Generative Adversarial Networks

Lam Huynh

PhD student

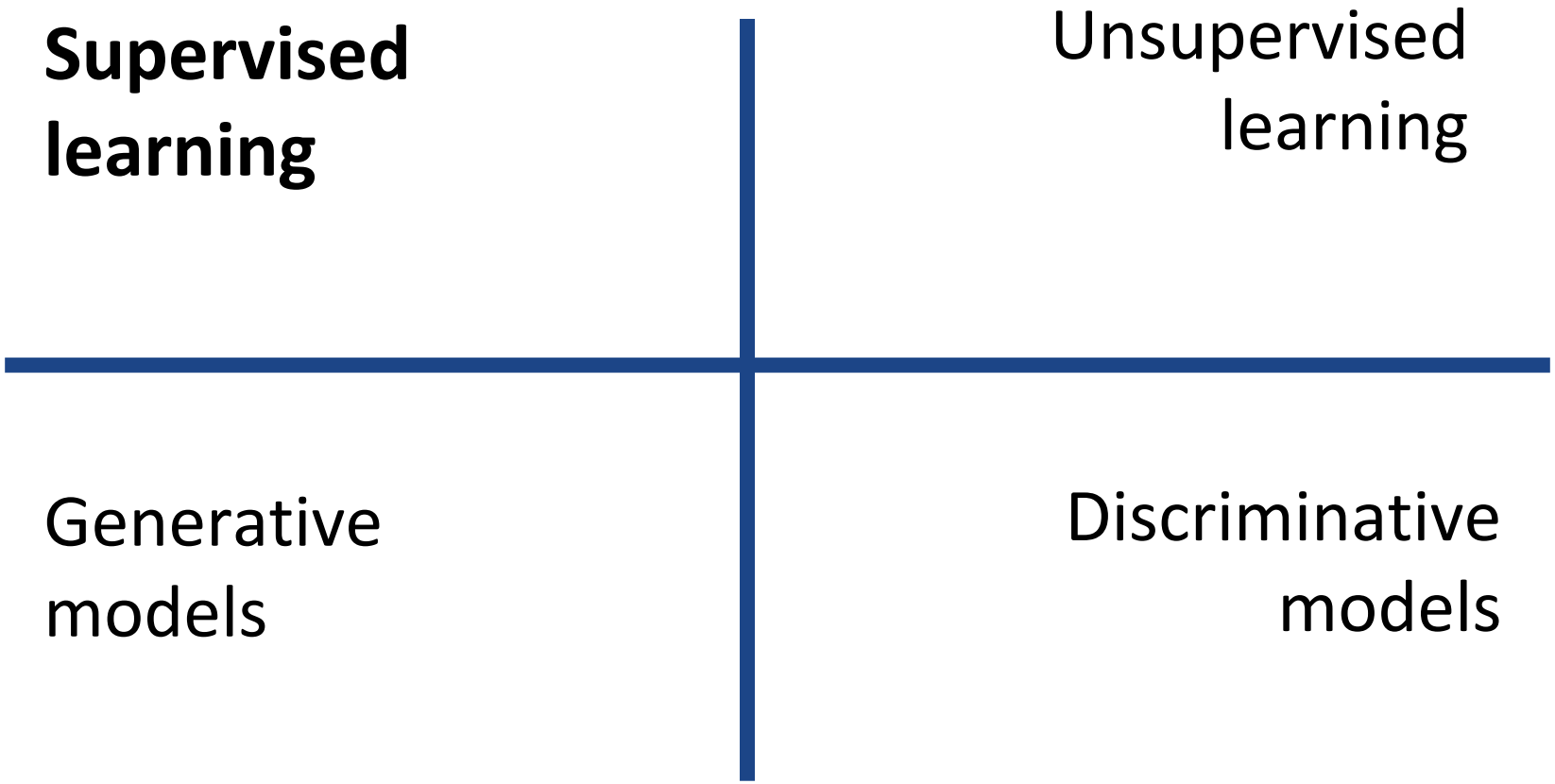
Center for Machine Vision and Signal Analysis, University of Oulu, Finland.

Outline

- **The learning types**
- So what is GANs and what make GANs ‘special’
- How does GANs actually work?
- DCGAN for the MNIST dataset
- GANs variations
- More readings
- Takeaway

The fundamental of GANs.

The learning types



Supervised learning

Data: (x, y)
x is data, y is label

Goal: Learn a function to map x -> y

Examples: Classification

Cat

Classification

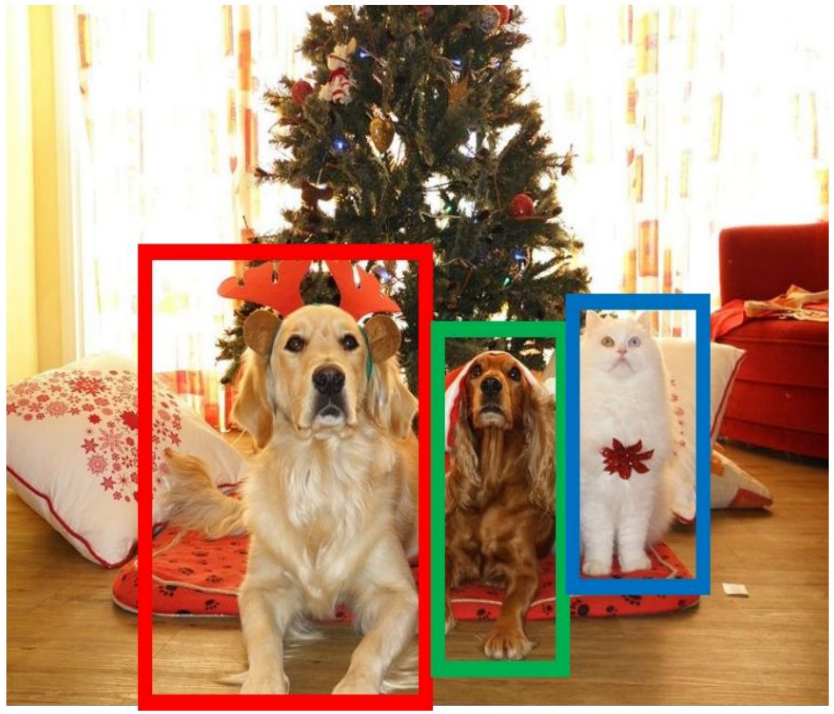
*slide from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

Supervised learning

Data: (x, y)
x is data, y is label

Goal: Learn a function to map x -> y

Examples: Object detection



DOG, DOG, CAT

Object Detection

*slide from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

Supervised learning

Data: (x, y)
x is data, y is label

Goal: Learn a function to map x -> y

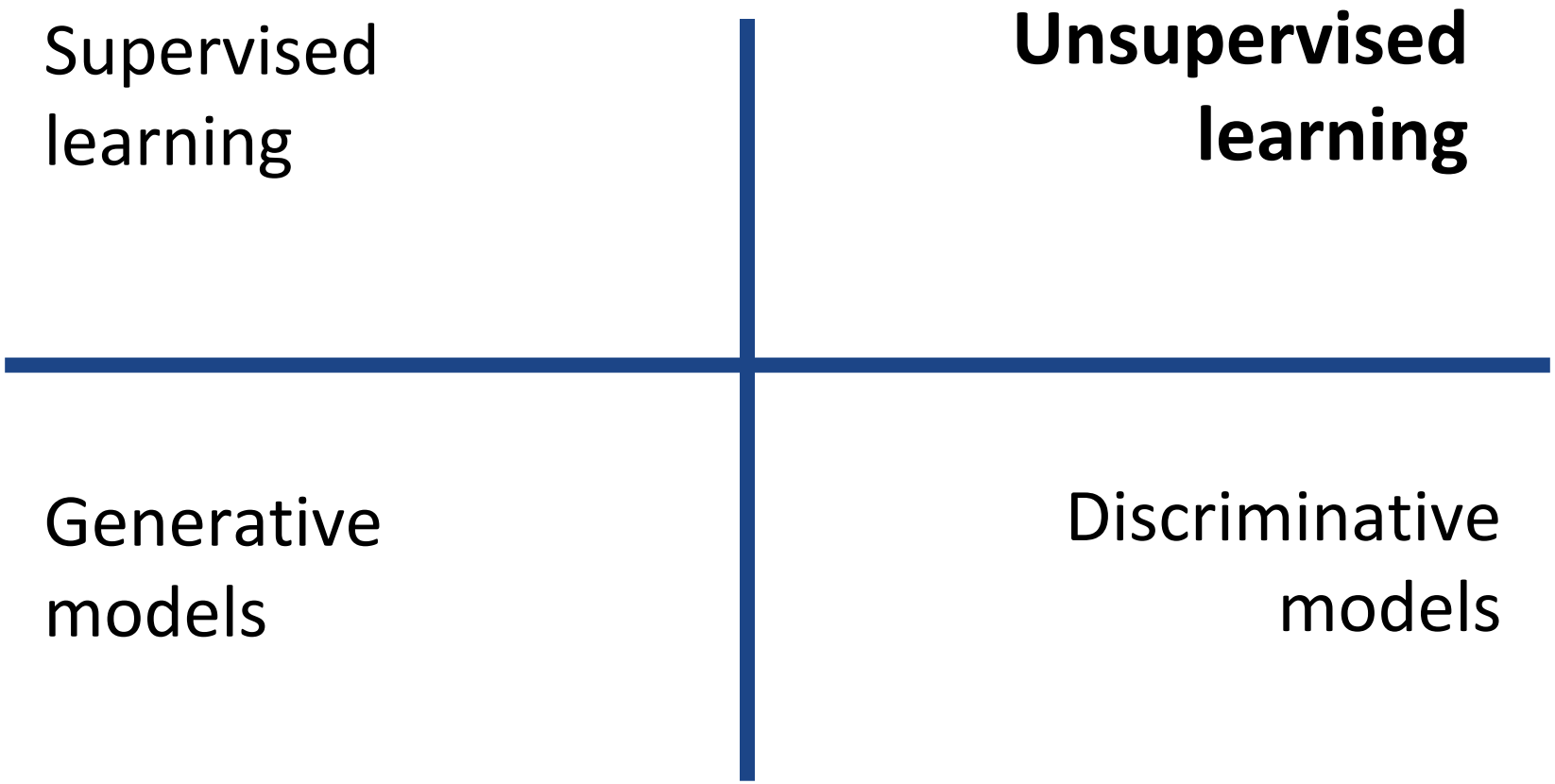
Examples: Image captioning

A cat sitting on a suitcase on the floor

Image captioning

*slide from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

The learning types

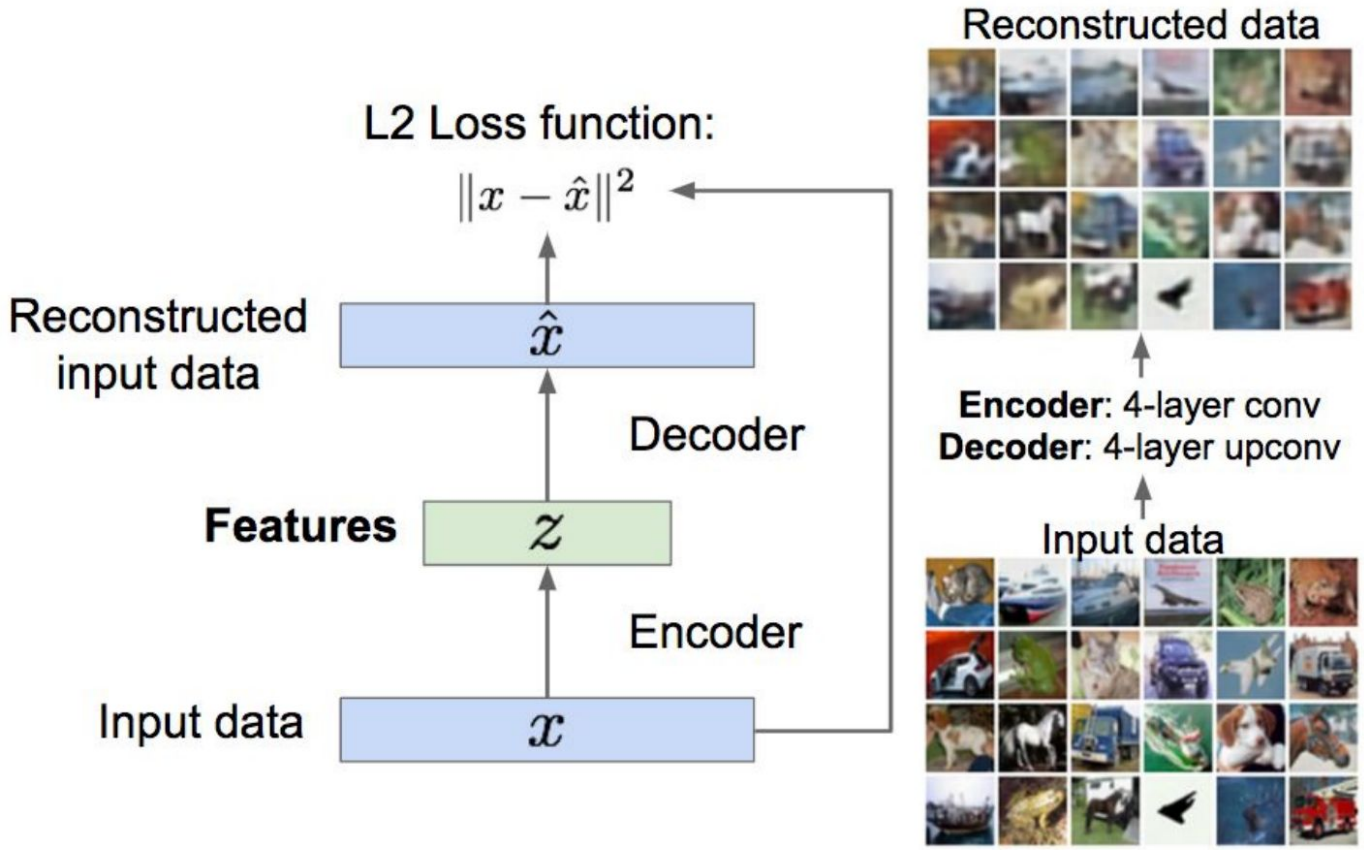


Unsupervised learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Feature learning



Autoencoders
(Feature learning)

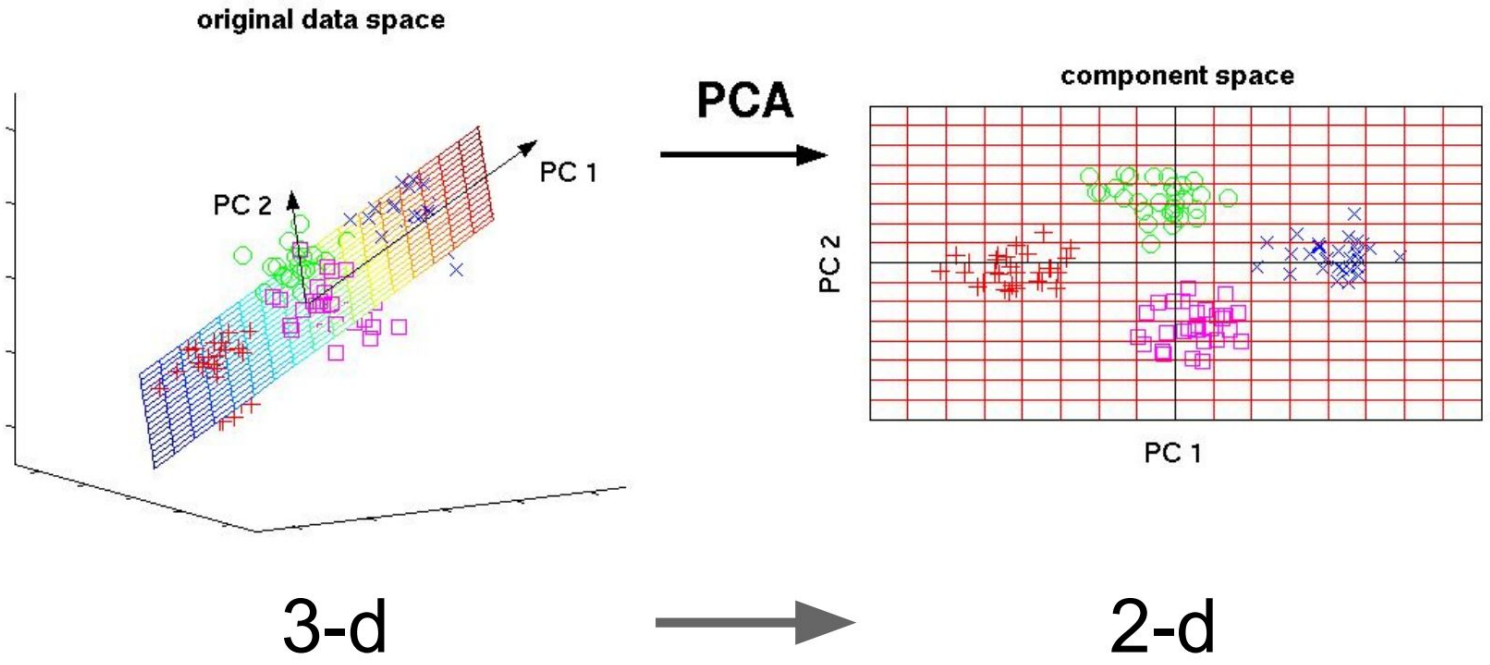
*slide from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

Unsupervised learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Dimensionality reduction



Principal Component Analysis
(Dimensionality reduction)

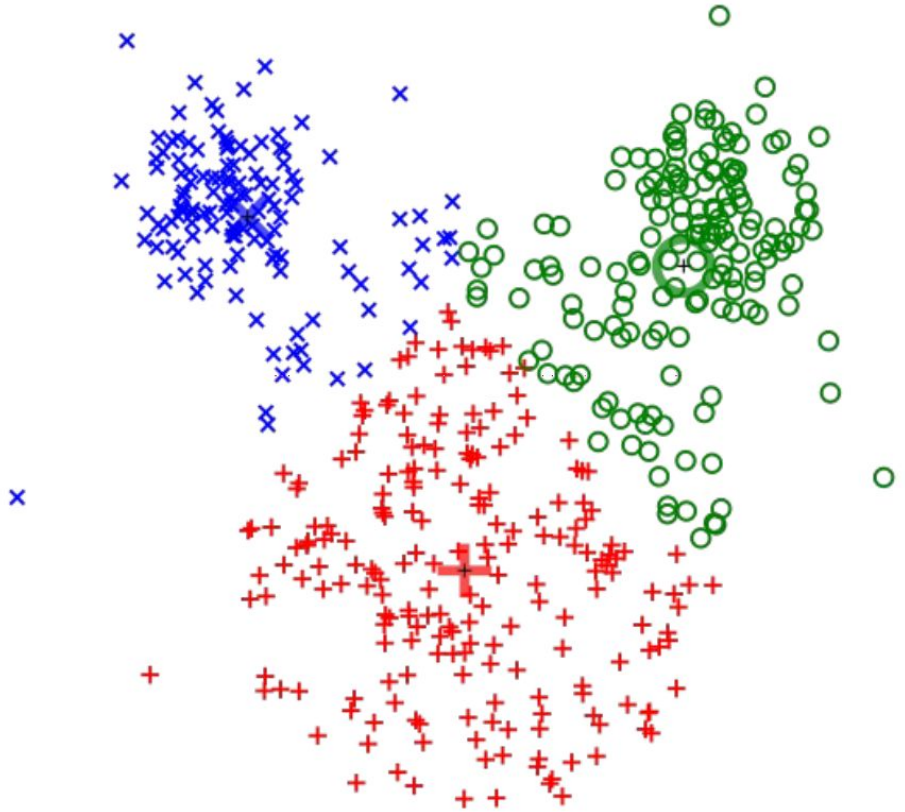
*slide from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

Unsupervised learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering



K-means clustering

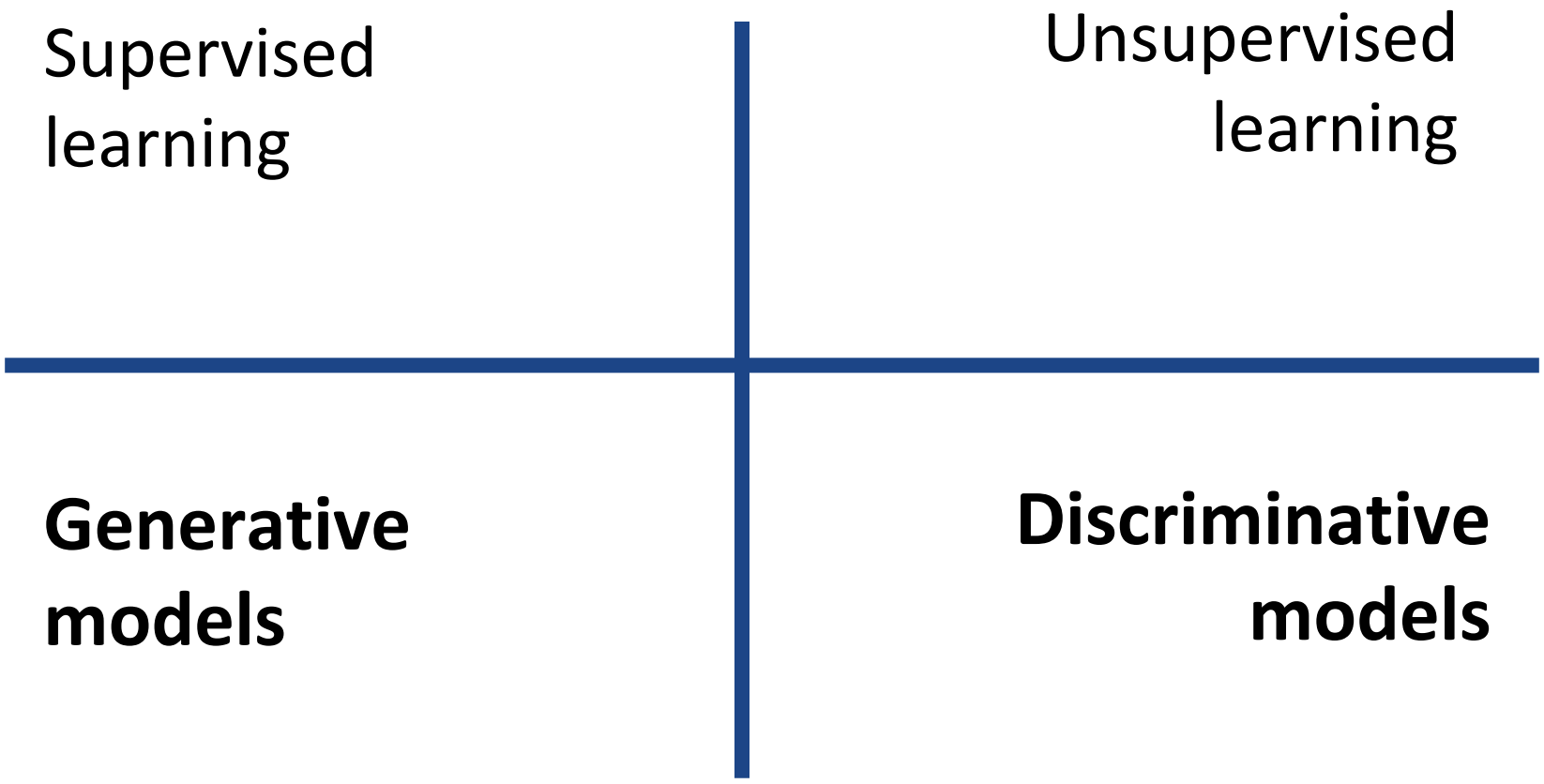
*slide from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

Supervised vs unsupervised learning

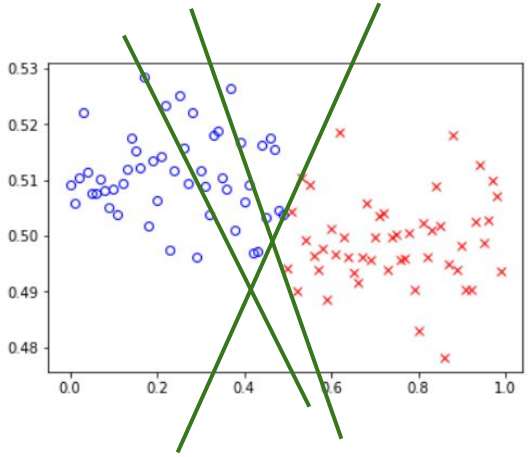
	Supervised	Unsupervised
Data	(x, y) x is data, y is label	(x) Just data, no labels!
Goal	Learn a function to map x -> y	Learn some underlying hidden *structure* of the data
E.g.	Classification, object detection, image captioning, etc.	Feature learning, dimensionality reduction, clustering
Adv	Relatively straightforward to learn the function f(x)	Easy to obtain a lot of training data
Dis	Required lots of labeled data	Loosely constraint, harder to converge
		=> understand structure of visual world (compelling)

*adapted from Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n Stanford

The learning types



Discriminative models



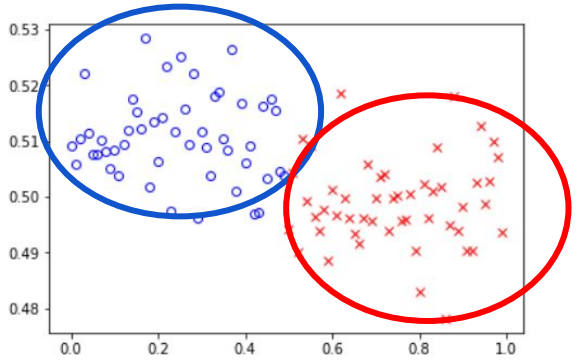
$p(y|x)$

Logistic regression, SVMs
Random forests
Classic neural networks
...

Adv: Easy to train, higher accuracy.

Dis: Fixate to the distribution of the training data.

Generative models



classes

features

$p(x|y)$ and $p(y)$

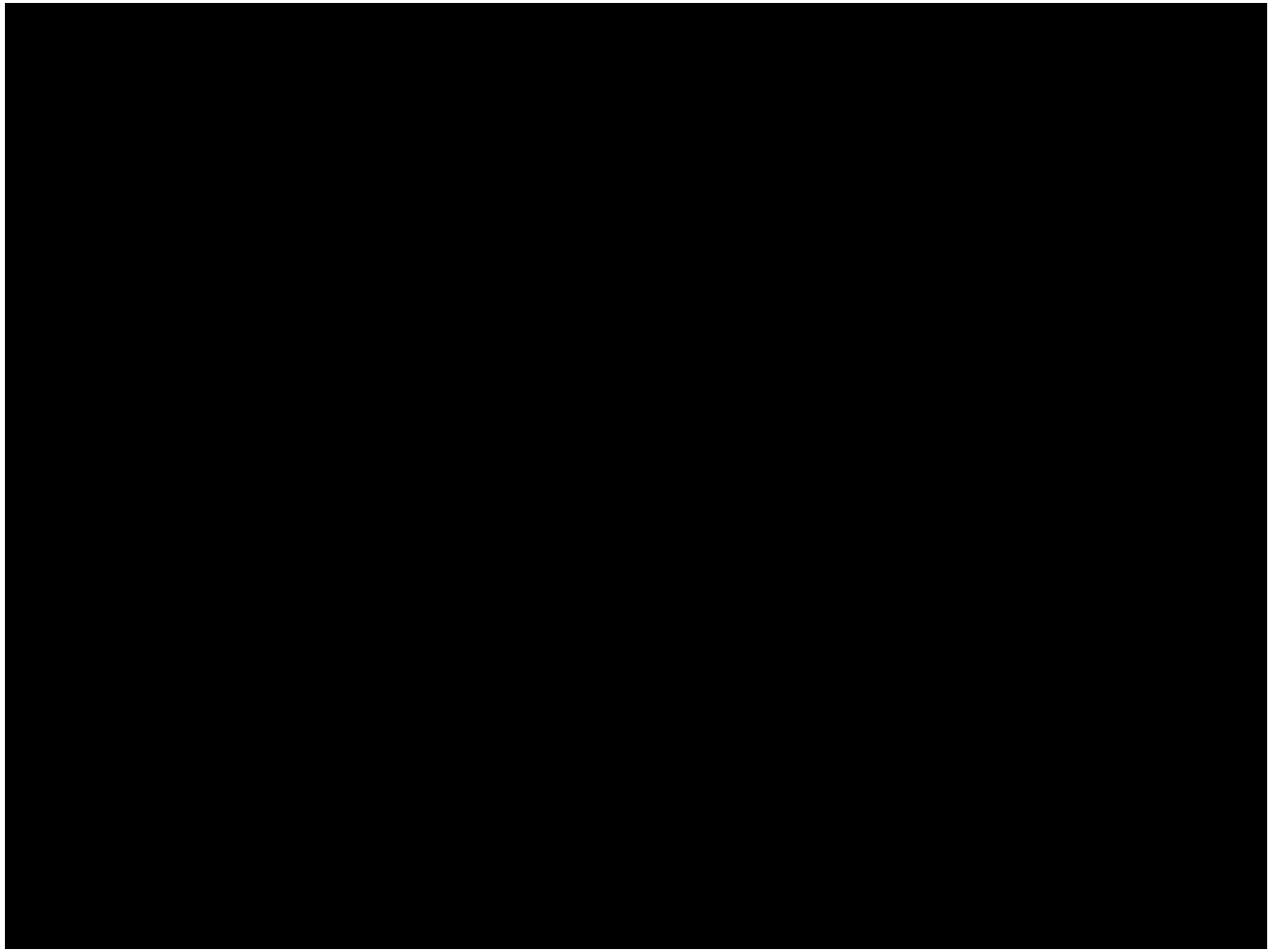
Generative adversarial networks (GANs)
GMM, Variational Autoencoders (VAE)
Flow-based models (e.g. PixelCNN, GLOW)
...

Adv: Have knowledge about the data distributions.

Dis: Hard to train, need lot of data.


Kahoot!

Why do we care about generative models?



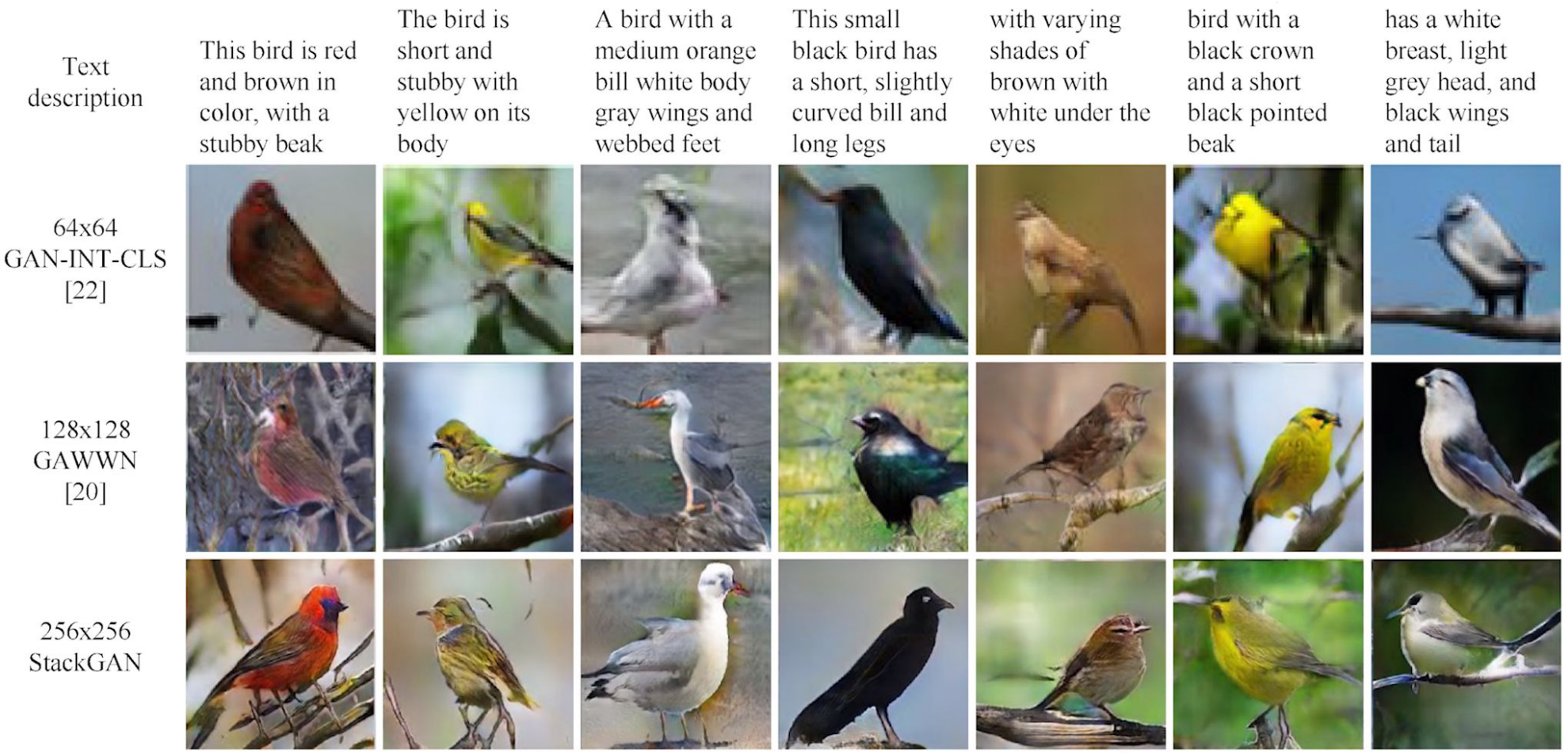
*source from Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018
Tero Karras et al.

Why do we care about generative models?

GauGAN demo

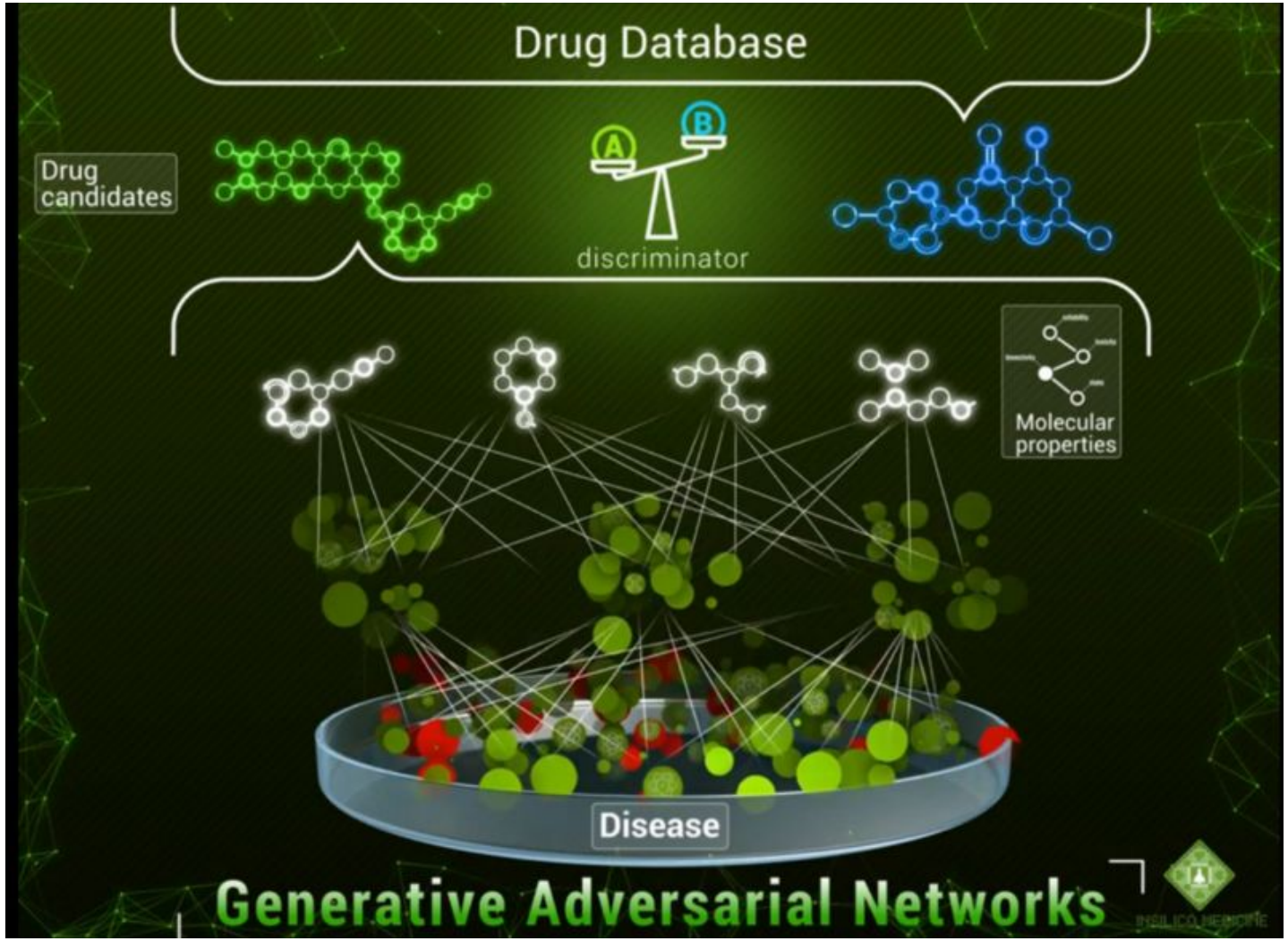
*source from GauGAN, CVPR 2019
Park et al.

Why do we care about generative models?



*source from StackGAN: Text to Photo-realistic Image Synthesis, ICCV 2017. Han Zhang et al.

Why do we care about generative models?



*source from Insilico Medicine.

Why do we care about generative models?

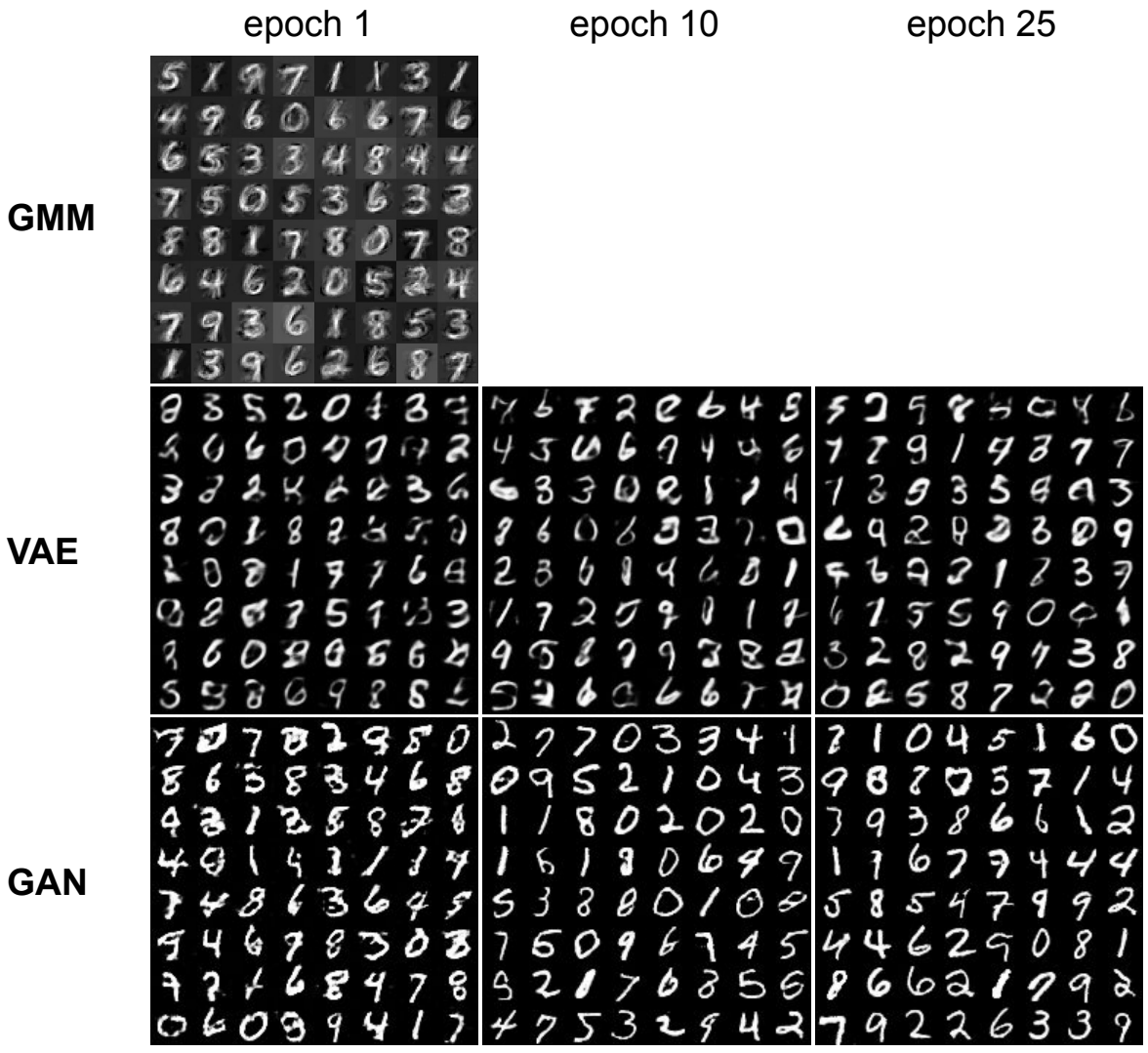
- Realistic article spinning by AI
- What if the news or articles you read are automatically written by a machine?
- What if a machine be able to create contents that can catch your attention, make you think that these are worth sharing or even convince you to spend the money to buy these contents?
- What if the media and entertainment products like movies, TV shows, music, games, etc are generated by the machine that based on your user personality? Will you get addicted to those?

Outline

- The learning types
- **So what is GANs and what make GANs ‘special’**
- How does GANs actually work?
- DCGAN for the MNIST dataset
- GANs variations
- More readings
- Takeaway

What make GANs special?

- GANs is a type of generative model. GANs:
  - Very good at generating realistic samples.
  - New training scheme call adversarial training.



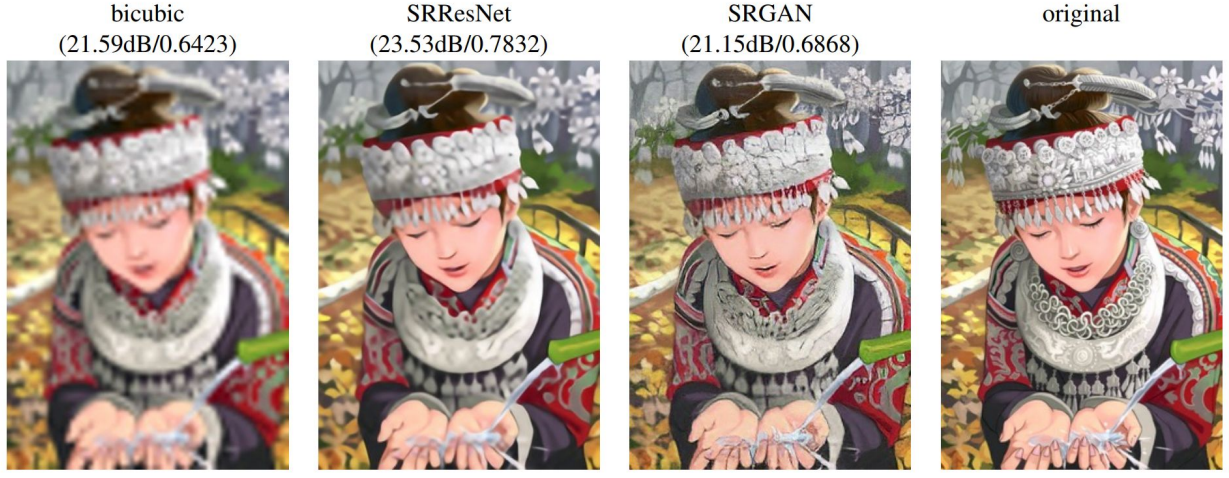
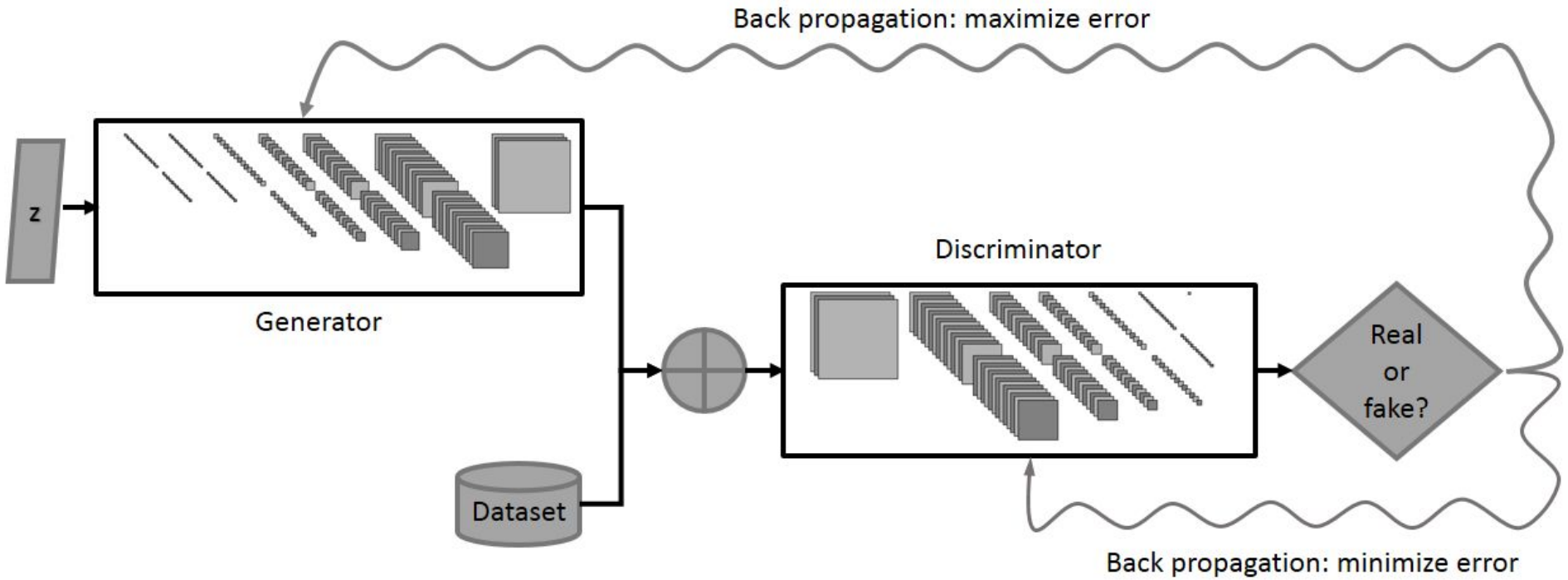


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

GANs components

- GANs consists of two modules: Generator (G) and Discriminator (D).
  - Novel idea of “adversarity training”, which means G and D compete against each other in the training.
  - (G) strives to generate the most realistic sample it can.
  - (D) manages to recognize the real and fake (generated) samples.



GANs analogy

Counterfeiter VS Police

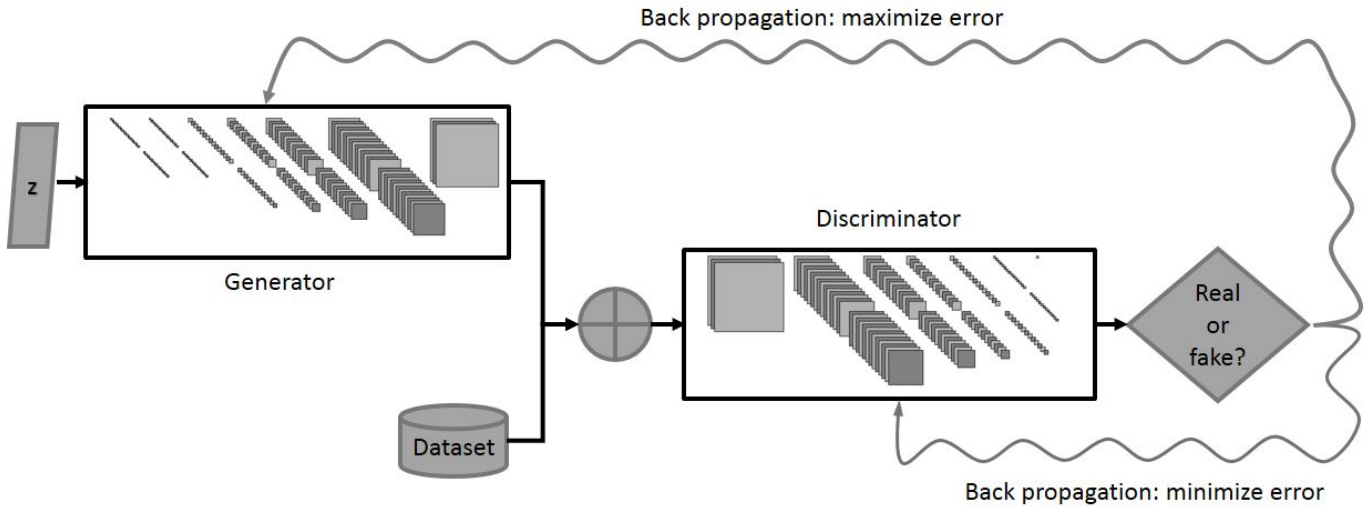
Quote from the original GANs paper: *"The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles."*

Outline

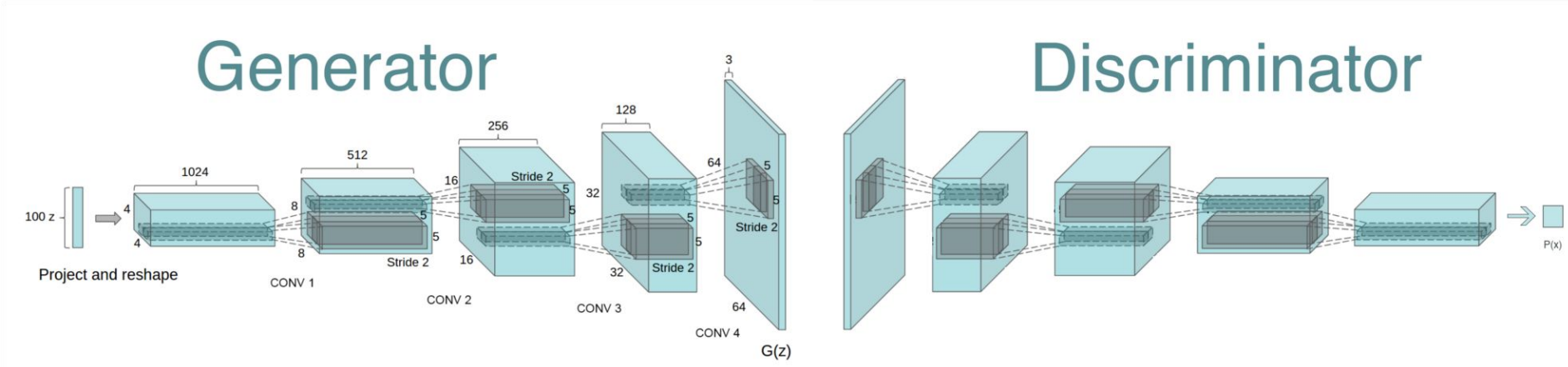
- The learning types
- So what is GANs and what make GANs ‘special’
- **How does GANs actually work?**
- DCGAN for the MNIST dataset
- GANs variations
- More readings
- Takeaway

GAN vs DCGAN

GAN (2014)
(Vanilla GAN)



DCGAN (2015)
(Deep Convolutional Generative Adversarial Networks)



=> Popular successor of vanilla GAN, most modern GANs are based on **DCGAN**

Dissect DCGAN

- General architecture of the DCGAN **Generator**:
  - Starting with a latent vector Z
  - Reshaping and projecting into a 3D tensor
  - Passing through several “transpose_conv_2d layers” to upsample the spatial dimensions of our feature maps
  - Final output is the generated image with size H x W x C

```python
class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # state size. (ngf*8) x 4 x 4
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            # state size. (ngf*4) x 8 x 8
            nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # state size. (ngf*2) x 16 x 16
            nn.ConvTranspose2d( ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # state size. (ngf) x 32 x 32
            nn.ConvTranspose2d( ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. (nc) x 64 x 64
        )

    def forward(self, input):
        return self.main(input)
```

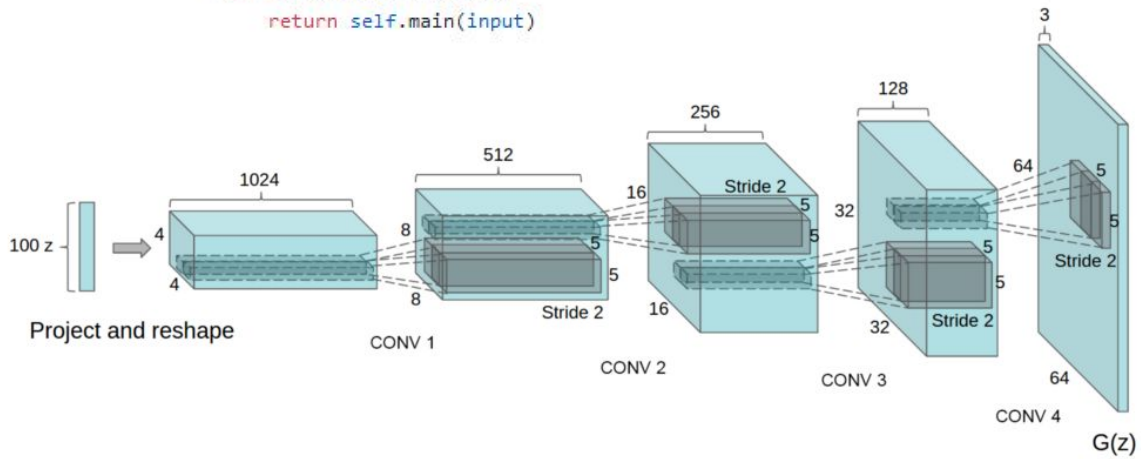
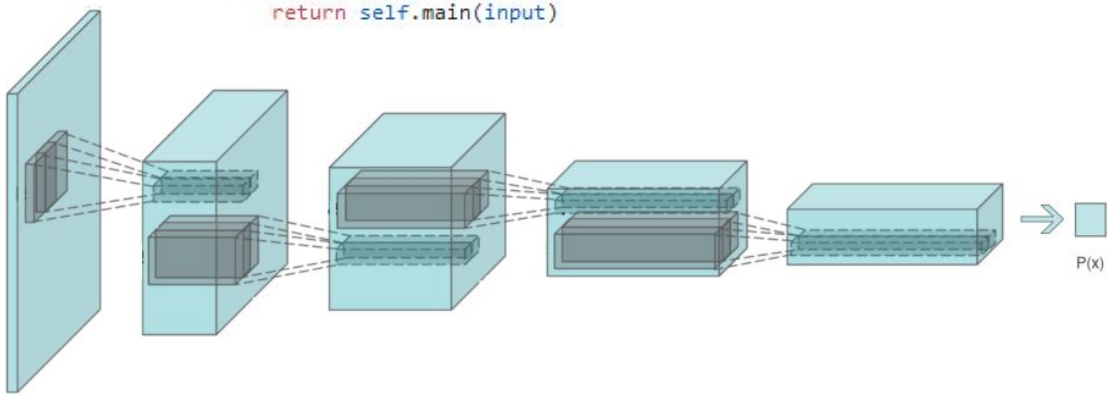


*figure from DCGANs, Alec Radford et al. 2016

Dissect DCGAN

```
class Discriminator(nn.Module):
    def __init__(self, ngpu):
        super(Discriminator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is (nc) x 64 x 64
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf) x 32 x 32
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*2) x 16 x 16
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*4) x 8 x 8
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*8) x 4 x 4
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input)
```



- General architecture of the DCGAN **Discriminator**:
  - Input placeholder is H x W x C image (generated_image.shape == real_image.shape)
  - Passing through several convolutional layers
  - Performing binary logistic regression at the end

*figure from DCGANs, Alec Radford et al. 2016

Properties of DCGAN

- ***Transpose convolutional layers** for generator.
- ***Batch Normalization** used to normalize input volumes at every layers.
- **Fully convolutional network** which contains only convolutional layer with stride > 1 and no pooling (less computation)
- **Adam** optimizer (computes individual adaptive learning rates for different parameters (weights) from estimates of first and second moments of the gradients)
- **Leaky ReLU** for discriminator and **ReLU** for generator.

Transpose convolutional layers

- Transpose convolutional block a.k.a deconvolutional layer (bad naming, nothing to do with deconvolving operation)
- Convolving and upsampling the spatial dimension of the feature maps
- Transferring the data from feature maps into spatial dimension

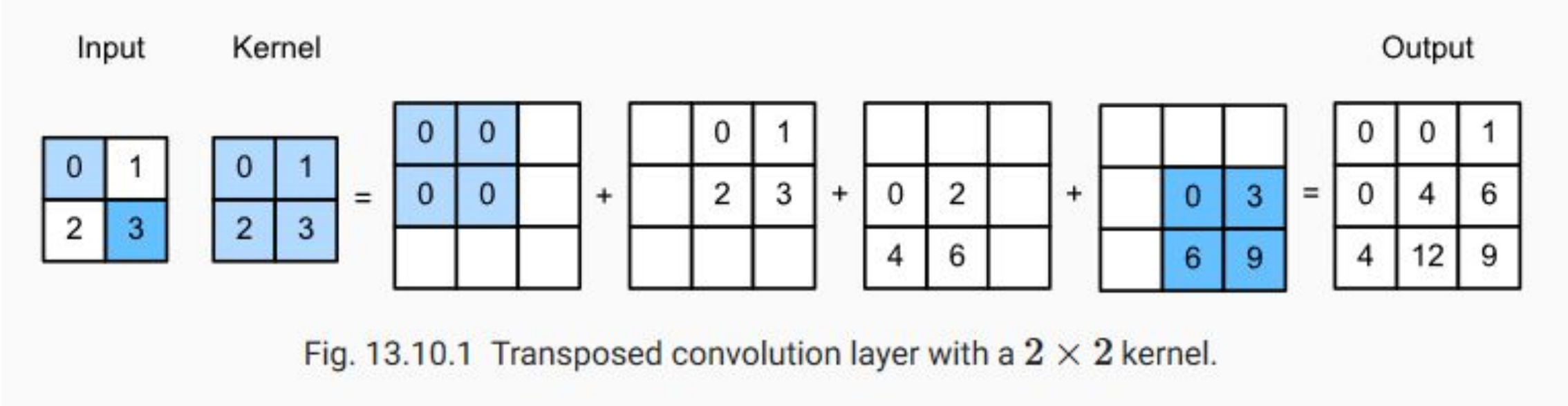


Fig. 13.10.1 Transposed convolution layer with a 2×2 kernel.

Batch Normalization

- Batch normalization used to normalize input volumes at every layers.
- Training:

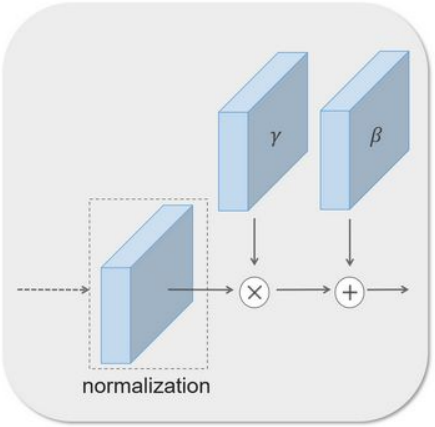
X_B = next batch of data

$\mu_B = \text{mean}(X_B)$

$\sigma_B = \text{std}(X_B)$

$Y_B = (X_B - \mu_B)/\text{sqrt}(\sigma_B^2 + \varepsilon)$

$Y = \gamma \hat{X}_B + \beta$



- Testing:

μ, σ^2 *collected during training*

$\hat{x}_{test} = (x_{test} - \mu)/\sqrt{\sigma^2 + \varepsilon}$

$y_{test} = \gamma \hat{x}_{test} + \beta$

Loss function

- Let first define the loss function for the discriminator, which simply is the binary cross-entropy:

$L_D = - [y \log(\hat{y}) + (1 - y) \log (1 - \hat{y})]$

$L_D = \{E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]\}$ (for training batch) (1)

- We label as 0.0 for the generated images and 1.0 for the real images

$L_G = - L_D$ (minimax game) (2)

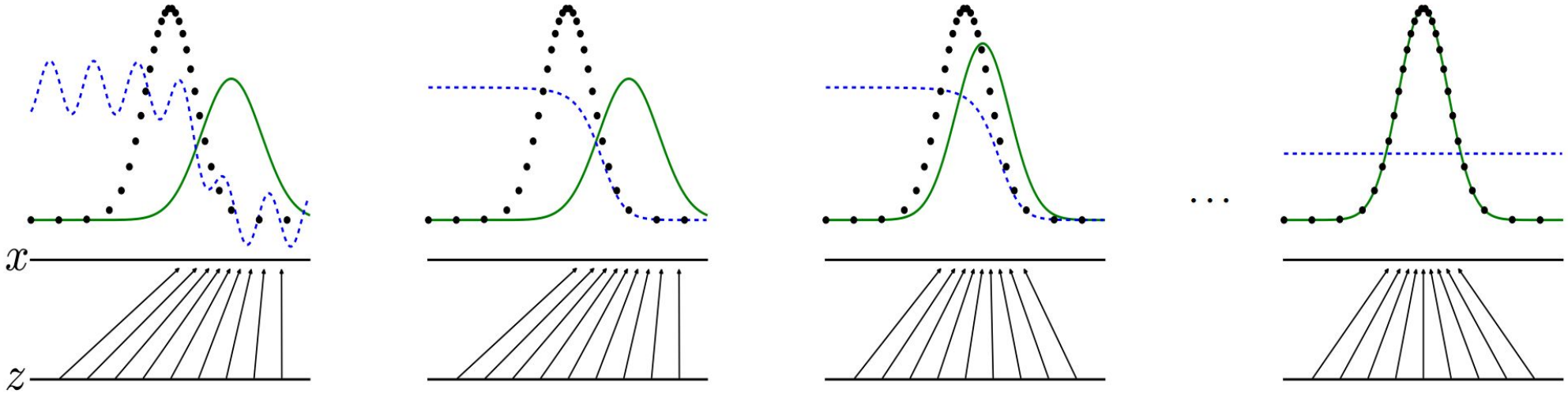
- Quote from Ian Goodfellow: *"In the minimax game, the discriminator minimizes a cross-entropy, but the generator maximizes the same cross-entropy. This is unfortunate for the generator, because when the discriminator successfully rejects generator samples with high confidence, the generator's gradient vanishes."*

=> It can be seen that (2) is not working well in practice.

- Therefore, if we label 1.0 for both the real and generated images, the generator loss will become: (better)

$L_G = - E_z[\log D(G(z))]$ (3)

Loss function



- We want to iteratively update the generator and discriminator weights
  - Updating the generator increase its ability to generate more realistic samples
  - On the other hands, updating the discriminator make it better at distinguishing between the real and generated images
- Apparently, we want to reach the final stage, which generator is very well in describing the data distribution, while discriminator is only 50% correctly classified its input samples.

Outline

- The learning types
- So what is GANs and what make GANs ‘special’
- How does GANs actually work?
- **DCGAN for the MNIST dataset**
- GANs variations
- More readings
- Takeaway

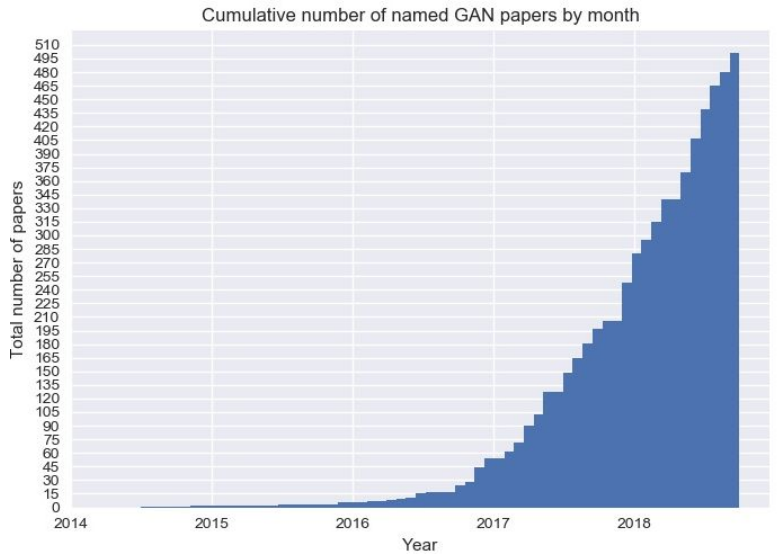
(Pytorch DCGAN example code)

Outline

- The learning types
- So what is GANs and what make GANs ‘special’
- How does GANs actually work?
- DCGAN for the MNIST dataset
- **GANs variations**
- More readings
- Takeaway

The explosion of interest in GANs

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks



*source from [the GANs Zoo]

Problem with GANs

- Comparing to other DNN, GAN models can suffer badly from following problems:
    - **Non-convergence**: the models never converge and worse they become unstable.
    - **Mode collapse**: the generator produces a single or limited modes.
    - **Slow training**: the gradient to train the generator vanished.

=> New models and techniques appear to solve these problems

WGAN (Wasserstein GANs)

- The **Wasserstein** distance is used to measure the distance between two probability distributions.
- Applying to create a new loss function which include the **Wasserstein** distance (generated & real).
- As pointing out in [the paper](), Wasserstein metric provides a smooth measure, which is make the training using gradient descents more stable.

=> giving GAN more meaningful loss function.

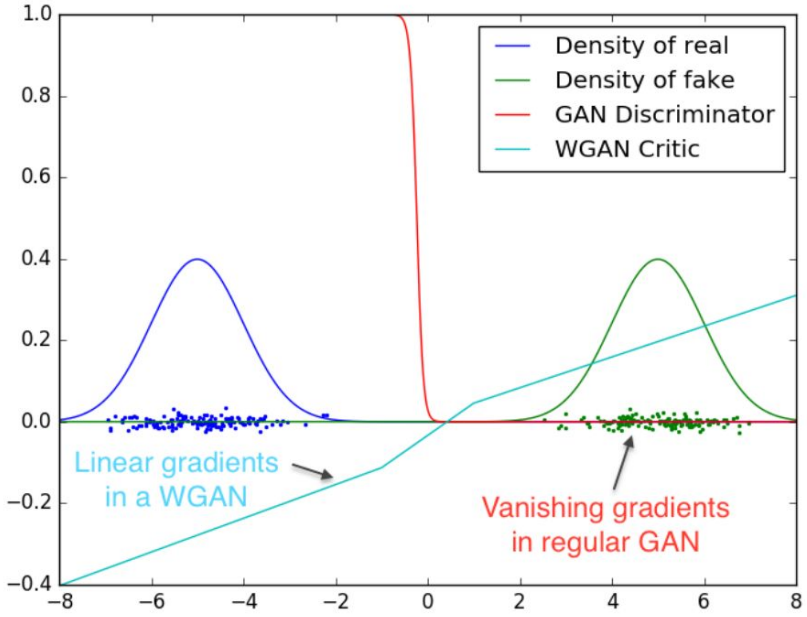


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

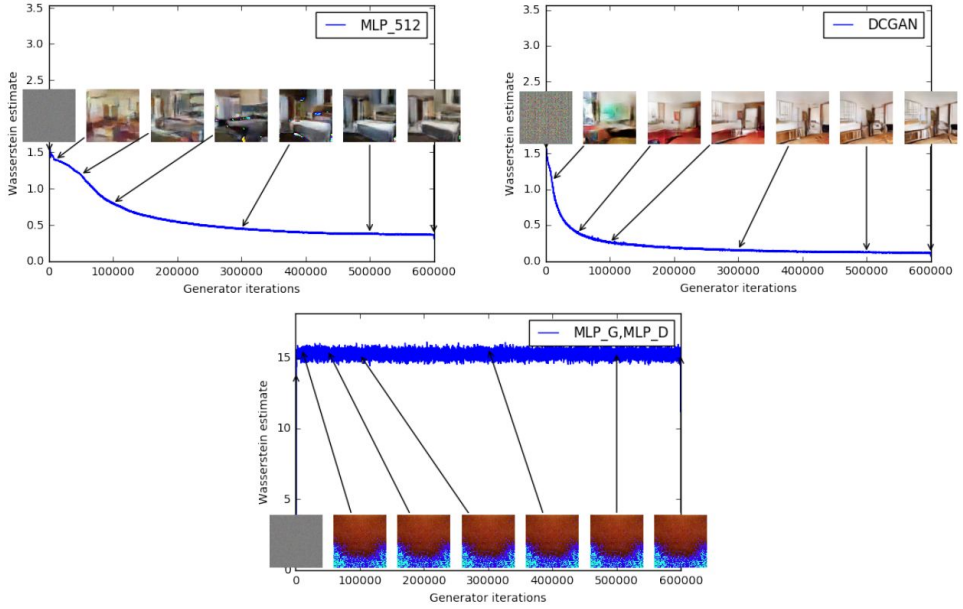
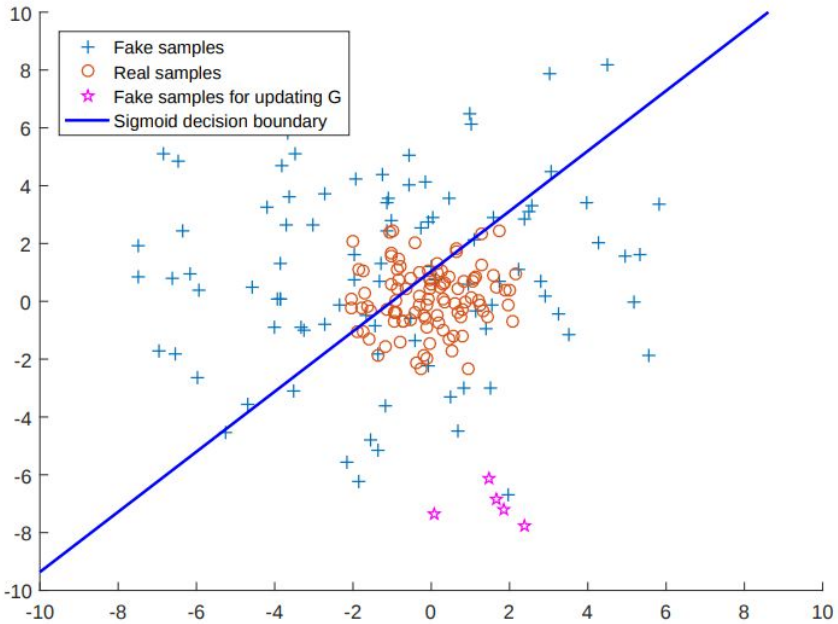


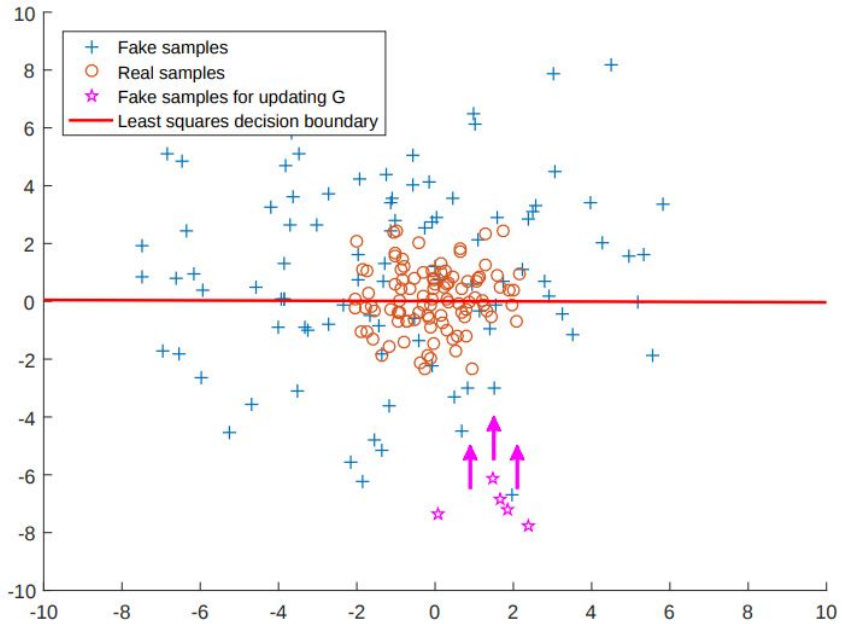
Figure 3: Training curves and samples at different stages of training. We can see a clear correlation between lower error and better sample quality. Upper left: the generator is an MLP with 4 hidden layers and 512 units at each layer. The loss decreases constistently as training progresses and sample quality increases. Upper right: the generator is a standard DCGAN. The loss decreases quickly and sample quality increases as well. In both upper plots the critic is a DCGAN without the sigmoid so losses can be subjected to comparison. Lower half: both the generator and the discriminator are MLPs with substantially high learning rates (so training failed). Loss is constant and samples are constant as well. The training curves were passed through a median filter for visualization purposes.

LSGAN (Least Squares GANs)

- Vanilla GAN using the sigmoid function for L_D , which unfortunately is saturated very fast. When data x are large, gradient of D move to zero => vanishing gradient in D hinders the learning of G.
- Changing to use the least squares loss gives smooth and non-saturation gradient in D.

$$L_D^{LSGAN} = E[(D(x) - 1)^2] + E[D(G(z))^2]$$
$$L_G^{LSGAN} = E[(D(G(z)) - 1)^2]$$





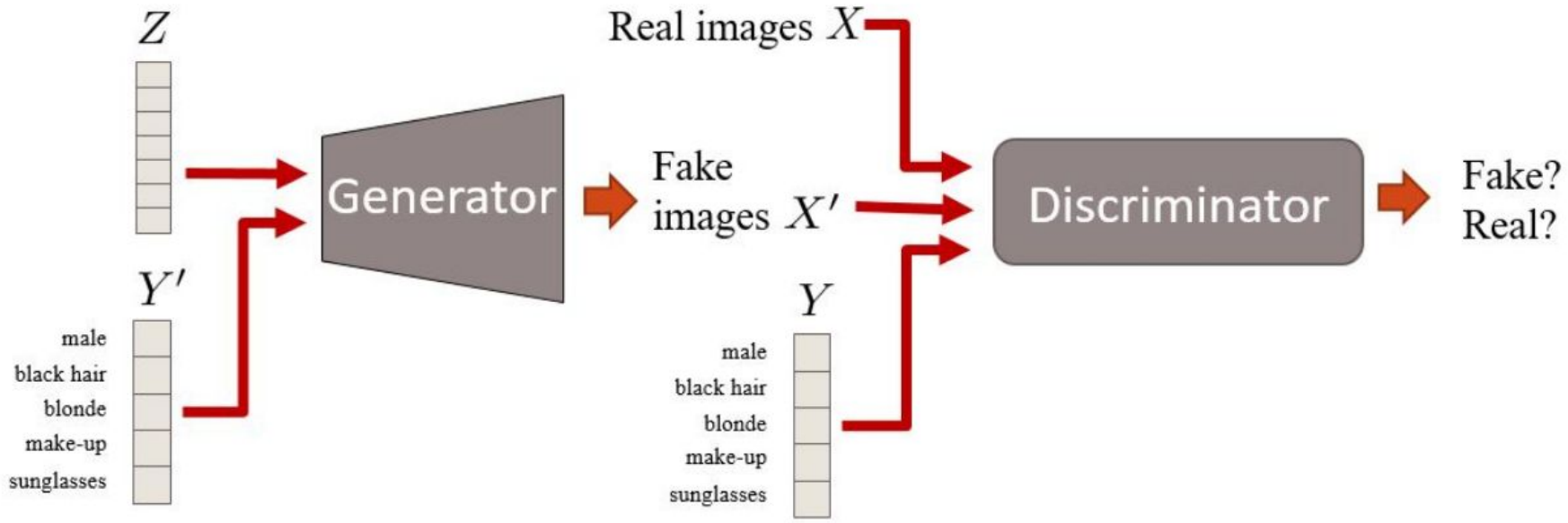
*source from [LSGAN by Xudong Mao et al. 2017]

CGAN (Conditional GANs)

- Conditional GANs add constraints into the generator and discriminator in a GAN.
- Vanilla GAN has a generator G(Z) where Z is a random vector and a discriminator D(G(z)).
- CGAN add more information to create a generator G(Z, Y') and a discriminator D(G(Z, Y'), Y)..

=> CGAN model can generate specific type of outputs

- Examples:
  - Generating faces with certain age, emotion, and race.
  - Another example is StackGAN.

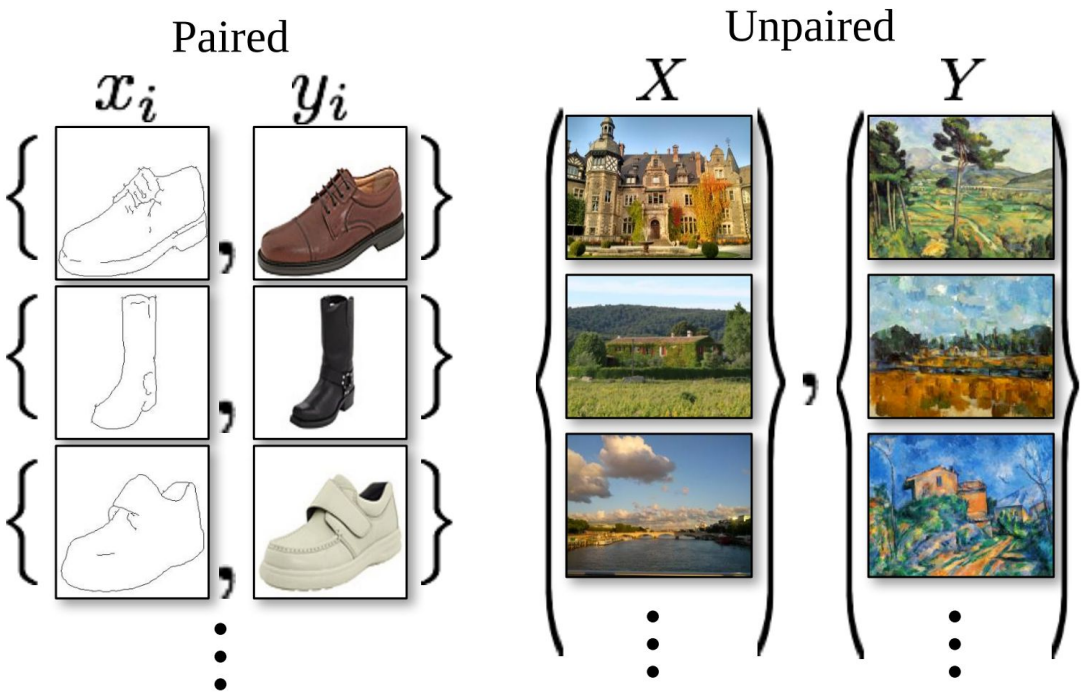


CycleGAN

- Image-to-image translation

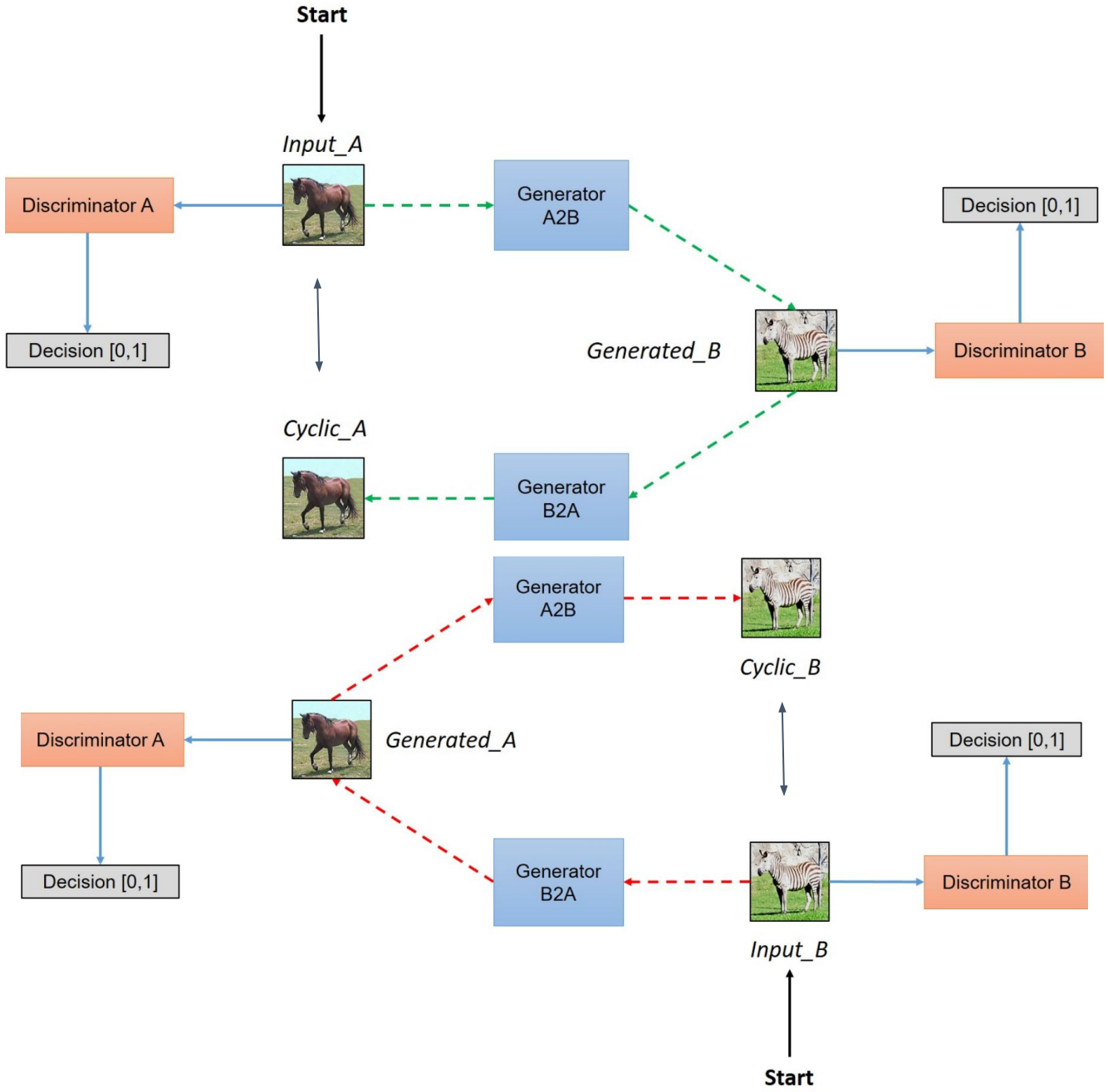
A happy hitman

Starry Doge



*source from Hardik Bansal, Archit Rathore blog and CycleGAN paper

CycleGAN



*source from [Hardik Bansal, Archit Rathore blog]

CycleGAN

Input

CycleGAN

Output

*source from [CycleGAN paper]

SAGAN (Self Attention GAN)

- Problem: small receptive field can not cover larger structure (increase filter size or network depth create harder GAN to train).
- Idea: SAGAN only uses the feature map region on the highlight area. These region has larger receptive field and the context is more focus and more relevant.

  => SAGAN model performs very good on the ImageNet.

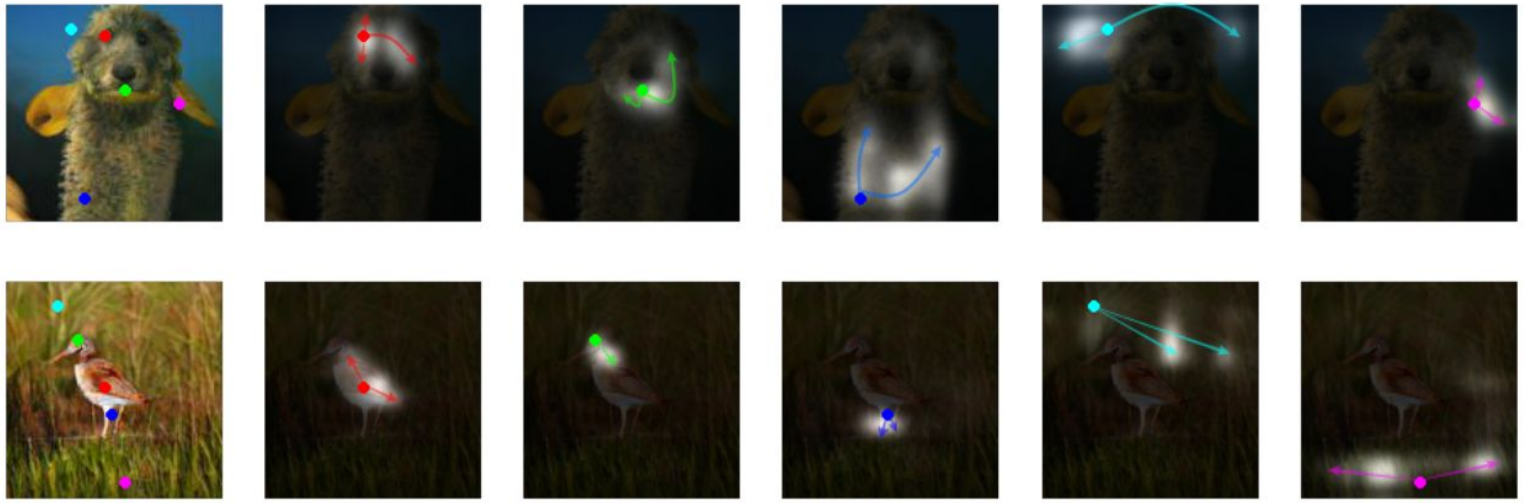
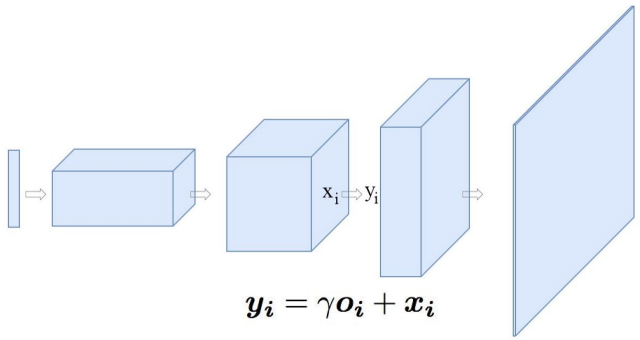


Figure 1: The proposed SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other five images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions.

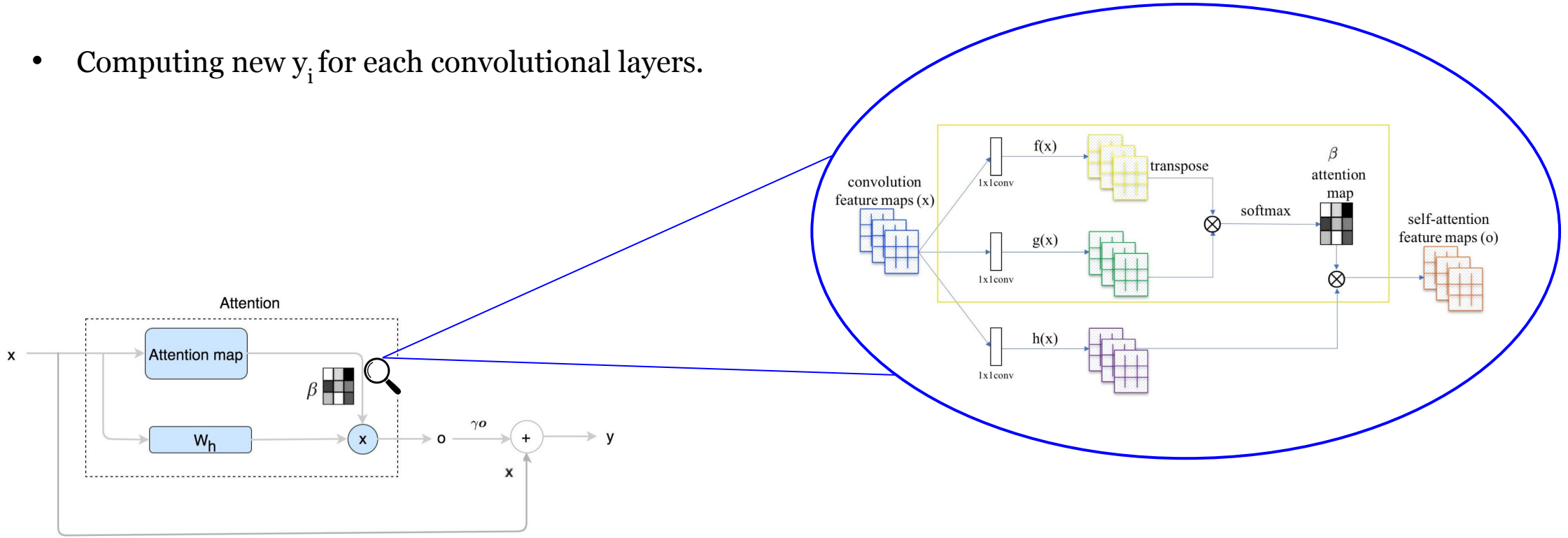
*source from SAGAN, Han Zhang et al. 2018

SAGAN (Self Attention GAN)

- Using the self-attention mechanism, the SAGAN model can refine each spatial location output by computing new o

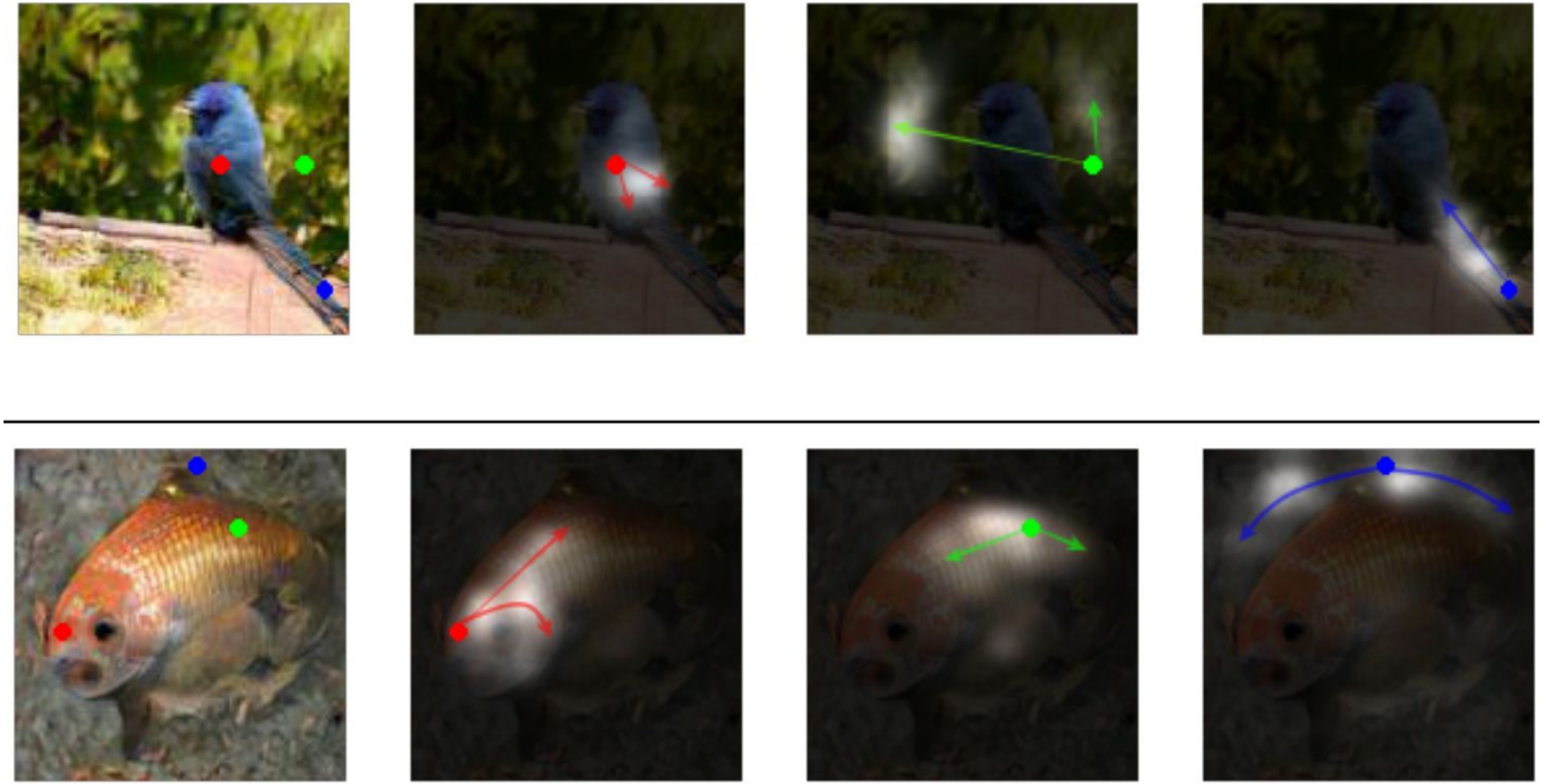


- Computing new y_i for each convolutional layers.



*source from SAGAN, Han Zhang et al. 2018

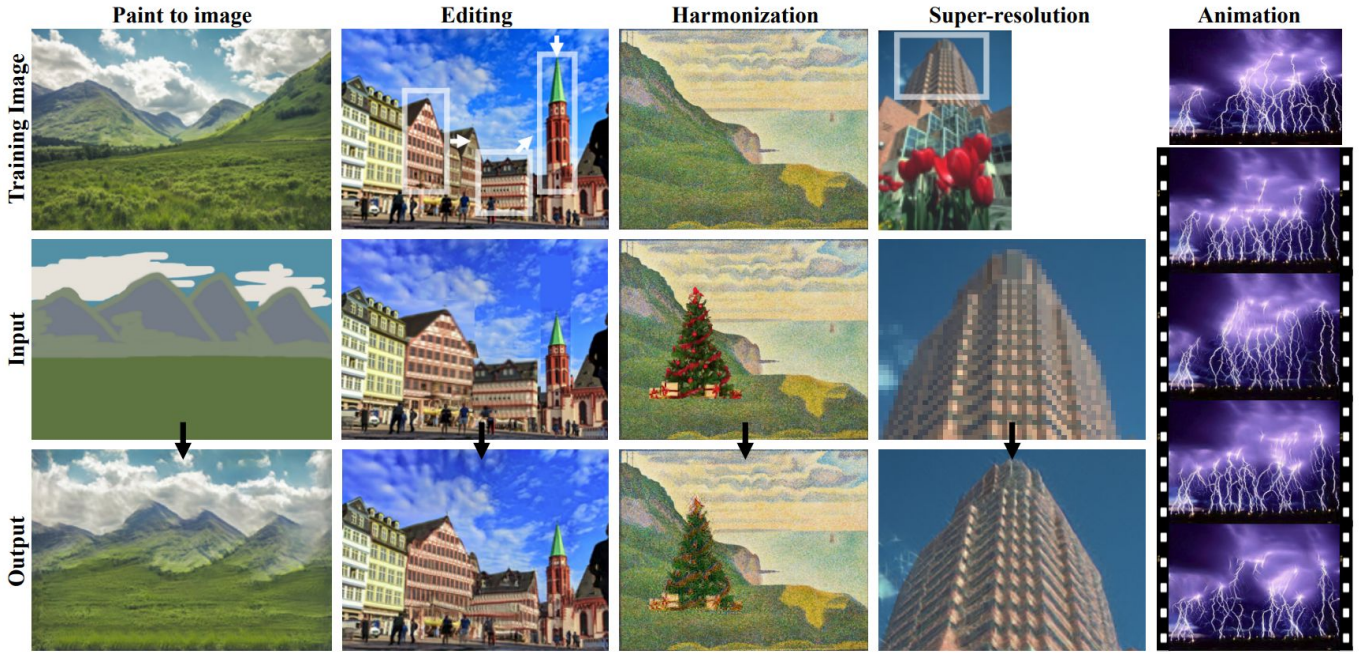
SAGAN (Self Attention GAN)



*source from SAGAN, Han Zhang et al. 2018

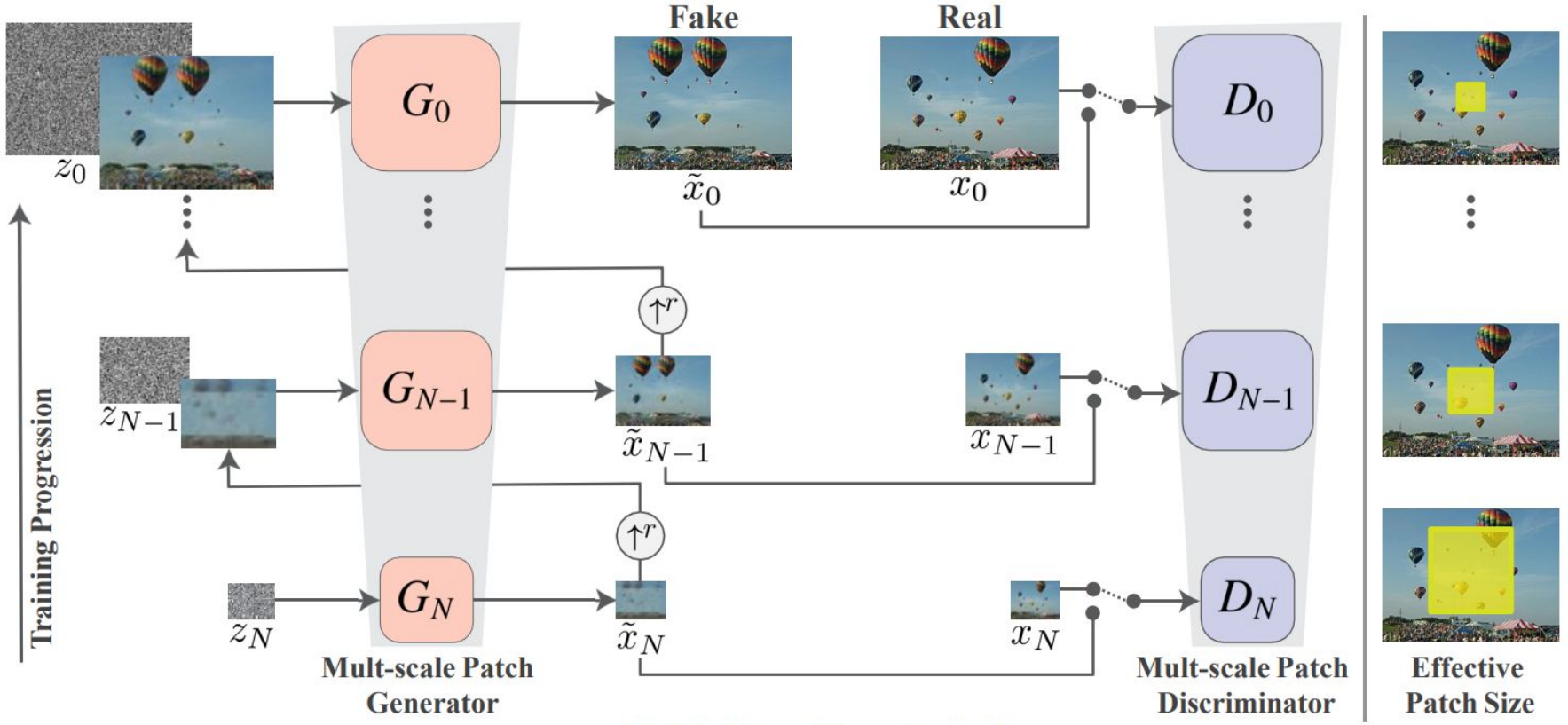
SinGAN

Learning a Generative model from a single image

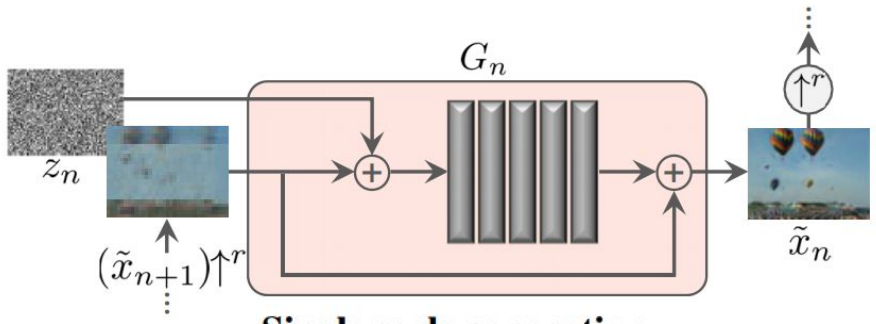


*source from SinGAN, Shaham et al. ICCV2019

SinGAN



SinGAN's multi-scale pipeline.

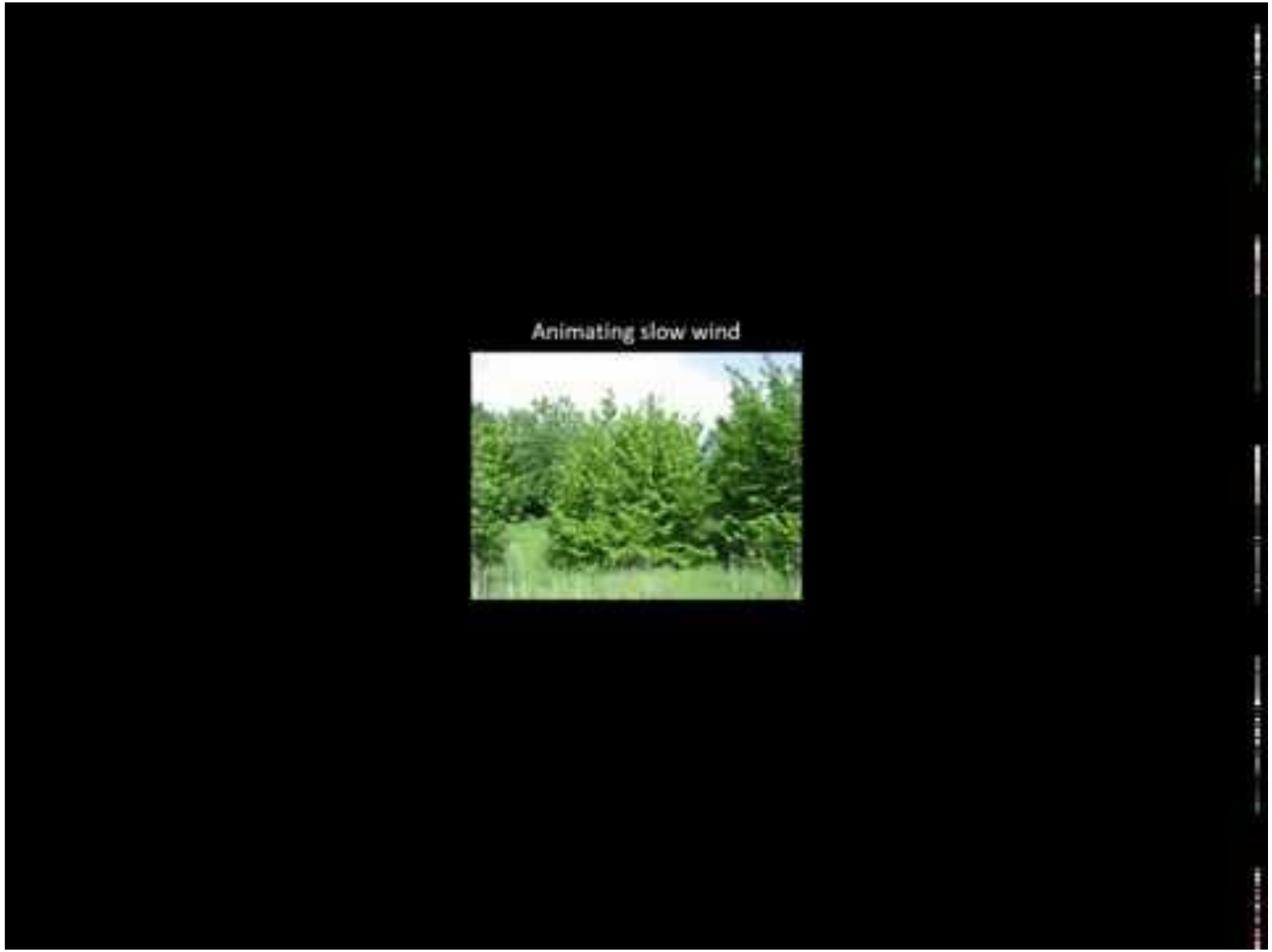


Single scale generation.

*source from [SinGAN, Shaham et al. ICCV2019]

SinGAN

Learning a Generative model from a single image



*source from SinGAN, Shaham et al. ICCV2019

More reading

The GANs zoo

In deep comparison between discriminative and generative models by Ng & Jordan (NIPS 2002)

Generative Adversarial Nets by Ian Goodfellow at NIPS 2014 (original paper)

Tutorial about GANs by Ian Goodfellow at NIPS 2016 (handouts) (slides)

CVPR 2018 Tutorial on GANs by Ian Goodfellow (slides) (video)

Towards principled methods for training generative adversarial networks by Martin Arjovsky et al., (2017)

Practical tips and tricks to make GANs work

Magenta: Music and Art Generation (Google Brain)

Generative models slide from cs231n, Stanford

Wasserstein GAN paper

From GAN to WGAN

Original, Wasserstein, and Wasserstein-Gradient-Penalty DCGAN

Least Squares GAN

Self Attention GAN

BEGAN: Boundary Equilibrium Generative Adversarial Networks

f-VAEs: Improve VAEs with Conditional Flows

Variational Autoencoder- Generative Adversarial Networks (VAE-GAN) (github)

Generative adversarial networks: introduction and outlook (GANs survey 2017)

NVIDA - GauGAN - CVPR 2019

SinGAN: Learning a Generative Model from a Single Natural Image, Shaham et al. ICCV 2019 (best paper)

… (much mores online)

Takeaway

- Generative models.

- The fundamental of GANs.

- *“Generative Adversarial Networks is the most interesting idea in machine learning in last ten years.”*

  Yann Lecun (Facebook AI Director)

Thank you