11-695: AI Engineering
GAN III

LTI/SCS

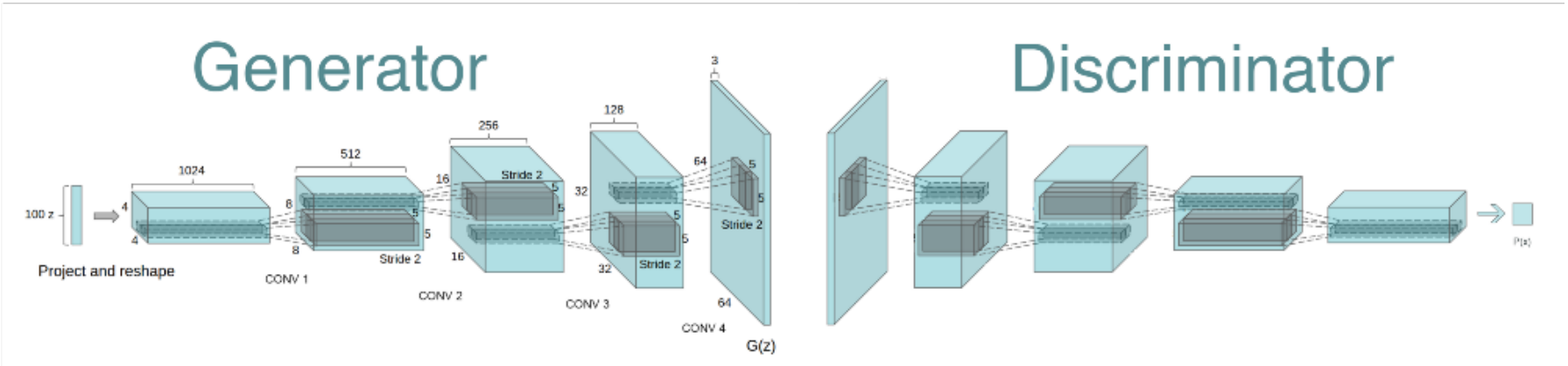
Spring 2020

Outline

Improving Architectures

- Deep Convolutional (DCGAN)
- Stacked GAN (SGAN)
- Progressive GAN

DCGAN[1]



- Strided Convolutional instead of Max Pooling
- Use TransposeConv for upsampling
- No FC layer
- ReLU for G (except last layer uses Tanh), Leaky ReLU for D
- Use Batch normalization BN.

[1] https://arxiv.org/pdf/1511.06434.pdf

Image credit: Alec Radford & Luke Metz & Soumith Chintala 2016, mxnet.io

Recall: DCGAN: Generator

Carnegie Mellon

dcgan_generator.py

```
1  model = tf.keras.Sequential()
2  model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
3  model.add(layers.BatchNormalization())
4  model.add(layers.LeakyReLU())
5
6  model.add(layers.Reshape((7, 7, 256)))
7  assert model.output_shape == (None, 7, 7, 256) # Note: None is the batch size
8  model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
9                                   padding='same', use_bias=False))
10 assert model.output_shape == (None, 7, 7, 128)
11 model.add(layers.BatchNormalization())
12 model.add(layers.LeakyReLU())
13
14 model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
15                                  padding='same', use_bias=False))
16 assert model.output_shape == (None, 14, 14, 64)
17 model.add(layers.BatchNormalization())
18 model.add(layers.LeakyReLU())
19
20 model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2),
21                                  padding='same', use_bias=False, activation='tanh'))
```

- No FC, BatchNorm and LeakyReuLU and Tanh

Recal: DCGAN: Discriminator

Carnegie Mellon

dcgan_discrminator.py

```
1  model = tf.keras.Sequential()
2  model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
3                          input_shape=[28, 28, 1]))
4  model.add(layers.LeakyReLU())
5  model.add(layers.Dropout(0.3))
6
7  model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
8  model.add(layers.LeakyReLU())
9  model.add(layers.Dropout(0.3))
10
11 model.add(layers.Flatten())
12 model.add(layers.Dense(1))
```

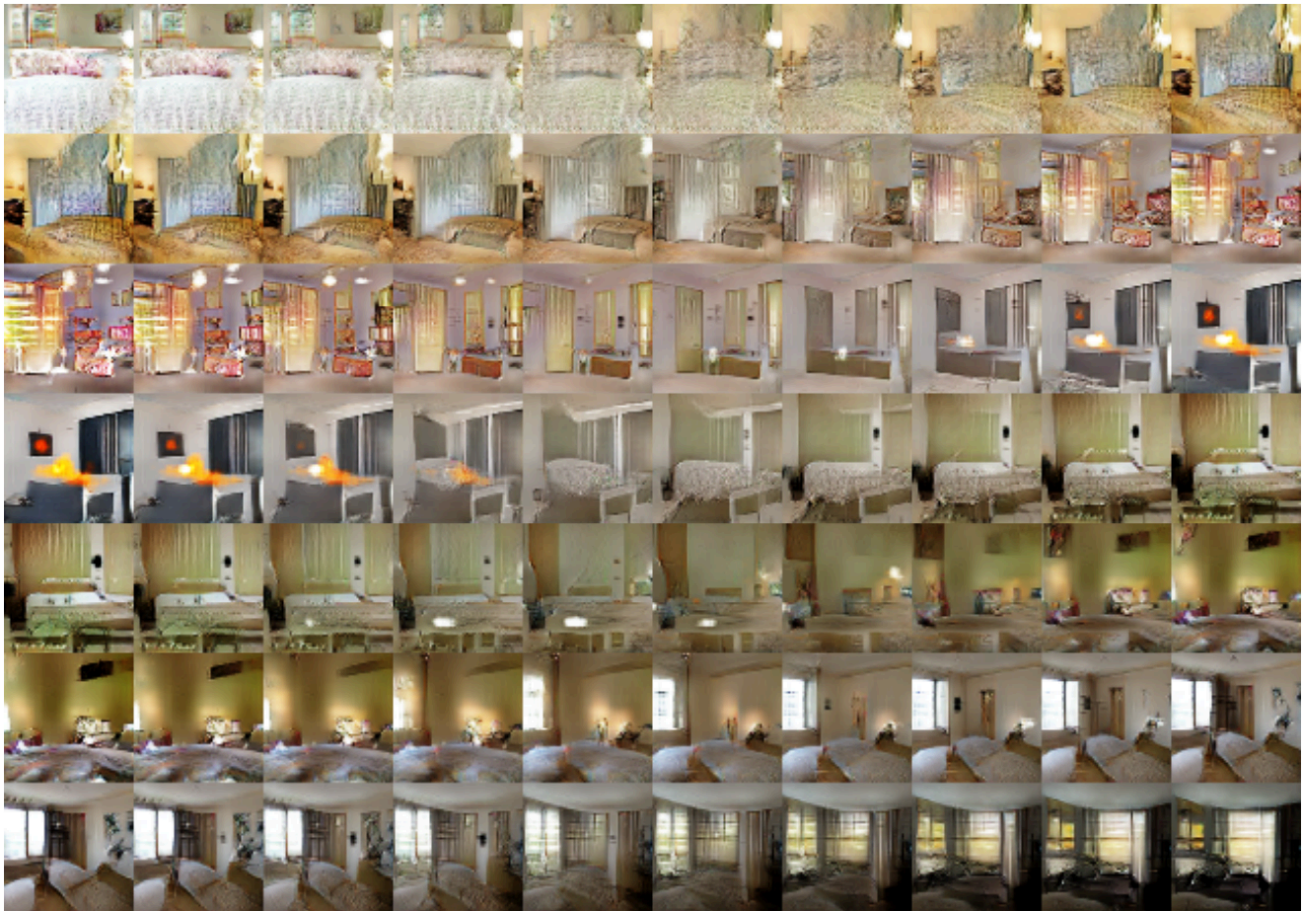
- No intermediate FC
- Have Dropout and Leaky ReLU
- Play more with ▸ code

DCGAN Results

- Mark an important milestone in GAN development
- Still a simple yet popular and widely-adopted one

Image credit: Alec Radford & Luke Metz & Soumith Chintala 2016

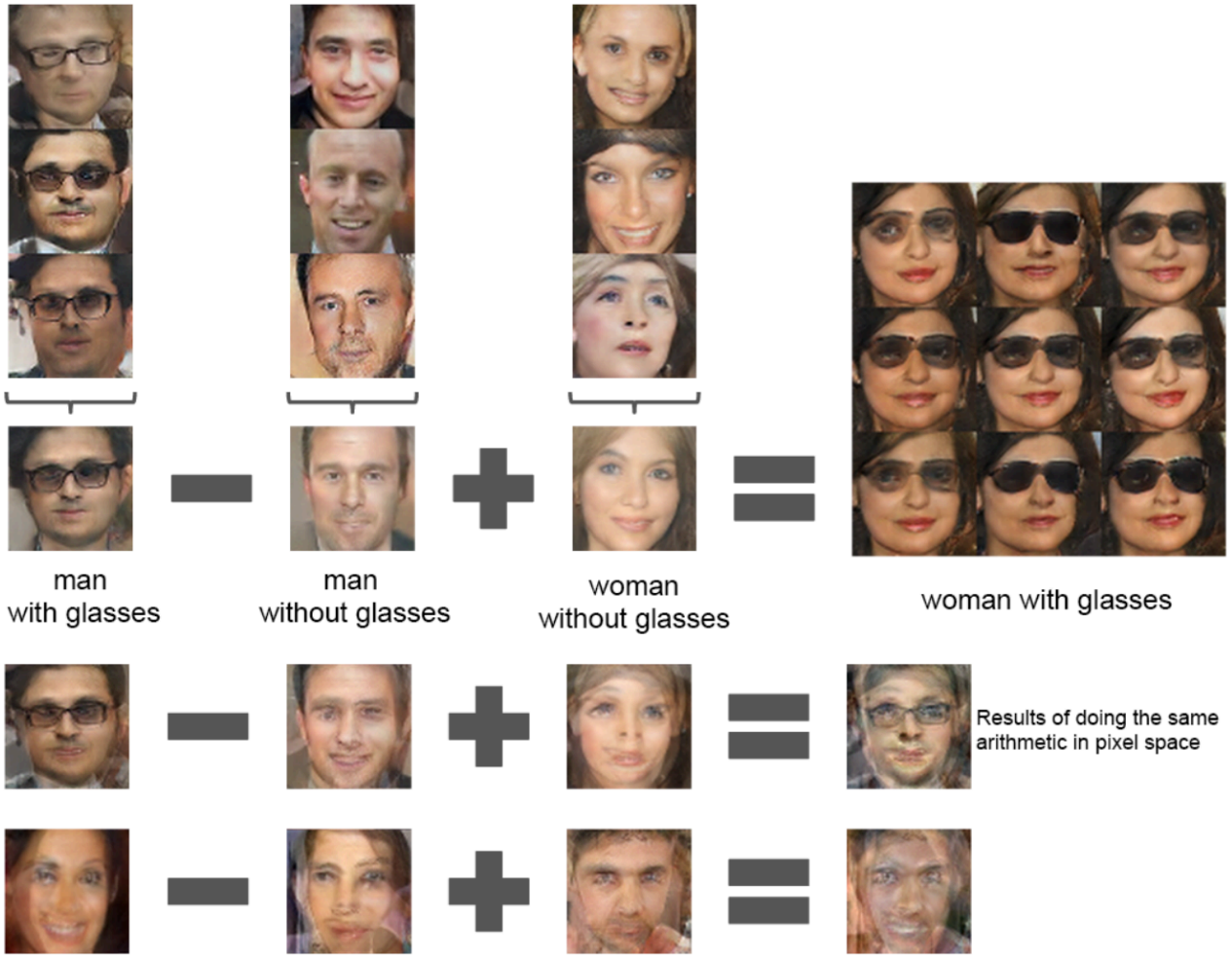
DCGAN Transition



- Transition is smooth
- *E.g.* 6th row: the TV changed smoothly to the window

Image credit: Alec Radford, Luke Metz and Soumith Chintala 2016

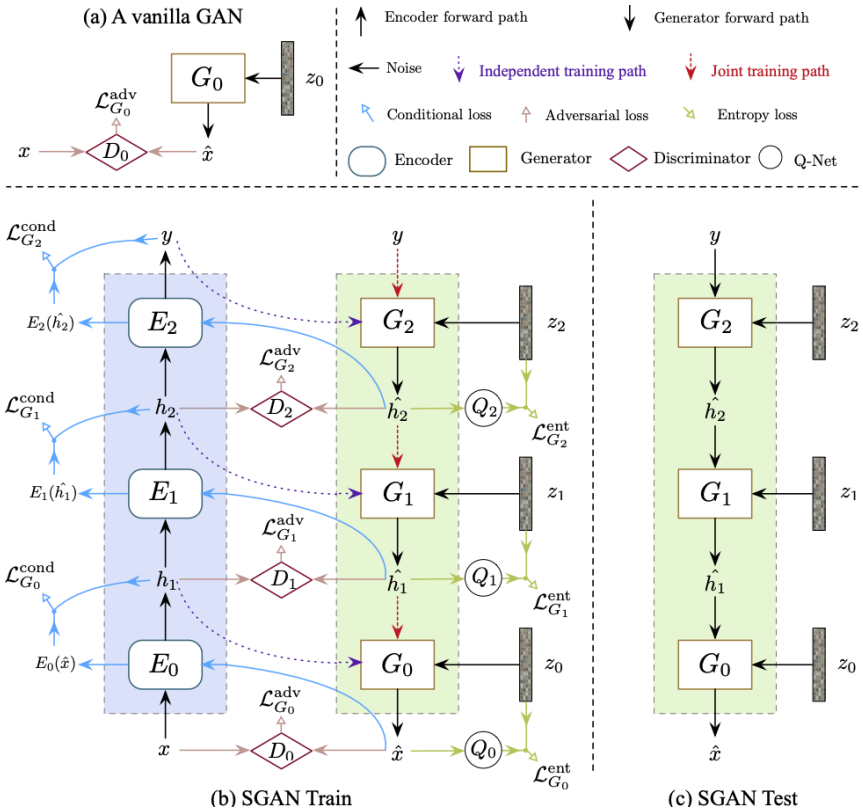
DCGAN Interpolation



- Similar to word2vec interpolation

Image credit: Alec Radford, Luke Metz and Soumith Chintala 2016

Stacked GAN (SGAN[2])



- Stack of GANs and help from stacked pre-trained Encoders
- Train them sequentially (hierarchically) and then train the whole

[2] https://arxiv.org/pdf/1612.04357.pdf

Image credit: Xun Huang *et al.* 2017

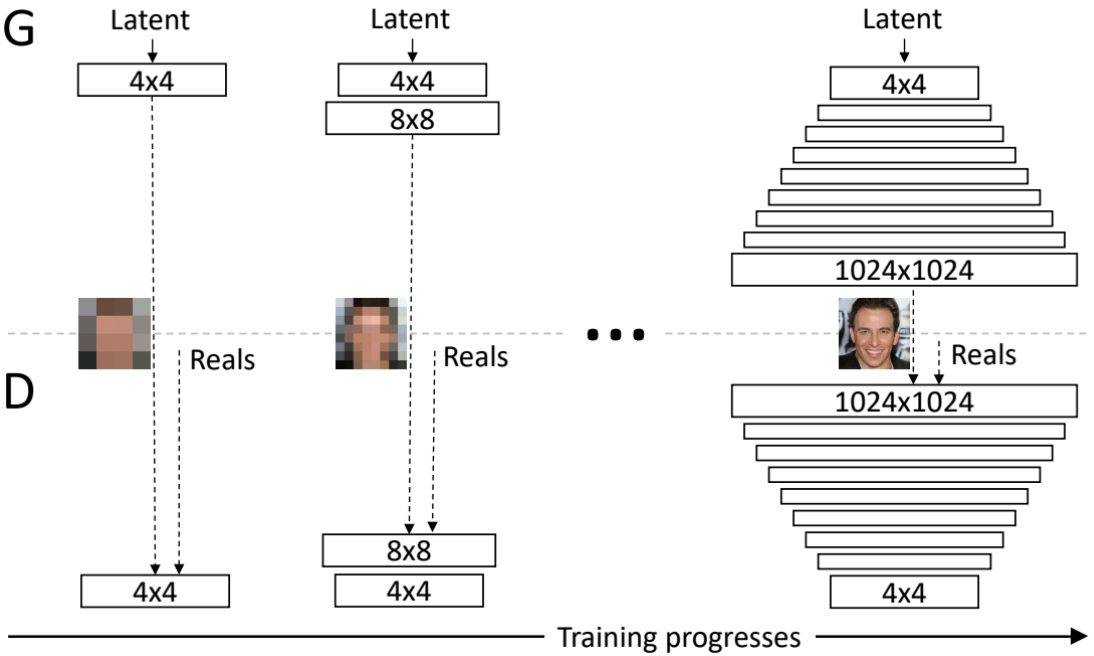
Stacked GAN (SGAN) Results

Carnegie Mellon

- Sharp and fine results

Image credit: Xun Huang *et al.* 2017

Progressive GAN[3]



- Different appoach from SGAN: train the whole model
- But improve the model over time → better resolution

[3] https://arxiv.org/pdf/1710.10196.pdf

Image credit: Karras *et al.* 2017

Progressive GAN

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

Image credit: Karras *et al.* 2017

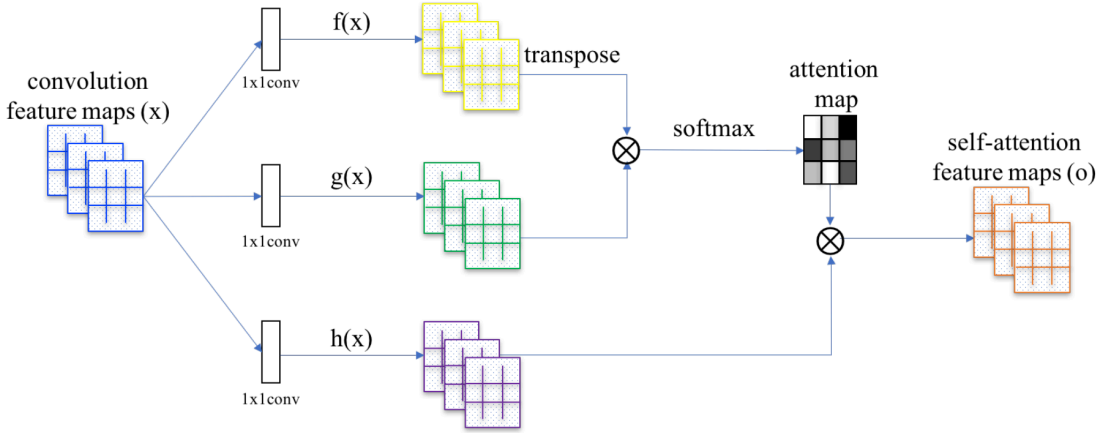


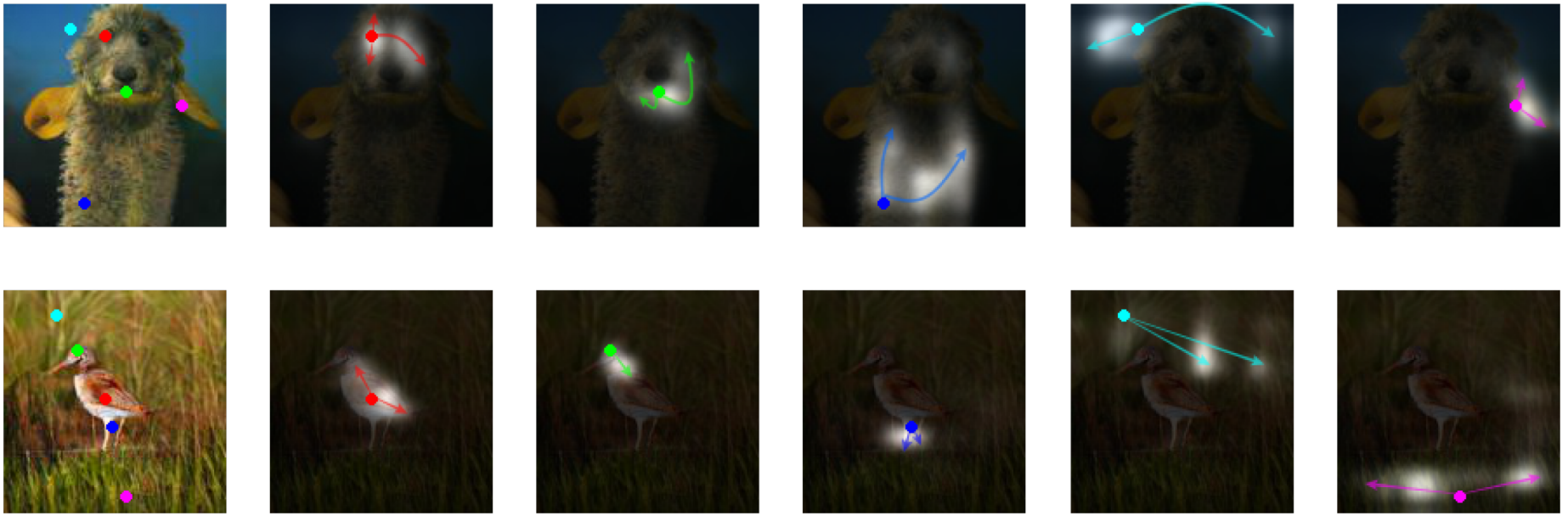
Figure 2: The proposed self-attention mechanism. The $\otimes$ denotes matrix multiplication. The softmax operation is performed on each row.

- Image $x \in \mathbb{R}^{C*N}$ transformed to $f(x) = W_f x,\ g(x) = W_g x$
- Attention score: $\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{k=1}^{N} \exp(s_{kj})}$ where $s_{kj} = f(x_k)^T g(x_j)$
- Output is $o = (o_1, o_2, ..., o_N) \in \mathbb{R}^{C*N}$ where $o_j = \sum_{k=1}^{N} \beta_{j,i} h(x_i)$, with $h(x_i) = W_h x_i$
- Shapes: $W_g \in \mathbb{R}^{\bar{C}*C}, W_f \in \mathbb{R}^{\bar{C}*C}, W_h \in \mathbb{R}^{C*C}$

[4] https://arxiv.org/pdf/1805.08318.pdf

Image credit: Han Zhang *et al.* 2018

SAGAN: Attention Map



- Self-Attention for both D and G
- Generation leverages cues from all feature locations (instead of local)

Image credit: Han Zhang *et al.* 2018

SAGAN: Results

Figure 4: 128×128 examples randomly generated by the baseline model and our models "SN on *G*/*D*" and "SN on *G*/*D*+TTUR".

- Mark a milestone: able to generate full ImageNet samples

Image credit: Han Zhang *et al.* 2018

Improving Losses

- Recall the problem: Zero-sum game loss is nice in theory, but unfortunately not (yet) in practice for training GAN
- Possible solutions:
  - Make the loss non-zero-sum
  - Use Wasserstein distance (WGAN)
  - Latent space optimization (LOGAN)
  - Use other losses

Zero-sum loss problem: Revisited

Carnegie Mellon

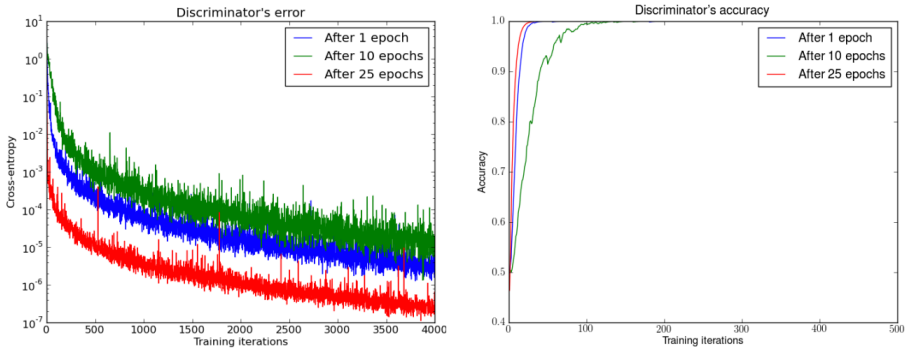


Figure 1: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch. We see the error quickly going to 0, even with very few iterations on the discriminator. This even happens after 25 epochs of the DCGAN, when the samples are remarkably good and the supports are likely to intersect, pointing to the non-continuity of the distributions. Note the logarithmic scale. For illustration purposes we also show the accuracy of the discriminator, which goes to 1 in sometimes less than 50 iterations. This is 1 even for numerical precision, and the numbers are running averages, pointing towards even faster convergence.

- Recall for G_ψ: $\nabla_\psi = \frac{1}{m}\sum_{i=1}^{m}[\log(1 - D_\theta(G_\psi(z^{(i)})))]$ tends to be zero at beginning $\rightarrow$ G learns nothing
- Now for D_θ:
  - When it gets bad, no useful feeback: G_ψ learns incorrectly
  - When it improves, D's loss will go to zero quickly

Image credit: Martin Arjovsky and Leon Bouttou 2017

Zero-sum loss problem: Revisited

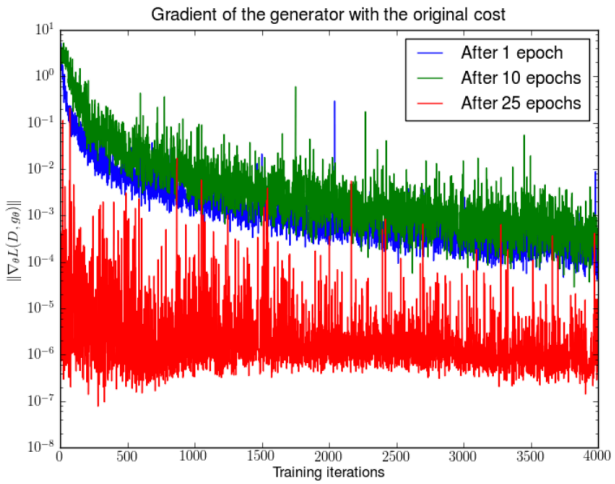


Figure 2: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the original cost function. We see the gradient norms decay quickly, in the best case 5 orders of magnitude after 4000 discriminator iterations. Note the logarithmic scale.

- Now for D_θ:
  - When it's perfect, G_ψ has no useful feedback to correct itself
  - And its gradients will also go to zero quickly $\rightarrow$ learns slowly to learns nothing
  - When it improves, D's loss will go to zero quickly,
  - and G's gradients will go to zeros quickly as well! Very bad!

Image credit: Martin Arjovsky and Leon Bouttou 2017

Fix: Make the loss non-zero-sum

Carnegie Mellon

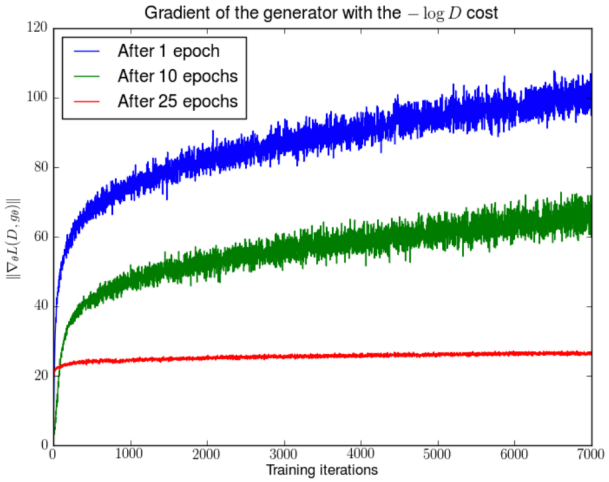
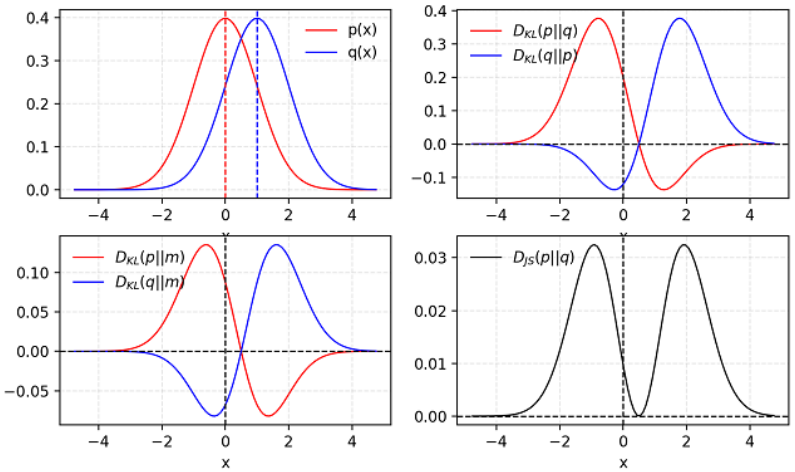


Figure 3: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the $-\log D$ cost function. We see the gradient norms grow quickly. Furthermore, the noise in the curves shows that the variance of the gradients is also increasing. All these gradients lead to updates that lower sample quality notoriously.

- Change $\log(1 - D_\theta(G_\psi(z^{(i)})))$ to $-\log(D_\theta(G_\psi(z^{(i)})))$
- Recall: don't be paralyzed by losses visualization $\rightarrow$ don't do early stoppping.

Image credit: Martin Arjovsky and Leon Bouttou 2017

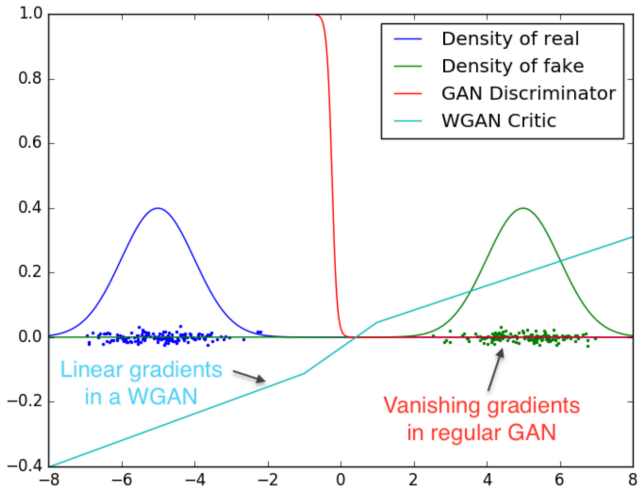
Even more about the original loss



- It minimizes $D_{JS}(p\|q) = \frac{1}{2}D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q\|\frac{p+q}{2})$
- It's symmetric, but recall from VAE lectures: only Reverse KL forces the learning of the appx distribution q close to p properly
- D_{JS} and D_{KL} has limitation: it considers similar but disjointed distributions different. E.g. $\mathbf{N}(\mathbf{1}, \mathbf{I})$ and $\mathbf{N}(-\mathbf{1}, \mathbf{I})$

Image credit: Prajjwal Bhargava intel.com

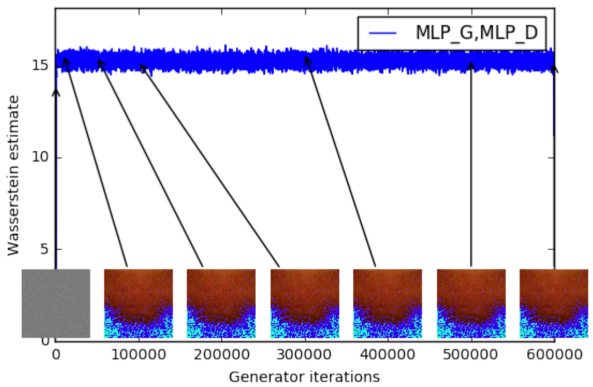
Fix: Use Wasserstein Distance

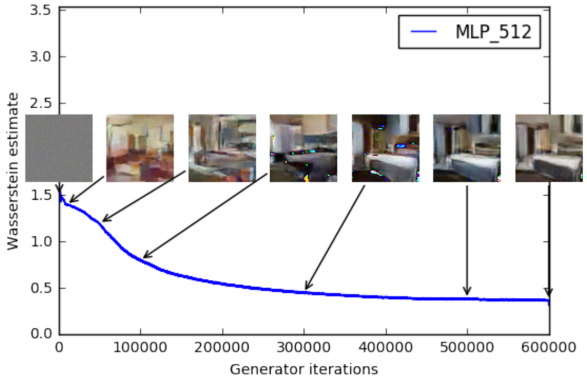


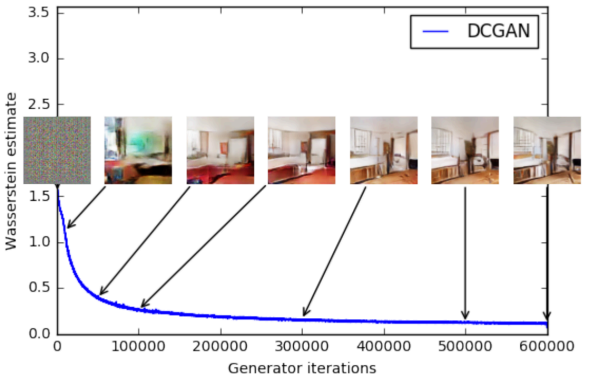
- $W(p, q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$ where $\Pi(p, q)$: set of all joint distribution $\gamma(x, y)$ whose marginals are p and q respectively
- Intuition: cost to move the "mass" to transform/morph q to p
- Borrowed from Optimal Transport, *a.k.a* Earth Moving distance, a well-backed theory

Image credit: Martin Arjovsky *et al.* 2017

WGAN[5] reflects training progress







- Loss does not decrease when training is not good
- And decreases when results are better and better

[5] https://arxiv.org/pdf/1701.07875.pdf

Image credit: Martin Arjovsky *et al.* 2017

WGAN results

Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.

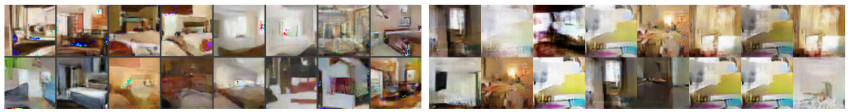
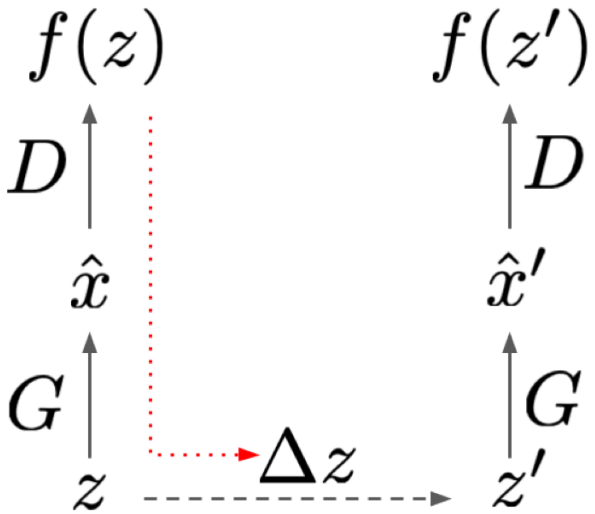


Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

- G can learn even when D is very good
- No mode collapse in authors' experiments

Image credit: Martin Arjovsky *et al.* 2017

Latent Space Optimization (LOGAN[6])



- Latent variable z is refined per sample before being fed to G
- Uses natural gradients ($F^{-1}\nabla_{\psi}$) to update z
- Introduces no new params

[6] https://arxiv.org/pdf/1912.00953.pdf

Image credit: Yan Wu *et al.* 2019

LOGAN

(a) (b)

Figure 1: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with similarly high IS. Samples from the two panels were drawn from truncation levels corresponding to points C and D in figure 3 **b** respectively. (FID/IS: (**a**) 27.97/259.4, (**b**) 8.19/259.9)

- One of the best, if not the best now for qualitative and quantitative perspectives

Image credit: Yan Wu *et al.* 2019

(Some) Other Cost Functions

Carnegie Mellon

- GAN: $L_D = \mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))]$ and $L_G = \mathbb{E}[\log D(G(z))]$
- LSGAN: $L_D = \mathbb{E}[(D(x) - 1)^2] + \mathbb{E}[D(G(z))^2]$ and $L_G = \mathbb{E}[(D(G(z)) - 1)^2]$
- WGAN: $L_D = \mathbb{E}[D(x)] - \mathbb{E}[D(G(z))]$ and $L_G = \mathbb{E}[D(G(z))]$
- WGAN_GP: $L_D = L_D^{WGAN} + \lambda \mathbb{E}[(\nabla D(|\alpha x - (1 - \alpha G(z)))| - 1)^2]$
- DRAGAN: $L_D = L_D^{GAN} + \lambda \mathbb{E}[(\nabla D(|\alpha x - (1 - \alpha x_p))| - 1)^2]$ and $L_G = L_{GAN}$
- None of them really outperforms others[7]

[7] https://arxiv.org/pdf/1711.10337.pdf

Heuristic Optimization Tricks

Carnegie Mellon

- Noise and labels:
  - Inject noise to input/noise
  - Ellipsoid-like noise space (e.g. Gaussian)
  - Label smoothing for D (*e.g* 0.9 and 0.1 instead of 1. and 0.)
- Monitor gradients
- Flip the labels (certainly, for D)
- Focus on one mode at a time
- It's unsupervised, but if you have labels, exploit them (semi)
- SGD for D, Adam for G
- Historical Averaging with L2 penalty $\|\theta - \frac{1}{t}\sum_{i=1}^{t}\theta_i\|_2$
- History replay (from DeepRL): D optimizes many G states
- Leaky ReLU, AVG Pooling, avoid sparse ops: ReLU, Max Pooling
- Other hyperparams tuning trick as usual.

Credit: Tim Salimans *et al.* 2016, Soumith Chintala and ++

Monitor (Vanishing) Gradients

Carnegie Mellon

- Monitor gradients of each layer for *both* D and G
- Check gradients norm: should not be too big, *e.g.* > 100

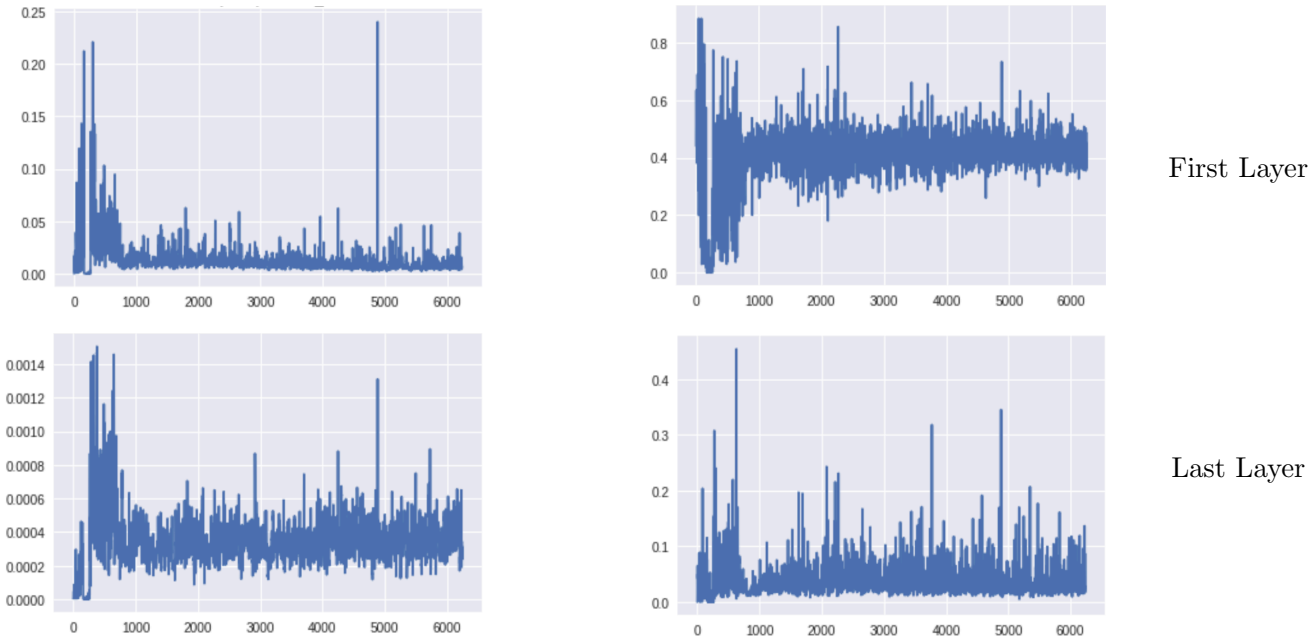
Image Credit: Utkarsh Desai (Medium.com)

Monitor (Vanishing) Gradients

Carnegie Mellon

GENERATOR

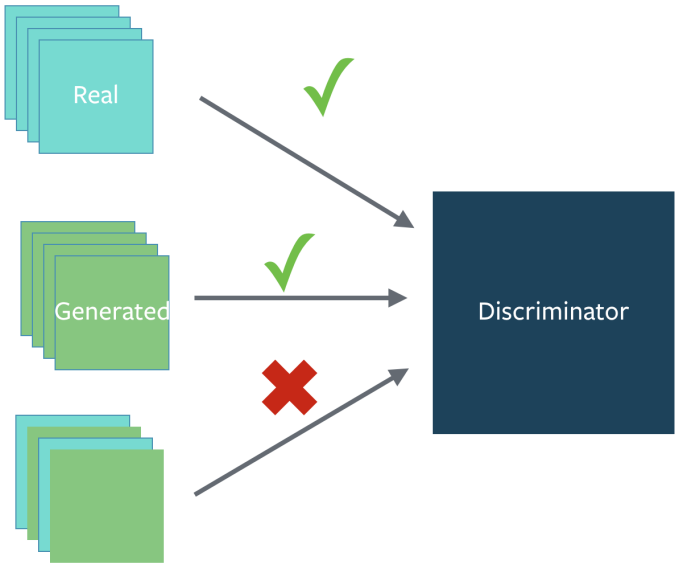
DISCRIMINATOR



- Inject noise for training input and smooth D's labels
- Flip the labels (similar to injecting noise): fake $= 1$ and real $= 0$
- Don't try to find a good schedule to uncollapse training. If so, have a principle, e.g. no update D if loss$(D) < c$
- Or sometimes, training D more than G

Image Credit: Utkarsh Desai (Medium.com)

Batch Norm also helps



- Label-based batch normalization
- Do single-label minibatches, don't mix
- Instance-based normalization: $x = \frac{x - \mu_x}{\sqrt{\sigma_x^2 + \epsilon}}$

Image Credit: Soumith Chintala github

Focus on One Mode

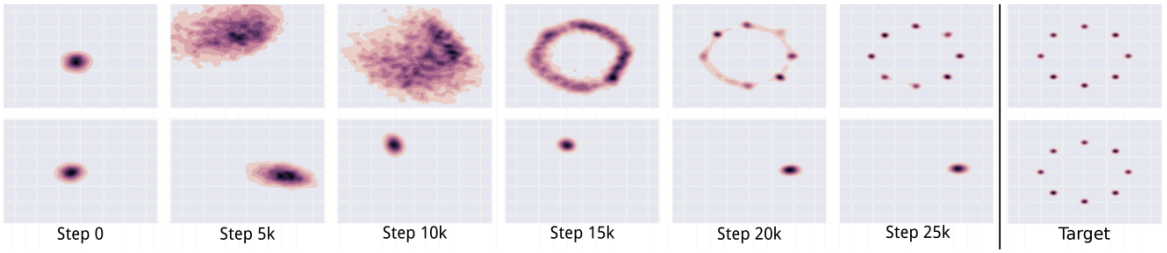


Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling steps. Its generator quickly spreads out and converges to the target distribution. The bottom row shows standard GAN training. The generator rotates through the modes of the data distribution. It never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

- Train one mode well before moving on (let collapse happen)
- Use multiple trained models to generate multi-modes

Image Credit: Kuke Metz *et al.* 2017

Outline

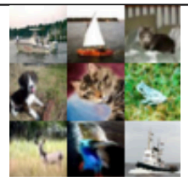
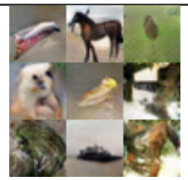
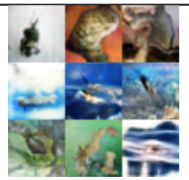
Evaluating GAN is Not Easy

Carnegie Mellon

- Recall: Optimization in GAN is 2 players try to fool each other, not directly towards the ground truths
- Human is still the most credible one.
- But it's qualitative and subjective.
- Quantitative metrics:
  - Inception Score (IS) (Salimans *et al* 2016)
  - Frechet Inception Distance (FID) (Heusel *et al.* 2017)
  - Precision, Recall and F1 (Lucic *et al.* 2018)

Inception Score (IS[8])

Carnegie Mellon

Samples							
Model	Real data	Our methods	-VBN+BN	-L+HA	-LS	-L	-MBF
Score ± std.	11.24 ± .12	8.09 ± .07	7.54 ± .07	6.86 ± .06	6.83 ± .06	4.36 ± .04	3.87 ± .03

- Use InceptionNet to classify generated images $x \sim G(z)$
- High quality generated image: $p(y|x)$ has low entropy
- Generated images are diverse: $p(y) = \int_z p(y|x = G(z))dz$ has high entropy $\rightarrow$ their distance should be *high* for a good model:

$$\text{IS} = \exp\left(\mathbb{E}_G\left[\mathbf{D}_{KL}[p(y|x)\|p(y)]\right]\right) = \exp\left(H(y) - H(Y|x)\right) \quad (1)$$

- Downside: Costly, needs a lot of samples to measure variety

[8] https://arxiv.org/pdf/1606.03498.pdf

Image Credit: Salimans *et al.* 2016

Frechet Inception Distance (FID[9])

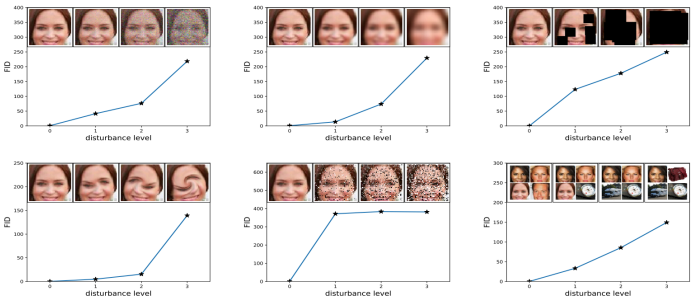


Figure 3: FID is evaluated for **upper left:** Gaussian noise, **upper middle:** Gaussian blur, **upper right:** implanted black rectangles, **lower left:** swirled images, **lower middle:** salt and pepper noise, and **lower right:** CelebA dataset contaminated by ImageNet images. The disturbance level rises from zero and increases to the highest level. The FID captures the disturbance level very well by monotonically increasing.

- Take Pool3 features from InceptionNet
- Assume real distribution $\mathbf{N}(\mu, \mathbf{\Sigma})$ and fake $\mathbf{N}(\mu_{\mathbf{w}}, \mathbf{\Sigma}_{\mathbf{w}})$
- Their distance should be *low* for a good model:

$$\text{FID} = \|\mu - \mu_{\mathbf{w}}\|_2^2 + Tr(\mathbf{\Sigma} + \mathbf{\Sigma}_{\mathbf{w}} - 2(\mathbf{\Sigma}\,\mathbf{\Sigma}_{\mathbf{w}})^{1/2}) \tag{2}$$

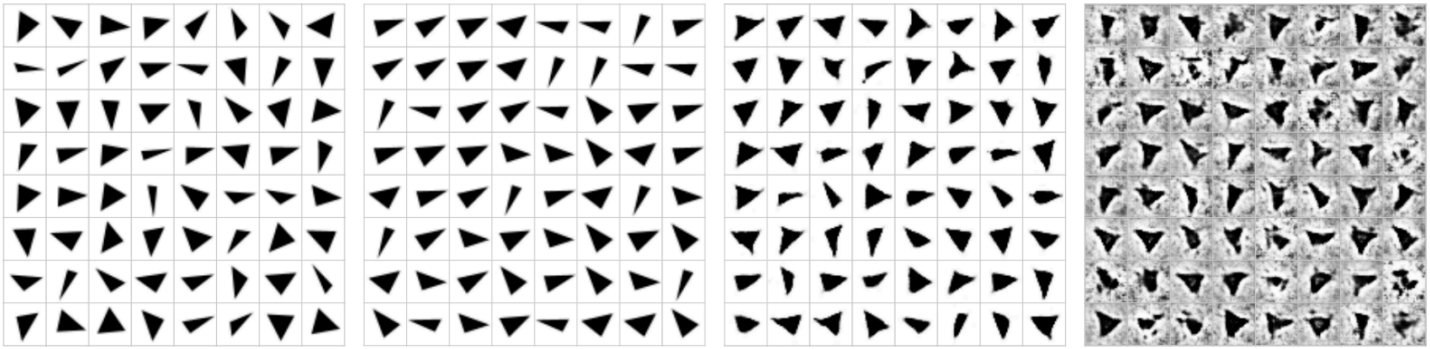
- Downside: still relies on feature extraction and has expensive cost
- Upside: sensitive to mode collapse

[9] https://arxiv.org/pdf/1706.08500.pdf

Image Credit: Martin Heusel *et al.* 2017

Precision, Recall and F1[10]

Carnegie Mellon



(a) High precision, high recall (b) High precision, low recall (c) Low precision, high recall (d) Low precision, low recall

Figure 2: Samples from models with (a) high recall and precision, (b) high precision, but low recall (lacking in diversity), (c) low precision, but high recall (can decently reproduce triangles, but fails to capture convexity), and (d) low precision and low recall.

- P, R and F1 are classical metrics
- IS captures P only, FID captures both P and R
- Those metrics are complementary to IS and FID

[10] https://arxiv.org/pdf/1711.10337.pdf

Image Credit: Mario Lucic *et al.* 2018