



Practical Course – **Machine Learning in Medical Imaging**

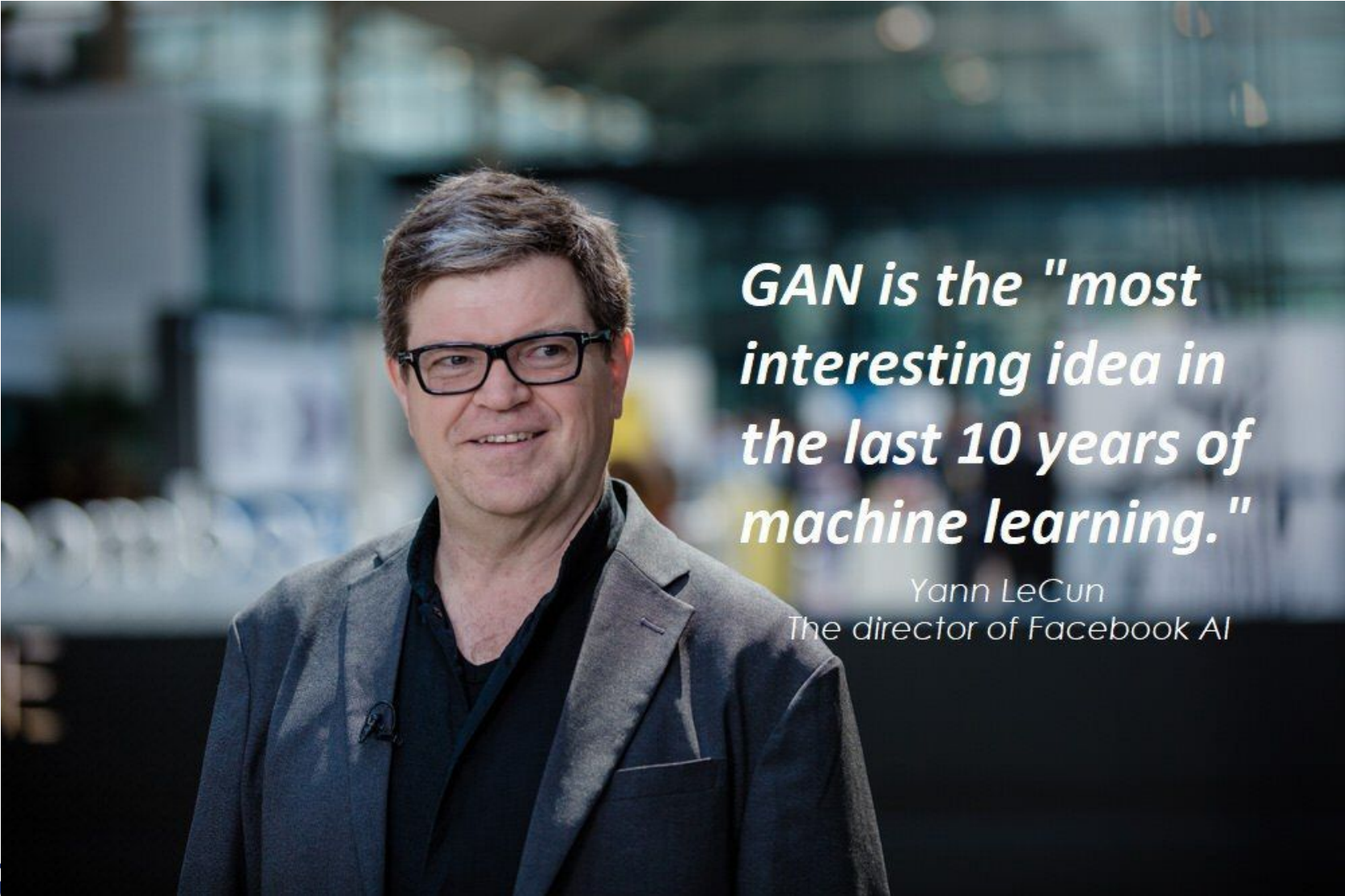
Generative Adversarial Networks

Azade Farshad, PhD Candidate

Chair for Computer Aided Medical Procedures

(slides adapted from Gustavo Carneiro)





GAN is the "most interesting idea in the last 10 years of machine learning."

Yann LeCun
The director of Facebook AI

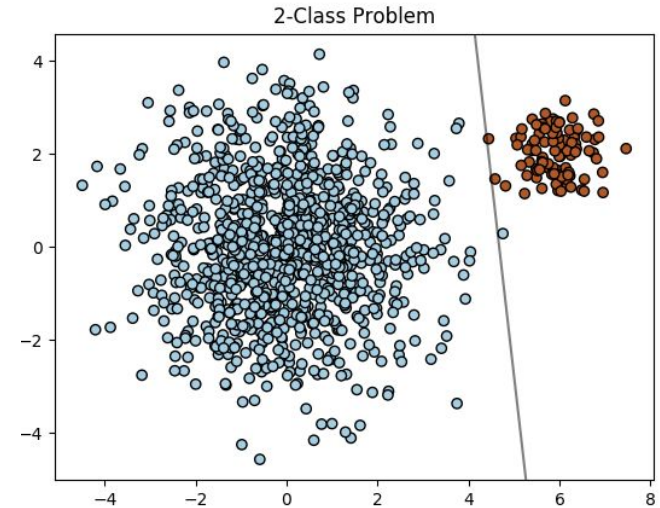
Discriminative Models

- Models conditional probability (target conditioned on data): $p(y|x)$
- Examples: SVM, Random Forest & Neural Network, CRF
- Can't sample, doesn't model distribution, but it's better for classification
- We optimize the conditional likelihood

$$-\log p(y|X)$$

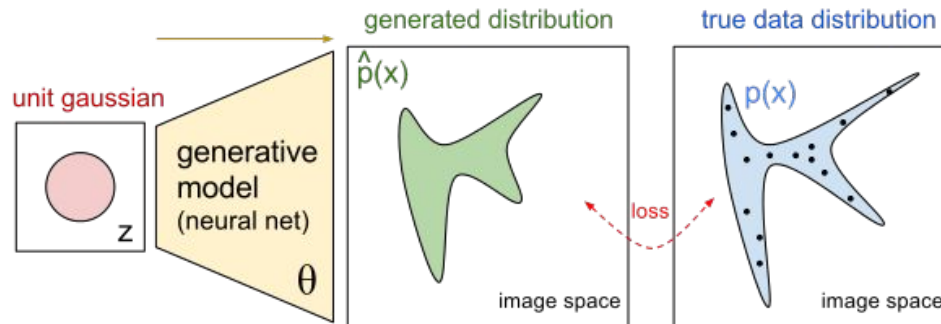
- Generative: joint log-likelihood

$$-\log p(y, X) = -\log(p(y|X)p(X)) = -\log p(y|X) - \log p(X)$$



Generative Models

- Sample from $p(x | \theta)$
 - Capture the joint probability between data (x) and target (y): $p(x,y)$
 - Can also model the probability of data (x) given target (y): $p(x|y)$
 - Can generate data and targets or data given target
 - Generate samples $y,x \sim p(y,x)$ or $y \sim p(y), x \sim p(x|y)$
 - Need a noise for sampling, $z \sim N(0,1), y,x \sim f_{\theta}(z)$
- Examples:
 - GMM, HMM, Naive Bayes classifier, Monte Carlo



Discriminative vs. Generative Models

Discriminative models

- Discriminate between different kinds of data instances.
- Capture the conditional probability $p(y | x)$.
- Pros:
 - Need less data
 - Computationally cheaper
- Cons:
 - Not useful for unsupervised learning
 - Can be more difficult to interpret

Generative models

- Can generate new data instances.
- Capture the joint probability $p(x, y)$, or just $p(x)$ if there are no labels.
- Pros:
 - Good at unsupervised machine learning
 - We get the underlying idea of what the class is built on
- Cons:
 - Very computationally expensive

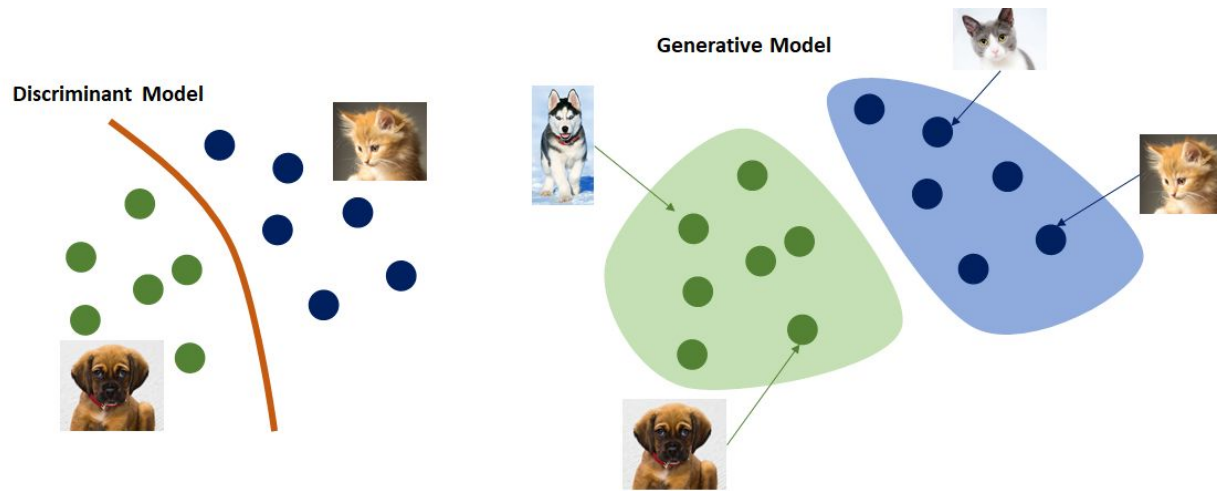


photo from: <https://medium.com/@jordi299/about-generative-and-discriminative-models-d8958b67ad32>



Generative Models - Autoencoder

- No need to labels

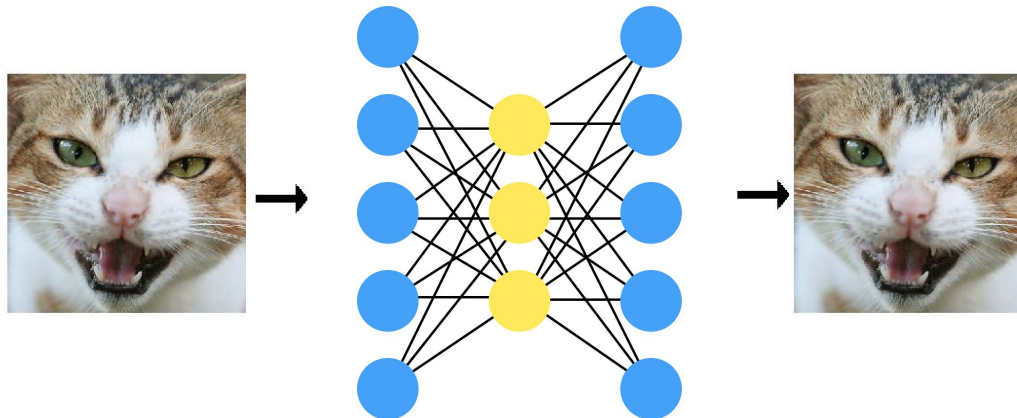
- reconstruction loss
- cross-entropy for binary values

$$f(x) = x'$$
$$l(f(x)) = - \sum_k (x_k \log(x'_k) + (1 - x_k) \log(1 - x'_k))$$

- Sum of squared errors for real values
- $$l(f(x)) = \frac{1}{2} \sum_k (x'_k - x_k)^2$$

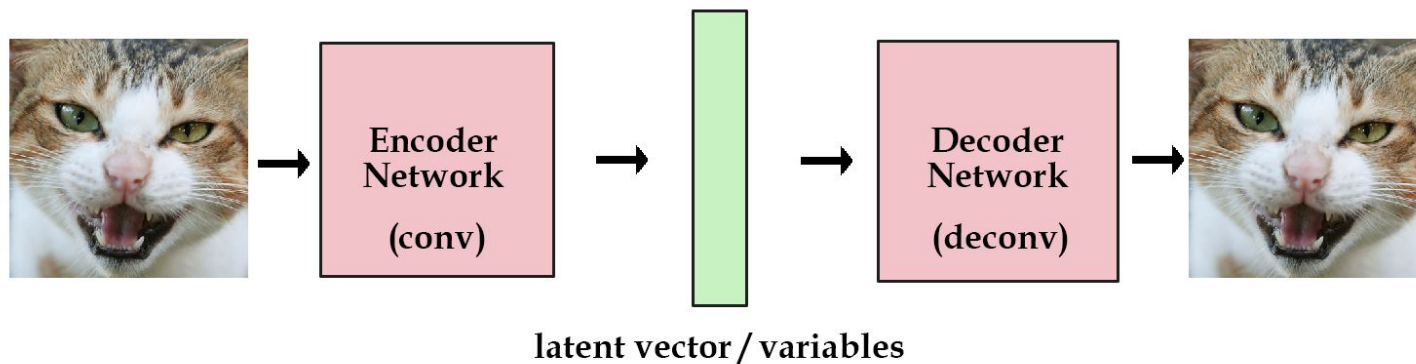
- latent vector: low dimensional representation

- can be used for data compression
- latent representation used for clustering



Generative Models - Autoencoder

- Variations
 - Vanilla Autoencoder
 - 3 layers network
 - Multilayer Autoencoder
 - Convolutional Autoencoder
 - Denoising Autoencoder



Generative Models - Variational Autoencoder

- Learn the distribution
 - selective sampling
- Reparameterization
 - Construct latent representation from mean, std

$$\mathcal{L}^B = -\text{KL}[\underbrace{q_\phi(\mathbf{z} | \mathbf{x}^{(i)})}_{\text{Encoder}} || \underbrace{p_\theta(\mathbf{z})}_{\text{Fixed}}] + \frac{1}{L} \sum_{l=1}^L \log \underbrace{p_\theta(\mathbf{x}^{(i)} | \mathbf{z}^{(l)})}_{\text{Decoder}}$$

$$\mathcal{L}_{\text{KLD}} = \sum_{i=1}^n \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

$$\mu_x, \sigma_x = M(\mathbf{x}), \Sigma(\mathbf{x})$$

Push \mathbf{x} through encoder (1)

$$\epsilon \sim \mathcal{N}(0, 1)$$

Sample noise (2)

$$\mathbf{z} = \epsilon \sigma_x + \mu_x$$

Reparameterize (3)

$$\mathbf{x}_r = p_\theta(\mathbf{x} | \mathbf{z})$$

Push \mathbf{z} through decoder (4)

$$\text{recon. loss} = \text{MSE}(\mathbf{x}, \mathbf{x}_r)$$

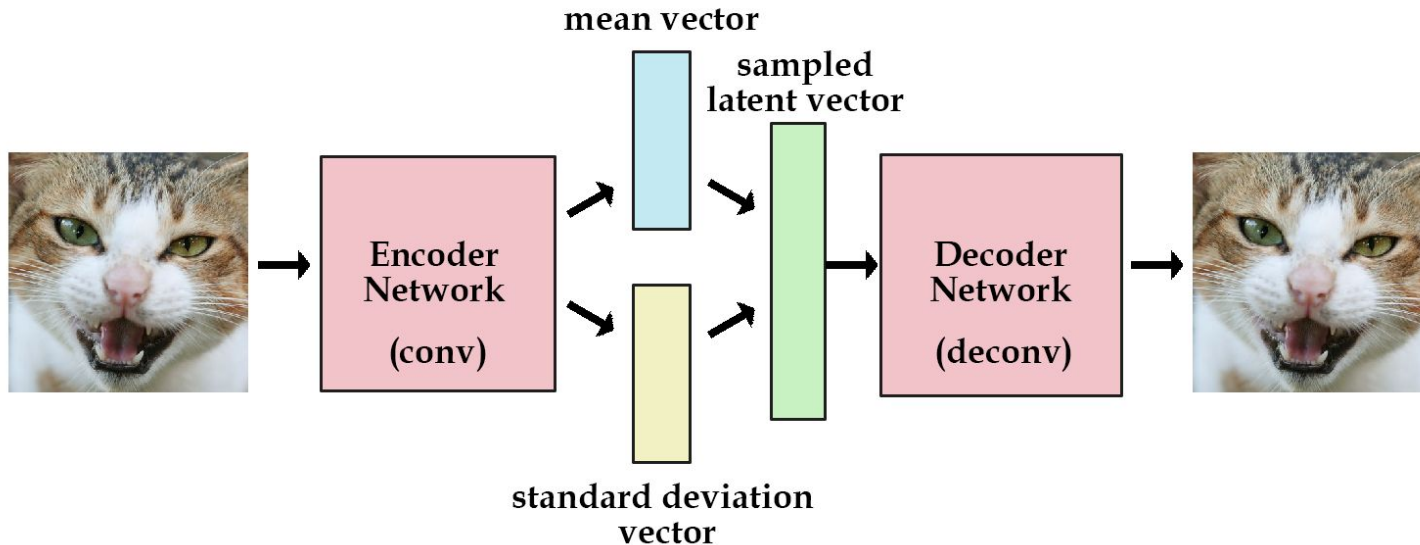
Compute reconstruction loss (5)

$$\text{var. loss} = -\text{KL}[\mathcal{N}(\mu_x, \sigma_x) || \mathcal{N}(0, I)]$$

Compute variational loss (6)

$$L = \text{recon. loss} + \text{var. loss}$$

Combine losses (7)



Generative Adversarial Networks

- Use Cases: generate realistic images, realistic text,

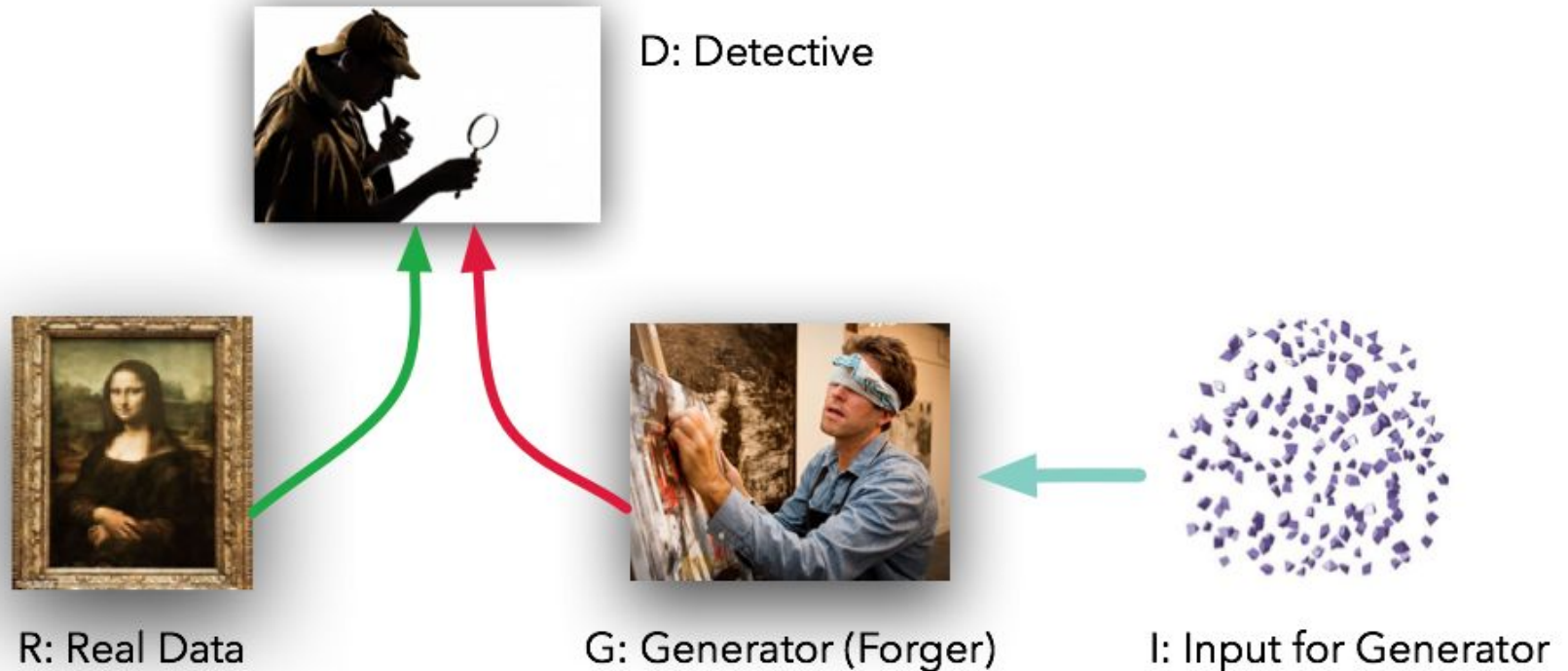


photo from:
<https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f&sa=D&ust=1574267641133000&usg=AFQjCNEI0qiToDdW7vYU3ZAdihHTZ9pXQ>

Generative Adversarial Networks

- Original Idea
 - Adversarial learning
 - Train two models (G and D) simultaneously
 - Model G generates images from input noise
 - Model D classifies images into real or fake / generated
 - Model G tries to fool model D
 - Model D acts as supervision on model G
 - G stands for Generator and D for Discriminator
- GAN Lab: Watch GAN training, draw distribution:
 - <https://poloclub.github.io/ganlab/>

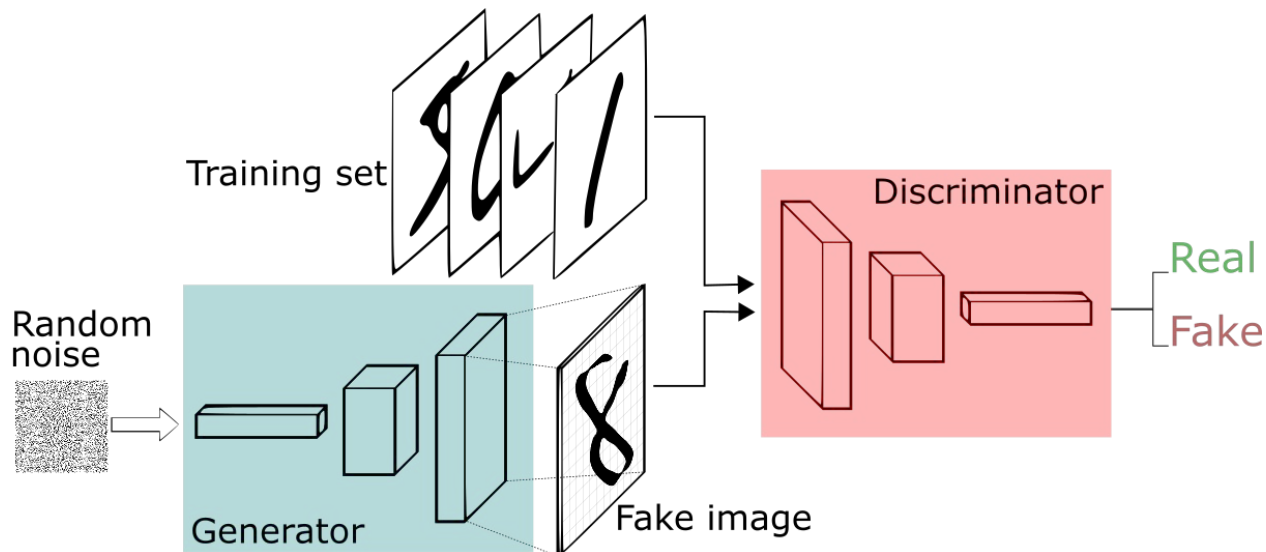
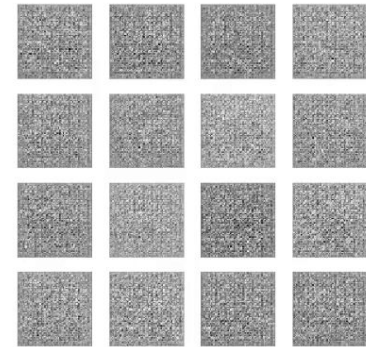
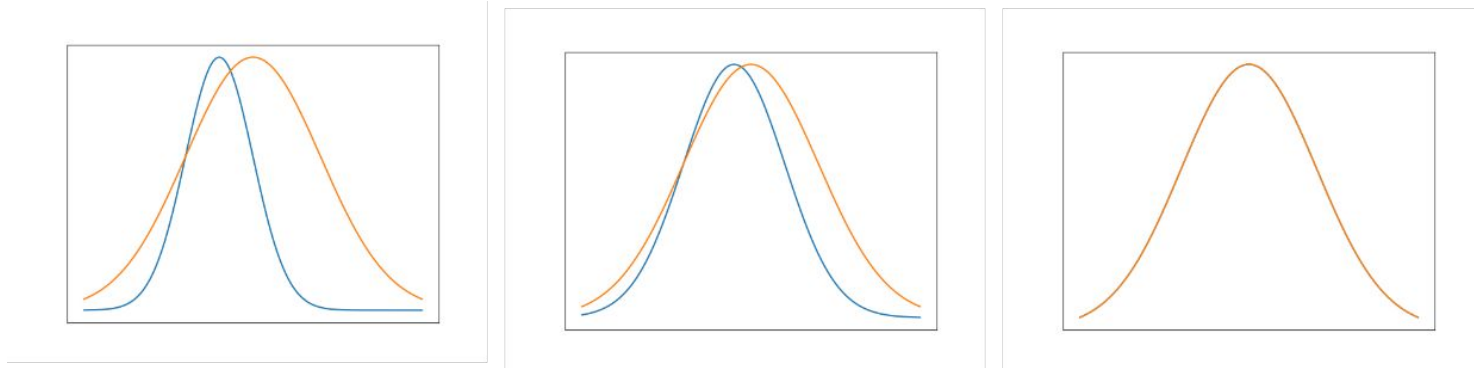


Image credit: [Thalles Silva](#)



Generative Adversarial Networks

- Nash Equilibrium
- Match distribution



- Draw samples (x) from the data distribution (p_{data}) **real images**
- Draw samples from the model distribution (p_g) based on a set of latent variables (z) drawn from a prior distribution (p_z) **generated images**
- Detect samples coming from the data distribution (e.g. real images)
Discriminator
- Goal: $p_g = p_{\text{data}}$ **Nash equilibrium**

GAN Learning Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log \left(1 - D(G(\mathbf{z}^{(i)})) \right) \right].$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(1 - D(G(\mathbf{z}^{(i)})) \right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

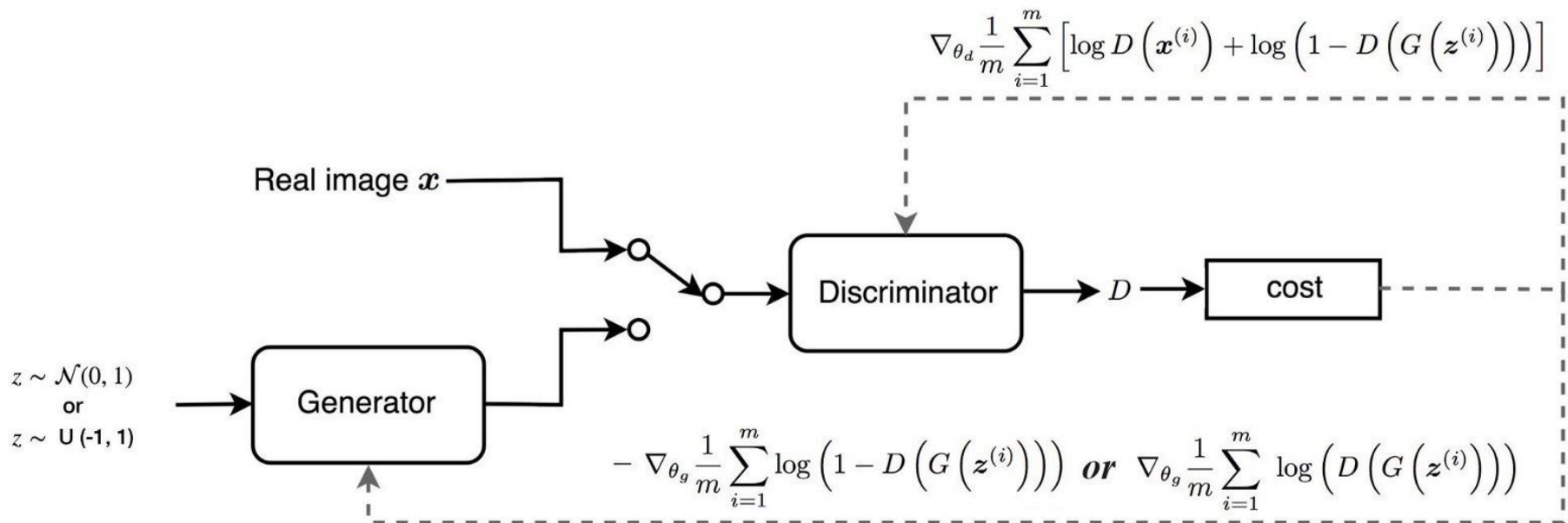


GAN Loss

- Minimax (each player tries to minimize her maximum loss)
- Optimal solution is $D(x) = D(G(z)) = 0.5$ (ideal case)

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

data distribution
Discriminator
input noise
Generator



Advantages and Disadvantages

- Hard to train
- Hard to scale up to deep architectures
- No explicit representation of $p_{g(x)}$

- + No need for sampler
- + No need for inference
- + Approximation of wide range of functions
- + Applicable to different vision tasks
- + Label-free training



Hard to train

- Non-convergence
 - the model parameters oscillate, destabilize and never converge,
- Mode collapse
 - the generator collapses which produces limited varieties of samples,
- Diminished gradient
 - the discriminator gets too successful that the generator gradient vanishes and learns nothing,
- Unbalance between the generator and discriminator causing overfitting
- Highly sensitive to the hyperparameter selections.



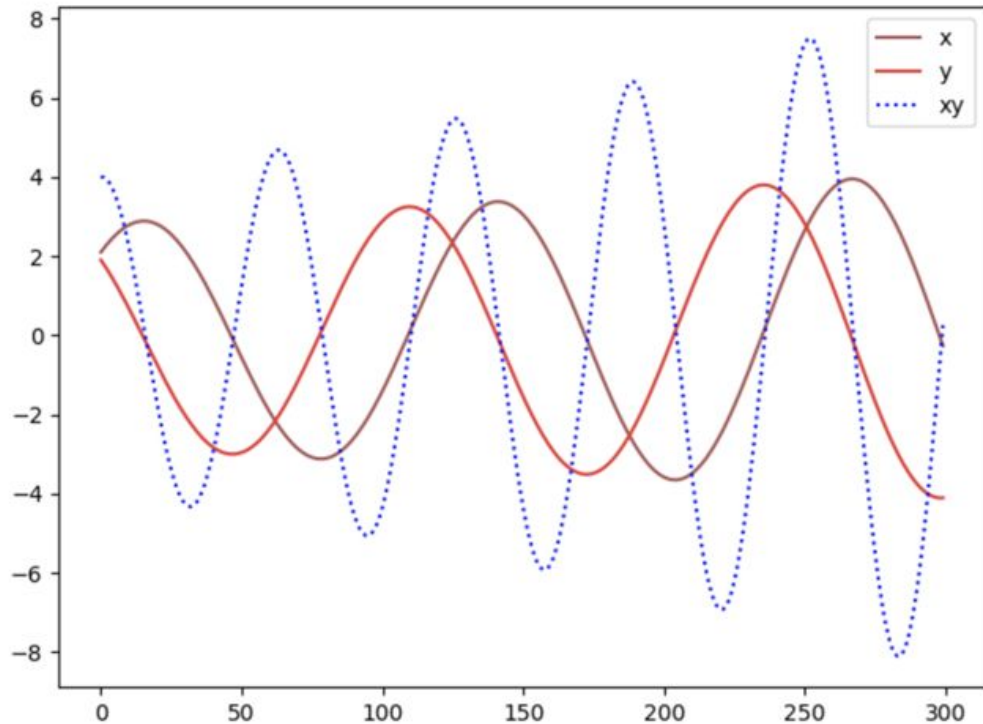
Hard to train - Non-convergence

Nash equilibrium: one player will not change its action regardless of the opponent's action

$$\min_B \max_A V(D, G) = xy$$

$$\Delta x = \alpha \frac{\partial(xy)}{\partial(x)}$$

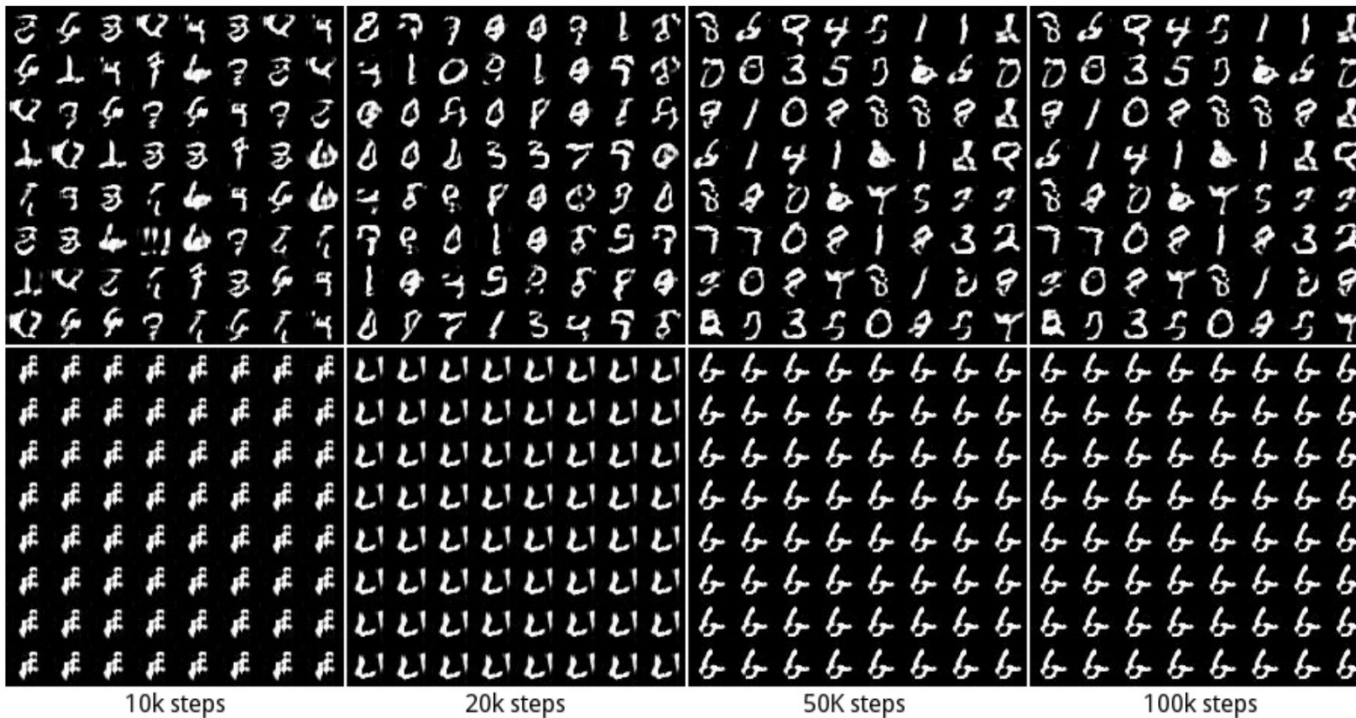
$$\Delta y = -\alpha \frac{\partial(xy)}{\partial(y)}$$



https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-adversary-networks-819a86b3750b



Hard to train - Mode Collapse



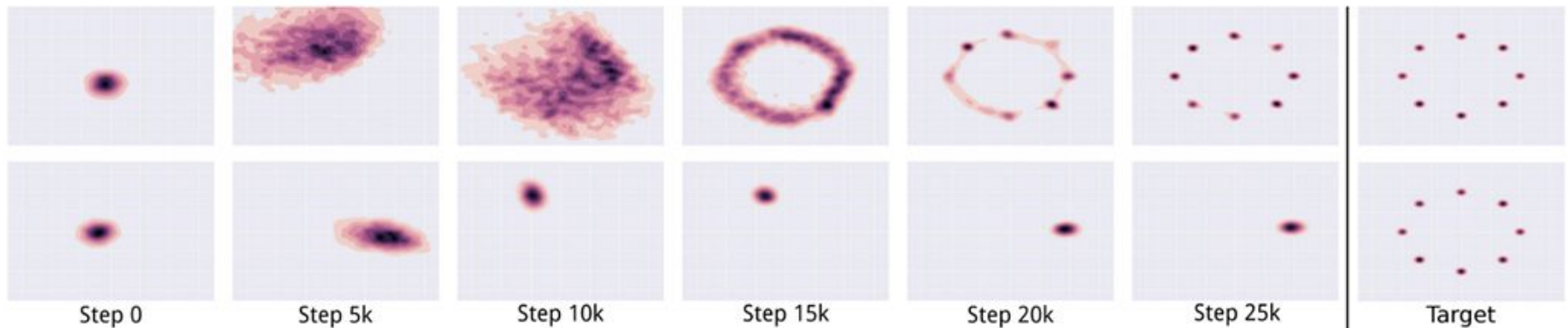
<https://arxiv.org/pdf/1611.02163.pdf>

<https://arxiv.org/pdf/1703.10717.pdf>

Hard to train - Mode Collapse

Solution:

1. Train G only (i.e., not D) until grad mag is zero. Single mode independent of z :
$$x^* = \operatorname{argmax}_x D(x)$$
2. Train D – very easy training, just detect this mode!
3. When training G again, it will search for next mode... wash, rinse and repeat.



<https://arxiv.org/pdf/1611.02163.pdf>

Hard to train

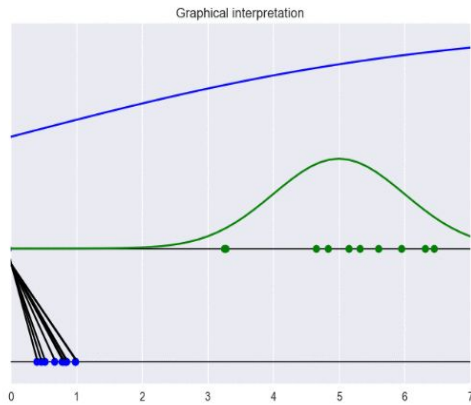
- Take your time for the tuning of hyperparameters
 - it takes a lot of patience, but it pays off
- Balance between the training for generator and discriminator
- GAN loss measures how well discriminator is doing wrt generator and vice-versa
 - not an absolute measure, so loss can go up, and generated images can improve



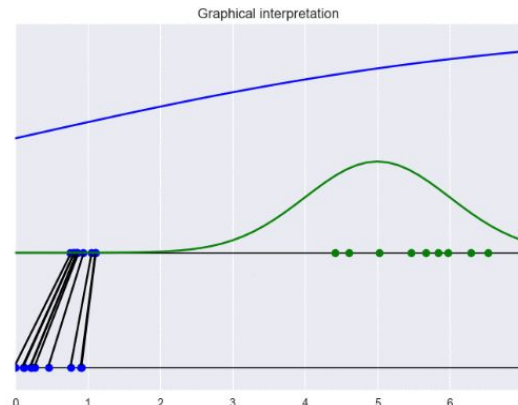
Regularization

- strong G: faster convergence to the optimal solution. Nevertheless, a bad initialization could lead to instability problems
- strong D: This examples shows the problem of vanishing gradients. It is clear that the generator cannot be learned properly due to the excessive strength of the discriminator. In this case, the training converges to a solution different from the optimal one

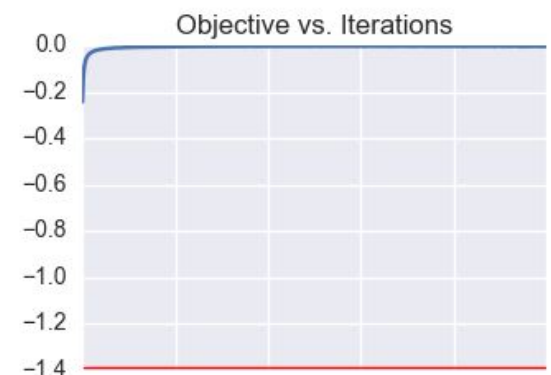
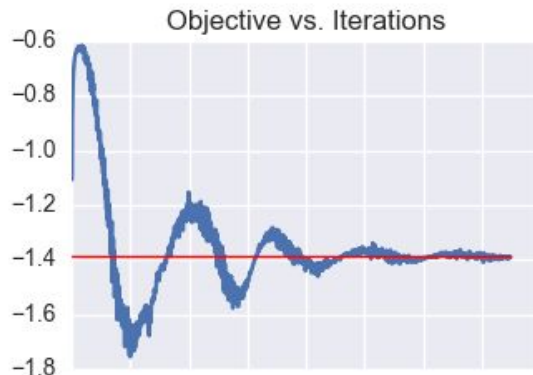
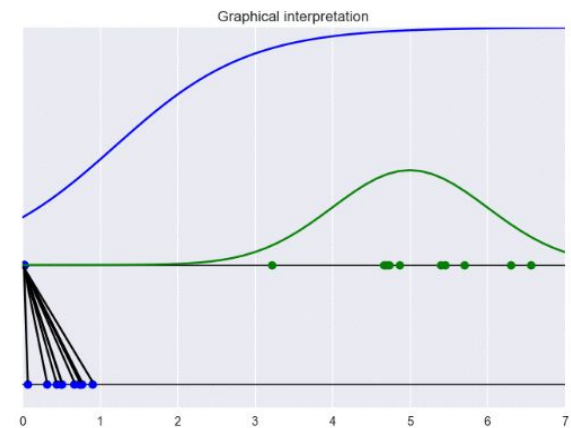
normal setting



strong Generator



strong Discriminator



<https://github.com/emansone/GAN>

GAN Performance

- Main difficulty of GANs: we don't know how good they are
- People cherry pick results in papers -> some of them will always look good, but how to quantify?
- Do we only memorize or do we generalize?
- GANs are difficult to evaluate!



GAN Performance

- Inception score (\uparrow)
 - Inception to measure image quality and diversity
 - Saliency: check whether the generated images can be classified with high confidence (i.e., high scores only on a single class)
 - Diversity: check whether we obtain samples from all classes
 - Train an accurate classifier
 - Train an image generation model (conditional)
 - Check how accurate the classifier can recognize the generated images
 - **Drawback: statistics** of the real data are **not compared** with the statistics of the generated data

$$\text{IS}(G) \approx \exp\left(\frac{1}{N} \sum_{i=1}^N D_{KL}(p(y|\mathbf{x}^{(i)}) \parallel \hat{p}(y))\right).$$

$$\hat{p}(y) = \frac{1}{N} \sum_{i=1}^N p(y|\mathbf{x}^{(i)}),$$

Inception model



Barratt, Shane, and Rishi Sharma. "A note on the inception score." arXiv preprint arXiv:1801.01973 (2018).

GAN Performance

- FRÉCHET INCEPTION DISTANCE (↓)
 - measure of similarity between two datasets of images
 - correlates well with human judgement of visual quality
 - Fréchet distance between two Gaussians fitted to feature representations of the Inception network
 - Inception network to extract features (from both real and generated images).
 - Model the data distribution with a multivariate Gaussian with mean μ and cov Σ .

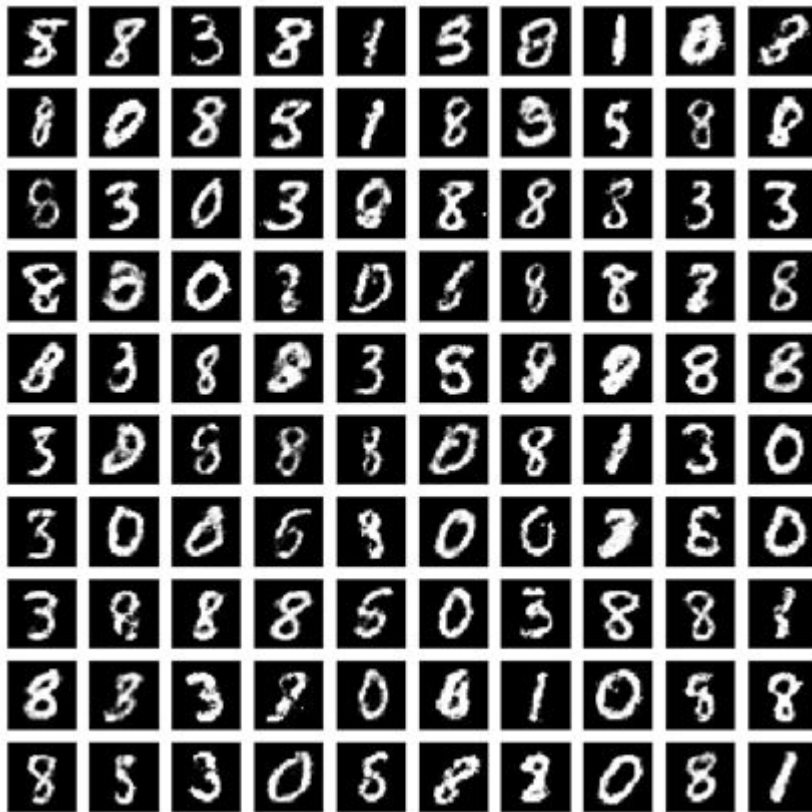
$$\text{FID}(x, g) = \|\mu_x - \mu_g\|_2^2 + \text{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}),$$

Tr: Trace Function (sum of elements on the main diagonal of the input matrix)



Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In NeurIPS 2017.

Applications (GAN)



Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

Applications (GAN)

- BigGAN (state-of-the-art in GANs)
- Large scale training
- Heavy architectures

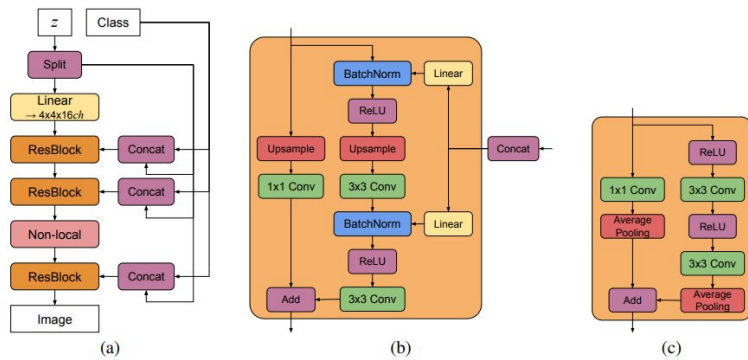


Figure 15: (a) A typical architectural layout for BigGAN's **G**; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's **G**. (c) A Residual Block (*ResBlock down*) in BigGAN's **D**.



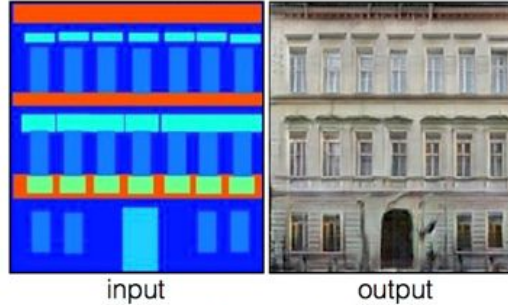
Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large scale gan training for high fidelity natural image synthesis." arXiv preprint arXiv:1809.11096 (2018).

Applications (cGAN)

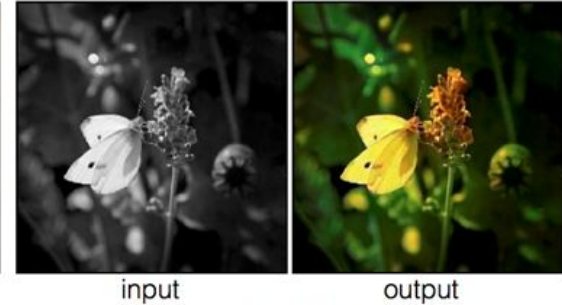
Labels to Street Scene



Labels to Facade



BW to Color



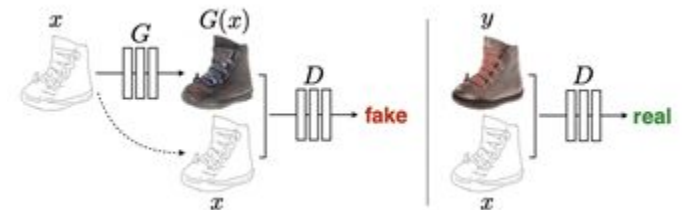
Aerial to Map



Day to Night



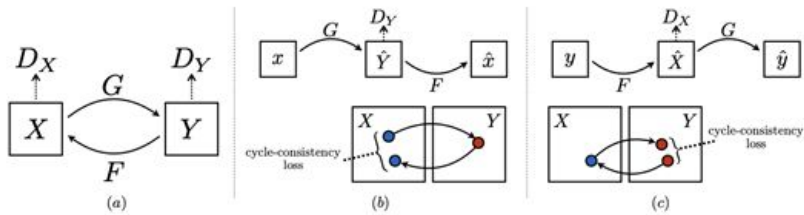
Edges to Photo



Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." *arXiv preprint arXiv:1611.07004* (2016).



Applications (cGAN)



Monet \leftrightarrow Photos



Monet \rightarrow photo

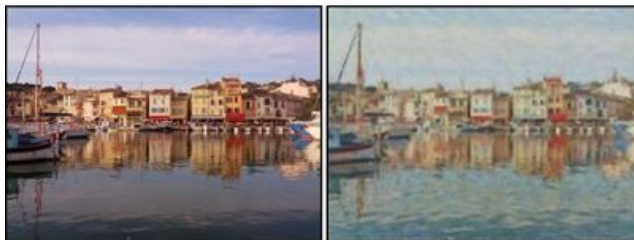


photo \rightarrow Monet

Zebra \leftrightarrow Horses



zebra \rightarrow horse

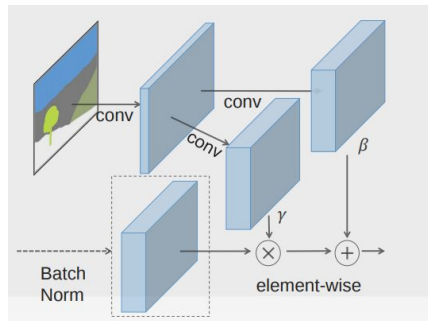
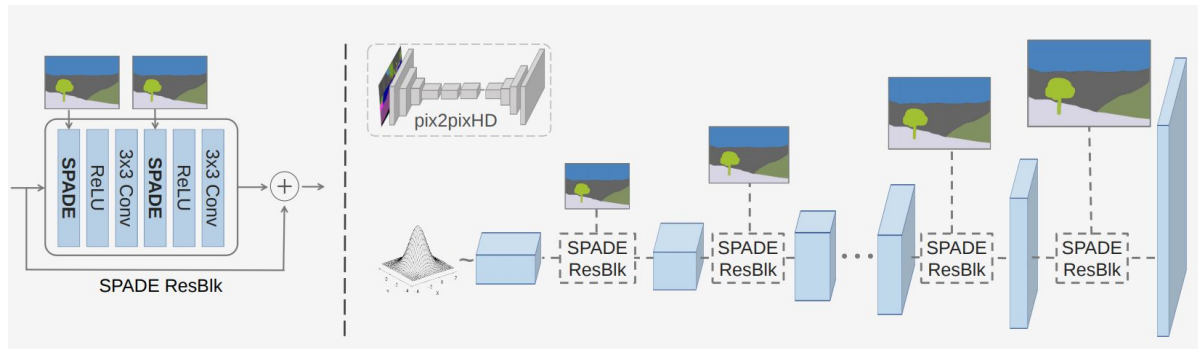


horse \rightarrow zebra

Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks.", ICCV 2017.



Applications (segmentation to image)



[Demo](#)

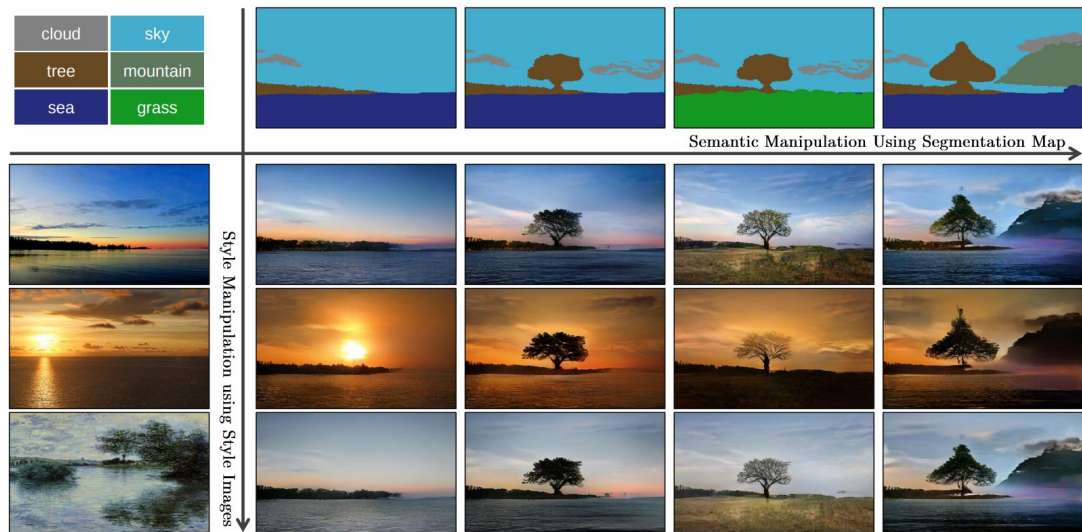
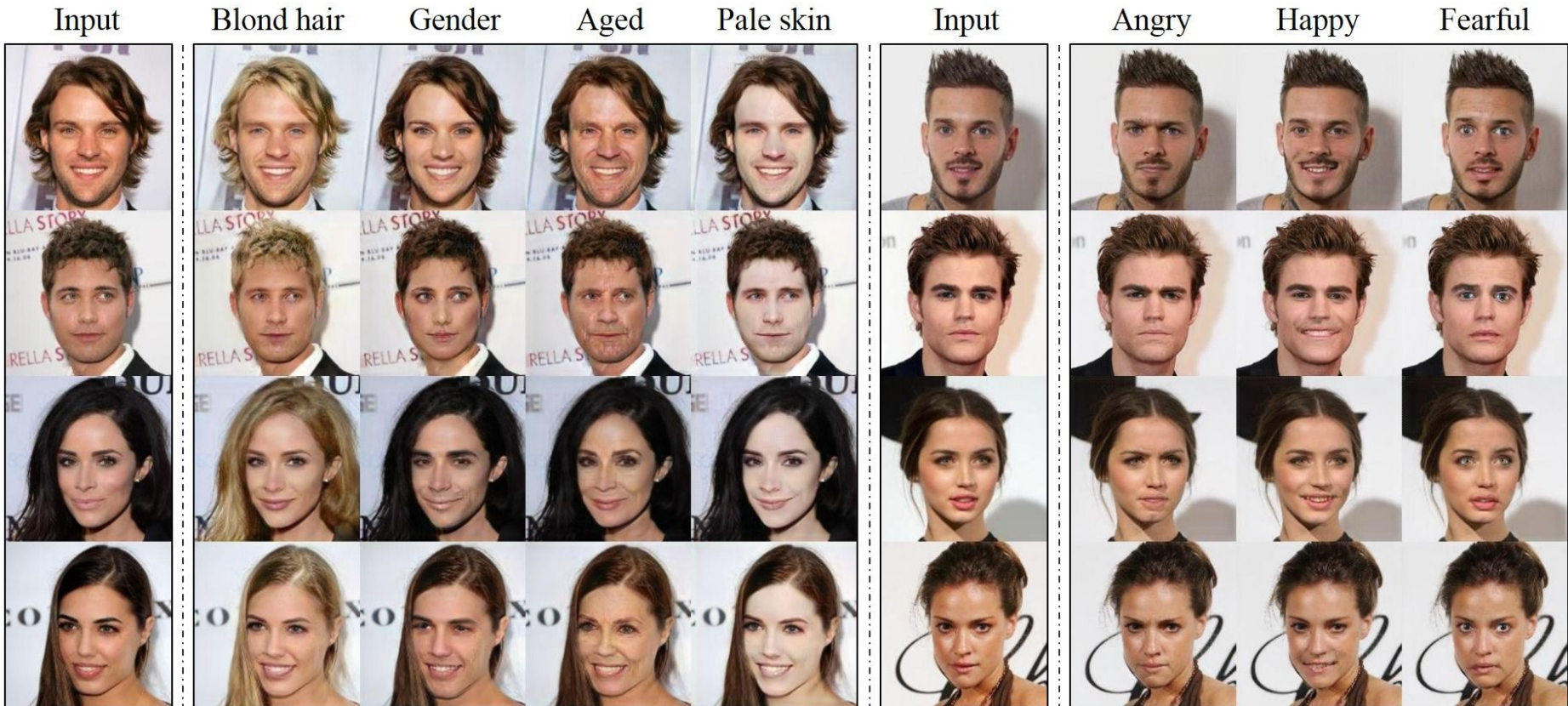


Figure 1: Our model allows user control over both semantic and style as synthesizing an image. The semantic (e.g., the existence of a tree) is controlled via a label map (the top row), while the style is controlled via the reference style image (the leftmost column). Please visit our [website](#) for interactive image synthesis demos.

Park, Taesung, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. "Semantic image synthesis with spatially-adaptive normalization." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2337-2346. 2019.



Applications (domain adaptation)



StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation . Y. Choi, M. Choi, M. Kim, J. Woo Ha, S. Kim, and J. Choo. CVPR 2018.

Applications (domain adaptation)

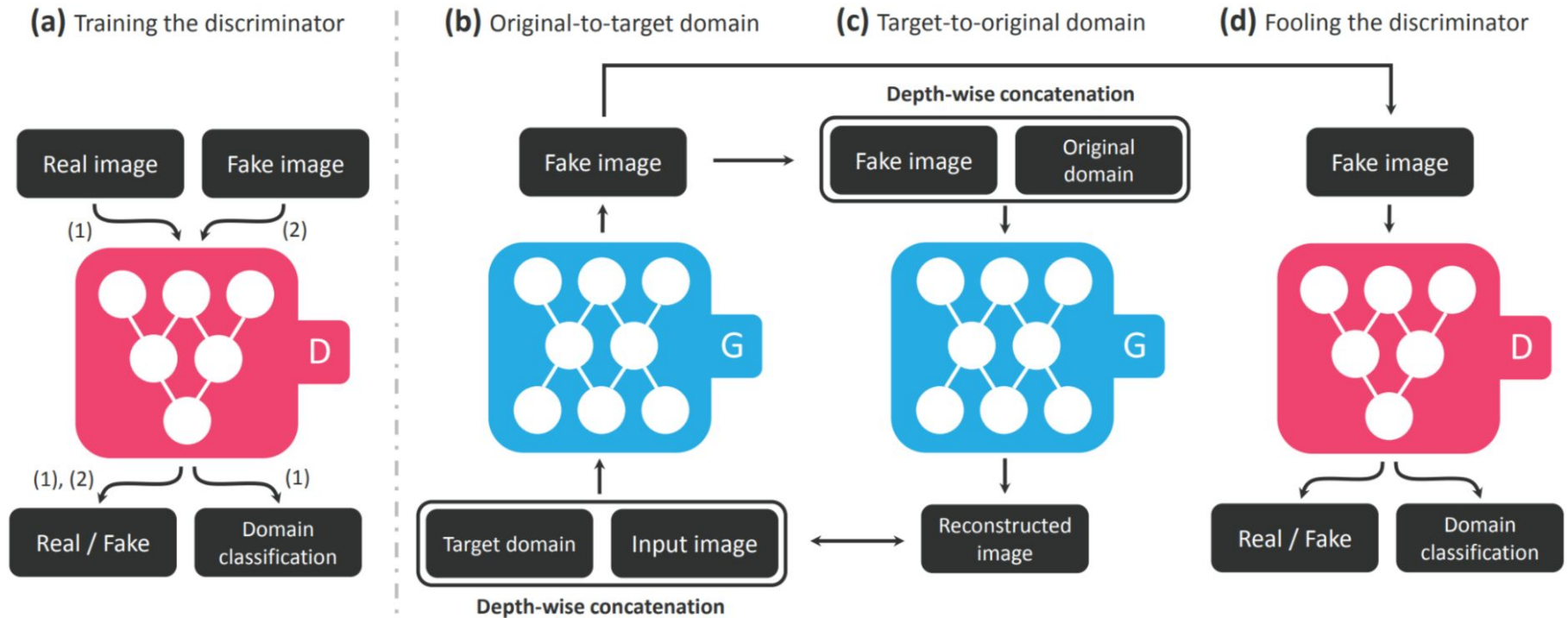


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G . (a) D learns to distinguish between real and fake images and classify the real images to its corresponding domain. (b) G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. (c) G tries to reconstruct the original image from the fake image given the original domain label. (d) G tries to generate images indistinguishable from real images and classifiable as target domain by D .



Applications (super-resolution)

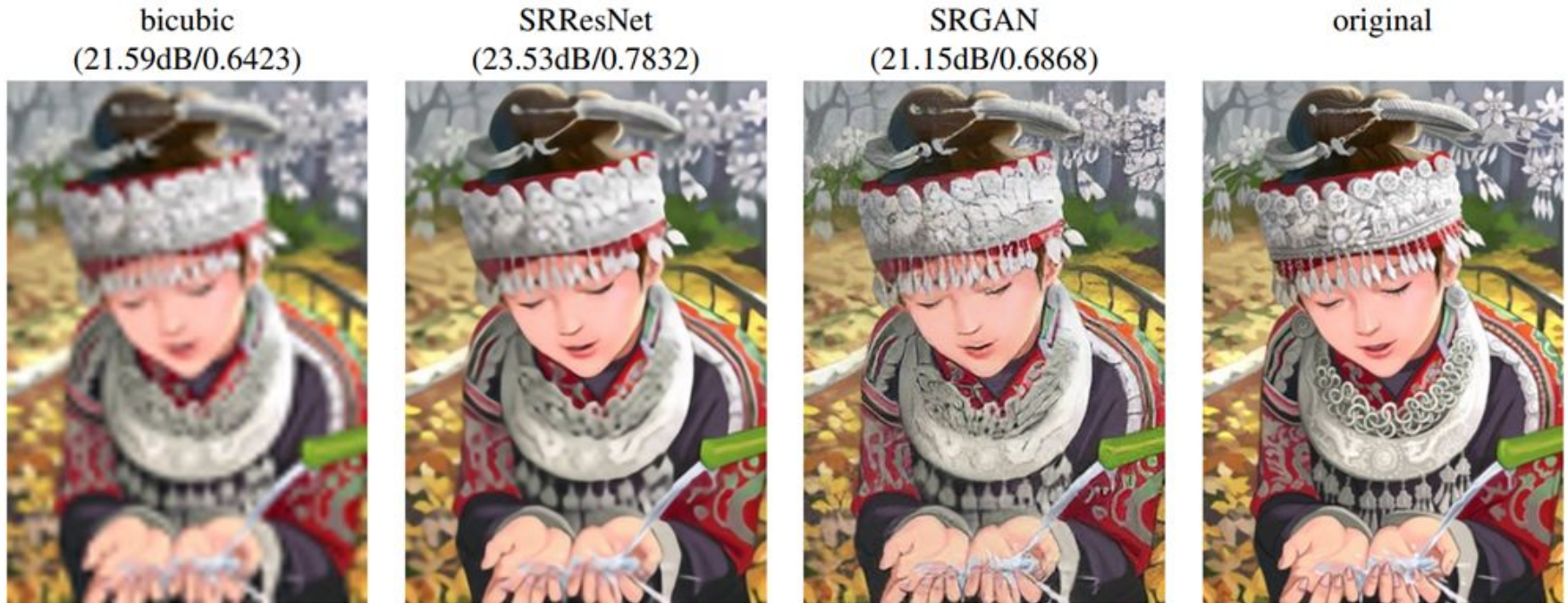


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Implementation: <https://github.com/junhocho/SRGAN>

Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *arXiv preprint arXiv:1609.04802* (2016).



Applications (text to image)

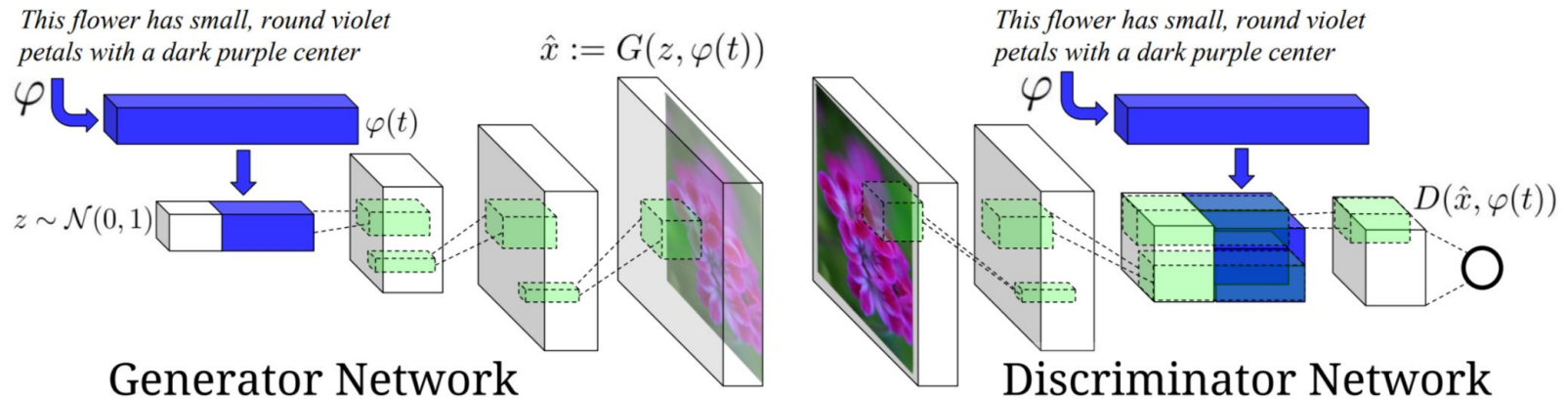


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen



this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



Implementation: <https://github.com/zsdonghao/text-to-image>

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. and Lee, H., 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.



Applications (scene graph to image)

- Box loss: penalizing the L1 difference between ground-truth and predicted boxes
- Pixel loss: penalizing the L1 difference between ground-truth generated images
- Image adversarial loss: encouraging generated image patches to appear realistic
- Object adversarial loss: encouraging each generated object to look realistic
- Auxiliary classifier loss: ensuring that each generated object can be classified by D_{obj}

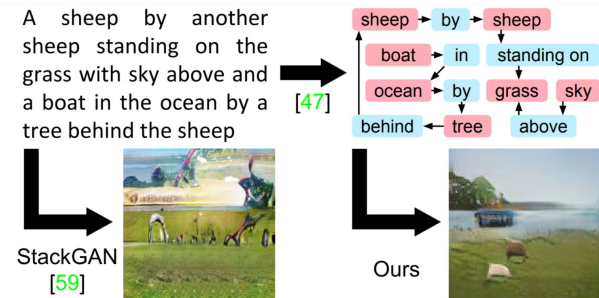
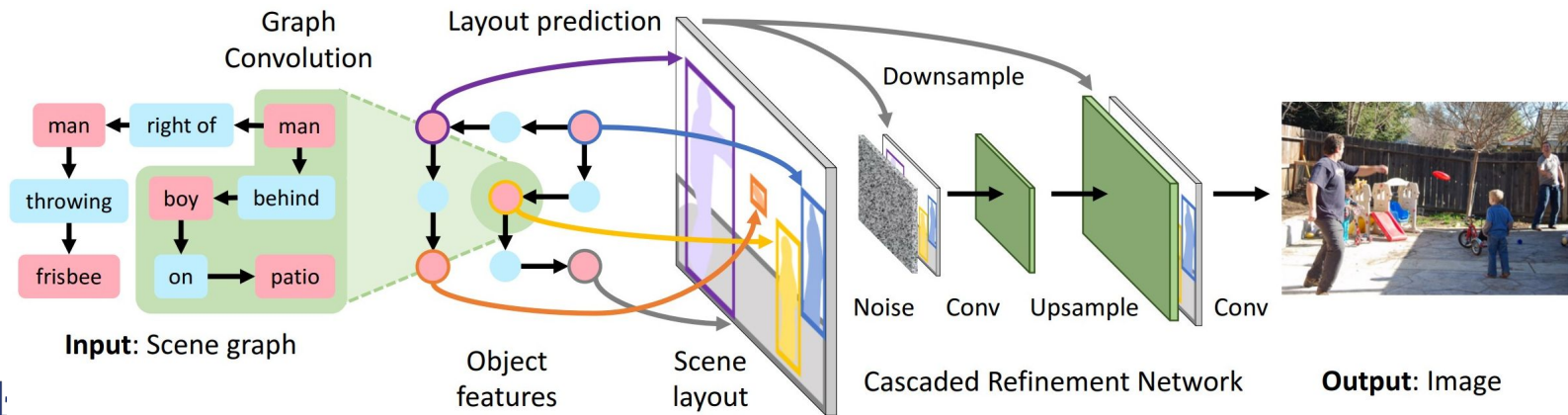


Figure 1. State-of-the-art methods for generating images from sentences, such as StackGAN [59], struggle to faithfully depict complex sentences with many objects. We overcome this limitation by generating images from *scene graphs*, allowing our method to reason explicitly about objects and their relationships.



Johnson, J., Gupta, A. and Fei-Fei, L., 2018. Image generation from scene graphs. In *CVPR* (pp. 1219-1228).

Applications (scene graph to image)

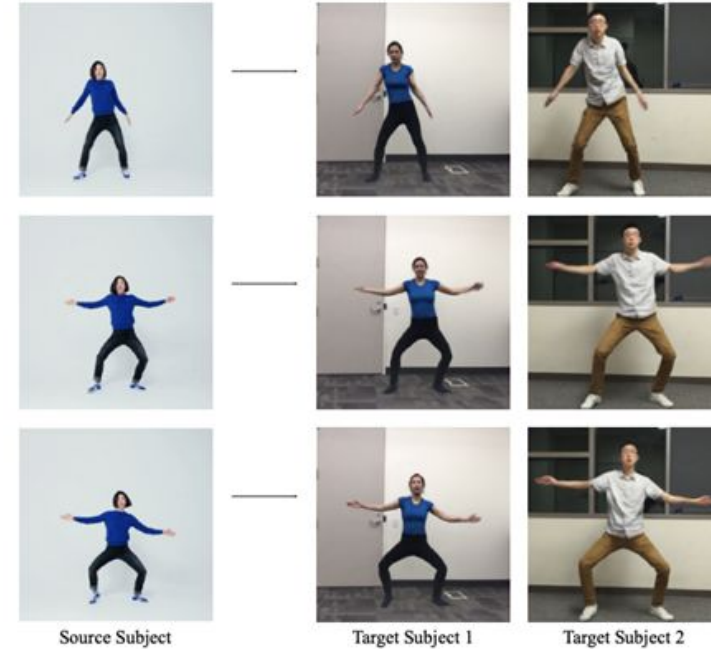
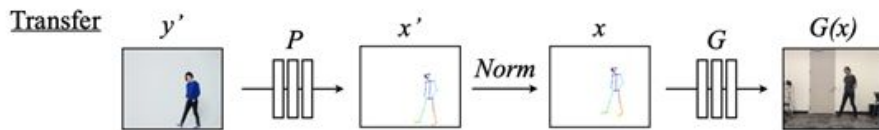
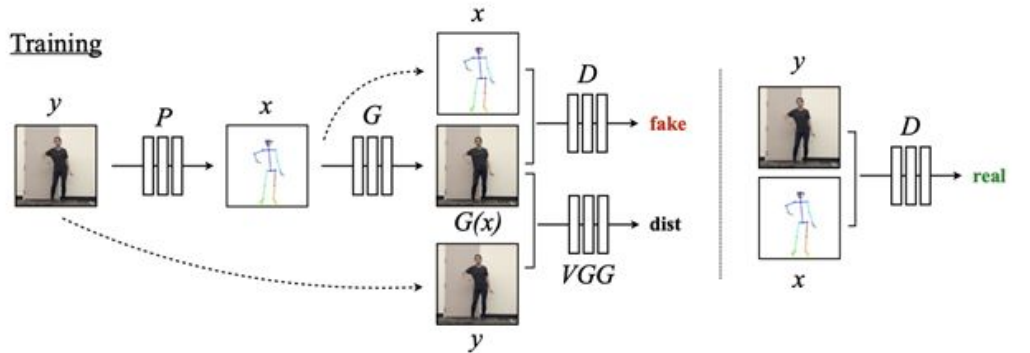


Figure 6. Images generated by our method trained on Visual Genome. In each row we start from a simple scene graph on the left and progressively add more objects and relationships moving to the right. Images respect relationships like *car below kite* and *boat on grass*.



Johnson, J., Gupta, A. and Fei-Fei, L., 2018. Image generation from scene graphs. In *CVPR* (pp. 1219-1228).

Applications (animations)



<https://www.youtube.com/watch?v=PCBTZh41Ris>

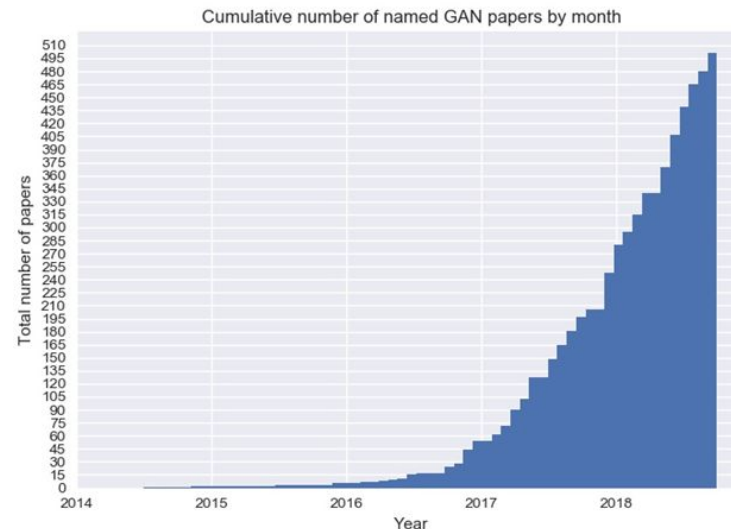
Implementation: https://github.com/llSource/Everybody_Dance_Now



Chan C, Ginosar S, Zhou T, Efros AA. Everybody dance now. arXiv preprint arXiv:1808.07371. 2018 Aug 22.

Advances in GANs

- Variations: Info-GAN¹, cGAN², Wasserstein-GAN³, f-GAN⁴, ...
- Inference with GANs⁵
- Feature learning (bi-GAN)⁶
- GANs + VAEs⁷
- Hundreds more GAN-derivations
- Database of GAN models:
<https://github.com/hindupuravinash/the-gan-zoo>



¹Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets.", NIPS, 2016.

²Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets.", arXiv, 2014

³Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan.", arXiv, 2017.

⁴Nowozin, Sebastian et al. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization." NIPS, 2016.

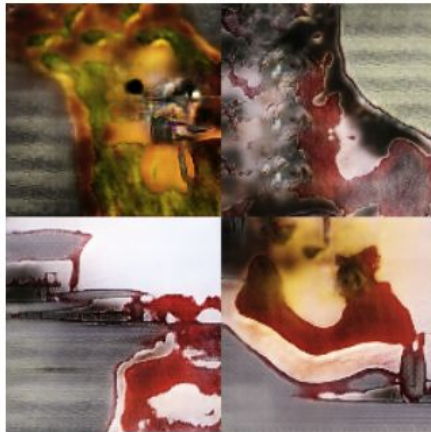
⁵Dumoulin, Vincent, et al. "Adversarially learned inference.", ICML 2017.

⁶Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning." ICLR, 2017

⁷Larsen, Anders Boesen Lindbo, et al. "Autoencoding beyond pixels using a learned similarity metric." arXiv, 2015.

Advances in GANs

- We still don't know exactly how they work!



Hands-on Exercise

- Google Colab Exercise
 - https://colab.research.google.com/drive/16PusNuycpfNRhm-NqsyIgp5rsb_dwAQQ
- Fill in the specified lines
 - Generator architecture
 - Discriminator architecture
 - Generator loss function
 - Discriminator loss function

