
$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (a_{x,y}^{j})^2 + \epsilon}, \text{ where } \epsilon = 10^{-8}$$

Multi-Scale Statistical Similarity for Assessing GAN Results

- **MS-SSIM**: Good at identifying global mode collapse, **not good for local mode collapse** like on colors and textures
- Do MS-SSIM on local patches drawn from **Laplacian pyramid** representations of generated and target images



Mode collapsing



Laplacian pyramid

Multi-Scale Statistical Similarity for Assessing GAN Results

- Sample 16,384 images and extract 128 descriptors from each level of the Laplacian pyramid. Each descriptor is a 7x7 pixel neighborhood with 3 color channels
- Compute **sliced Wasserstein distance** between samples. **Smaller distance** means that at that level of resolution training images and generator samples have **similar variation**

Experiments

Importance of Individual Contributions in Teams of Statistical Similarity

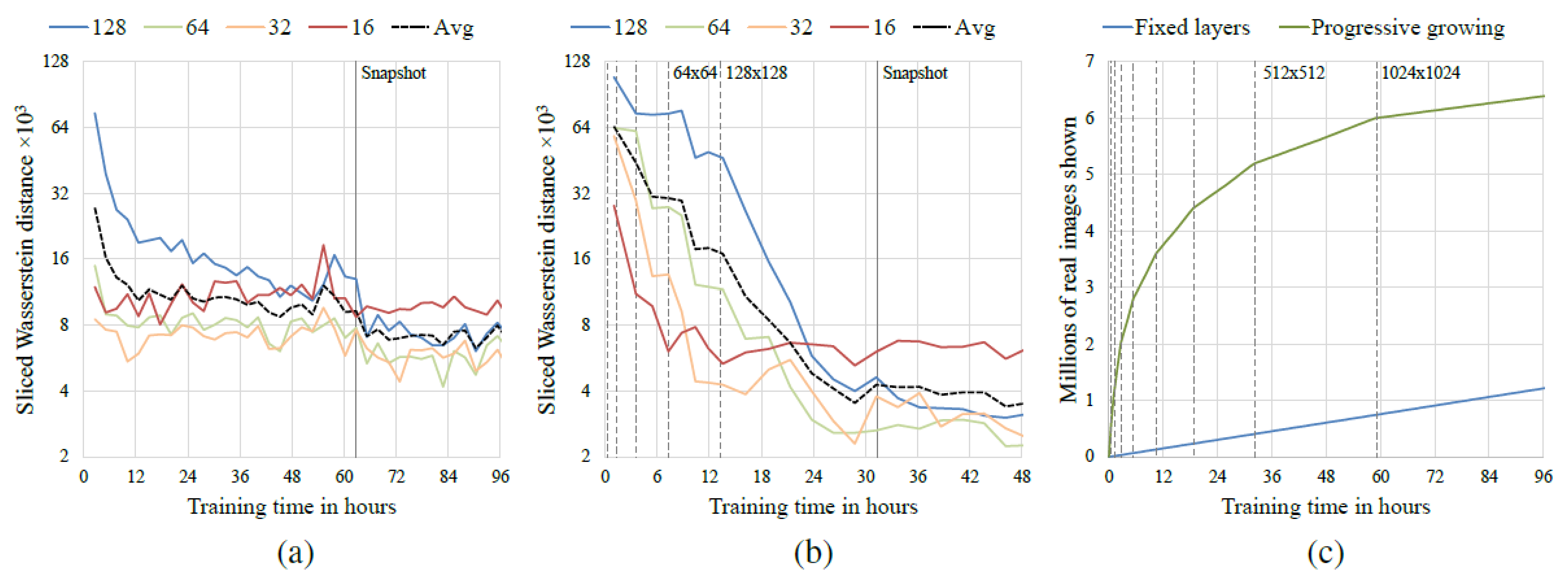
Training configuration	CelebA						LSUN bedroom					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

Importance of Individual Contributions in Teams of Statistical Similarity

Training configuration	CELEBA Sliced Wasserstein distance ×10^3 128	64	32	16	Avg	CELEBA MS-SSIM	LSUN BEDROOM Sliced Wasserstein distance ×10^3 128	64	32	16	Avg	LSUN BEDROOM MS-SSIM
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

(a) (b) (c) (d) (e∗) (e) (f) (g) (h) Converged

Convergence and Training Speed



High-Resolution Image Generator using CelebA-HQ Dataset

High-Resolution Image Generator using CelebA-HQ Dataset

LUSN Results

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

LUSN Results

Q & A