10-423/10-623 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Generative Adversarial Networks (GANs) + Probabilistic Graphical Models

Aran Nayebi & Matt Gormley
Lecture 6
Sep. 15, 2025

Reminders

- Homework 1: Generative Models of Text
  - Out: Mon, Sep 8
  - Due: Mon, Sep 22 at 11:59pm
- Quiz 2 now on Wed, Sep 24!

TASK: IMAGE GENERATION

Image Generation

- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- Text-to-image (TTI) generation

Figure from Razavi et al. (2019)

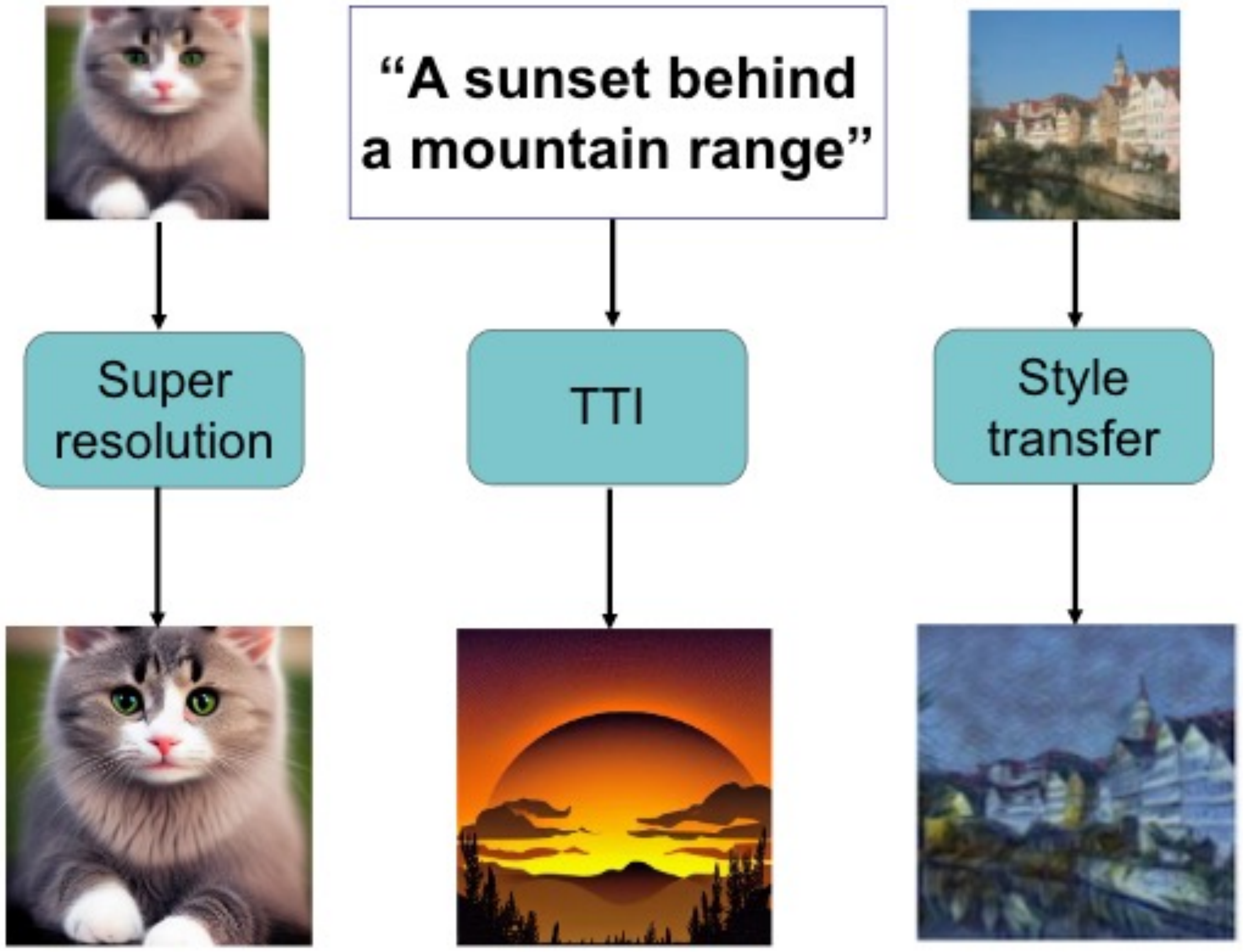


Figure from Bie et al. (2023)

Class Conditional Generation

- **Task**: Given a class label indicating the image type, sample a new image from the model with that type
- Image classification is the problem of taking in an image and predicting its label p(y|x)
- Class conditional generation is doing this in reverse p(x|y)

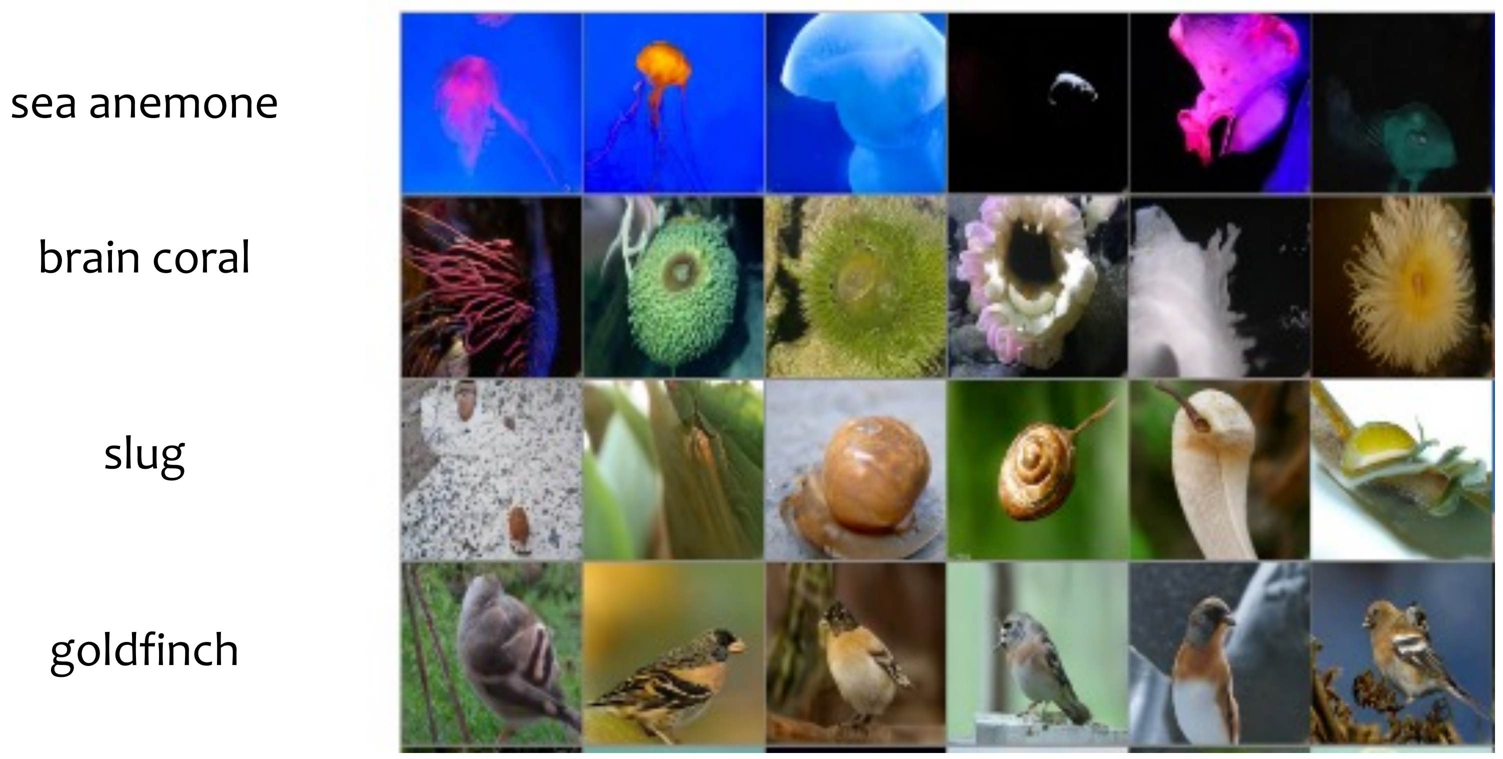
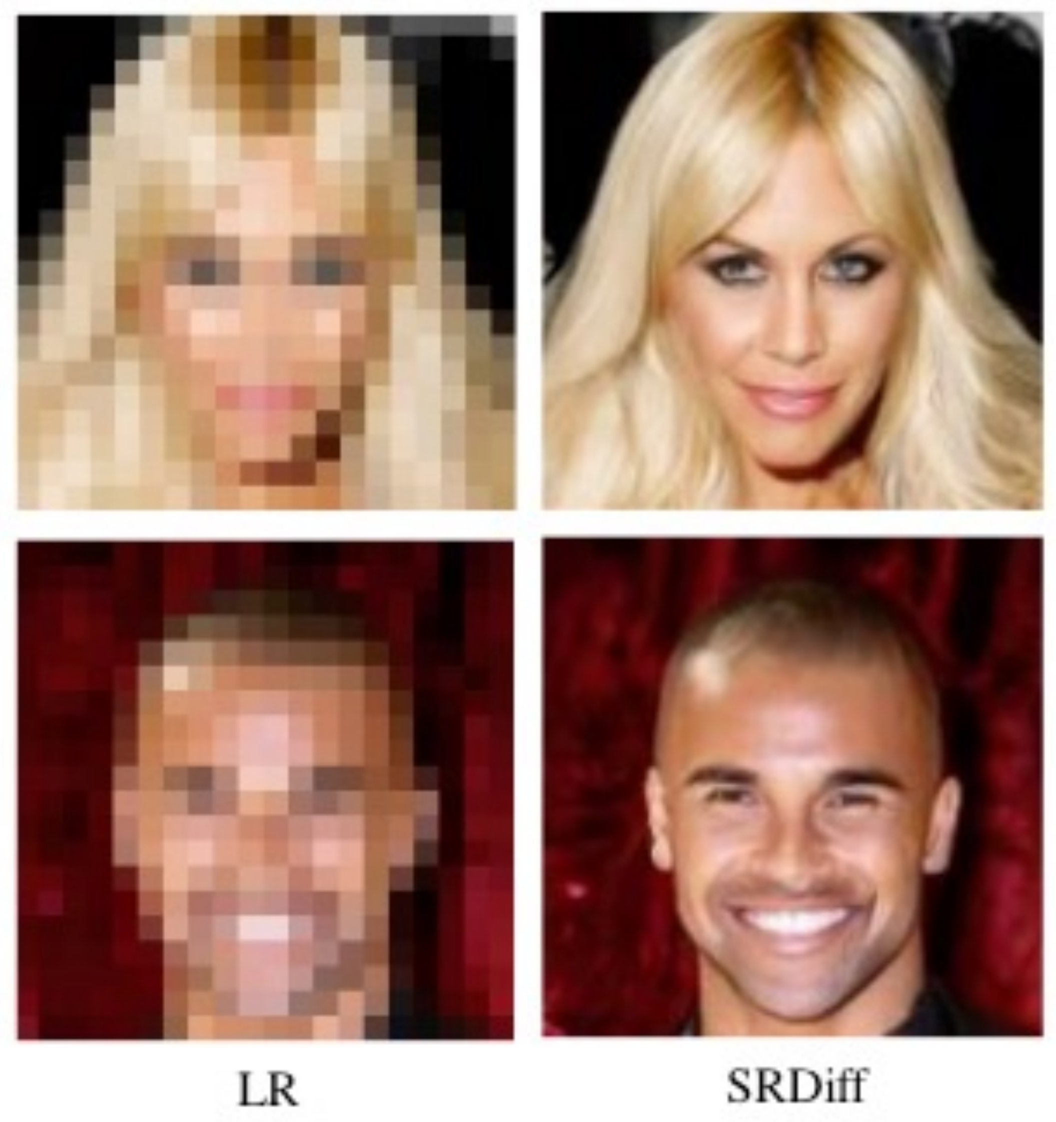


Figure from Razavi et al. (2019)

Super Resolution



- Given a low resolution image, generate a high resolution reconstruction of the image
- Compelling on low resolution inputs (see example to the left) but also effective on high resolution inputs

Figure from Li et al. (2021)

Image Editing

A variety of tasks involve automatic editing of an image:

- **Inpainting** fills in the (pre-specified) missing pixels
- **Colorization** restores color to a greyscale image
- **Uncropping** creates a photo-realistic reconstruction of a missing side of an image

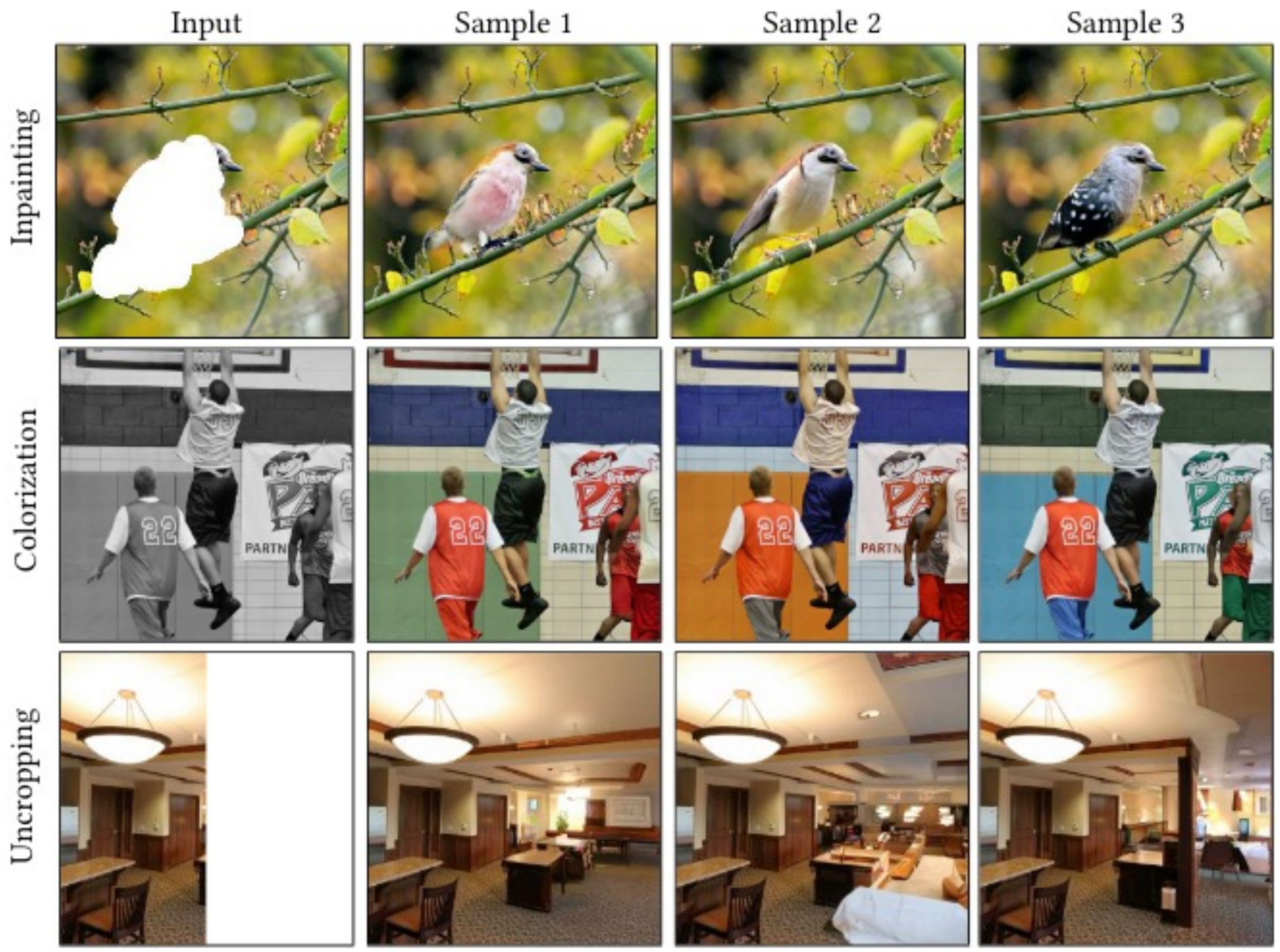
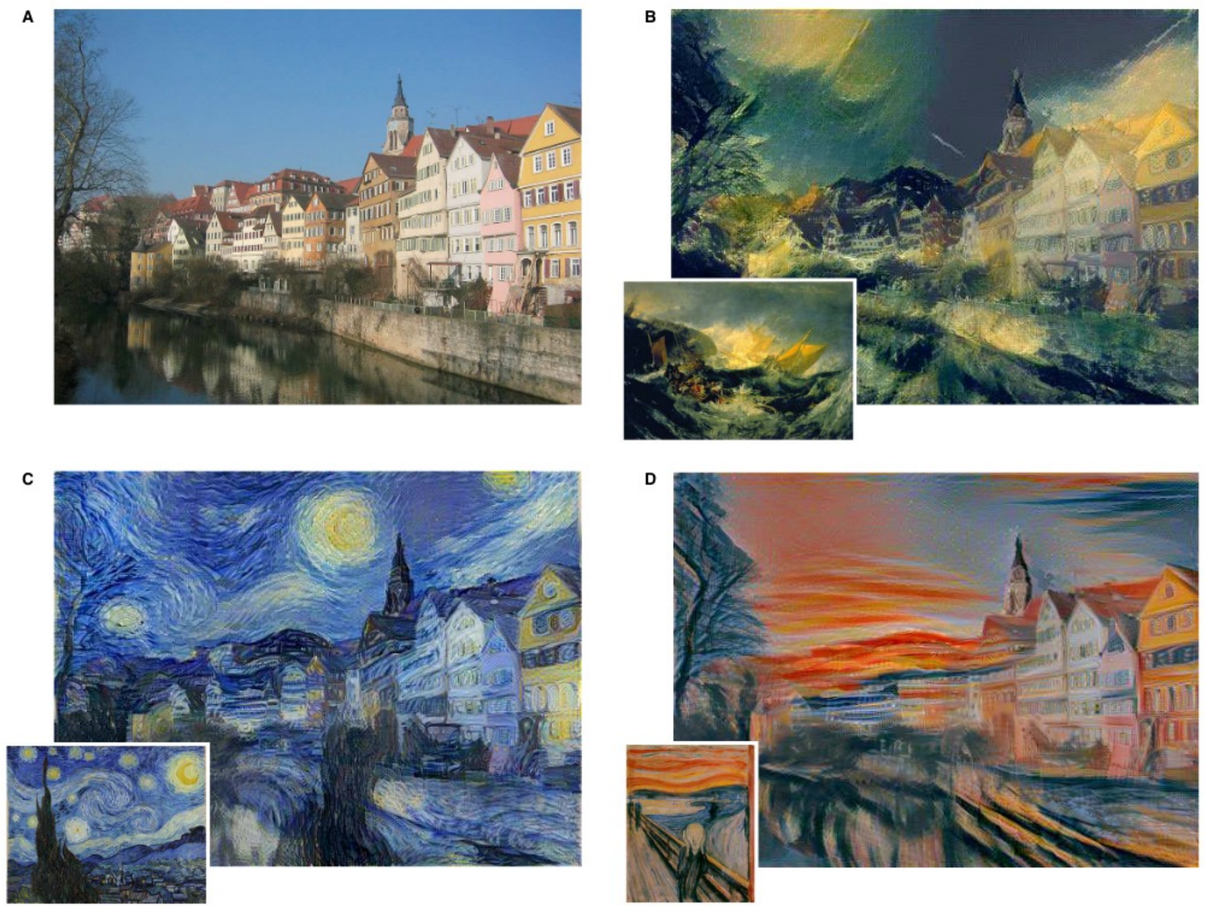


Figure from Saharia et al. (2022)

Style Transfer

- The goal of style transfer is to blend two images
- Yet, the blend should retain the semantic content of the source image presented in the style of another image



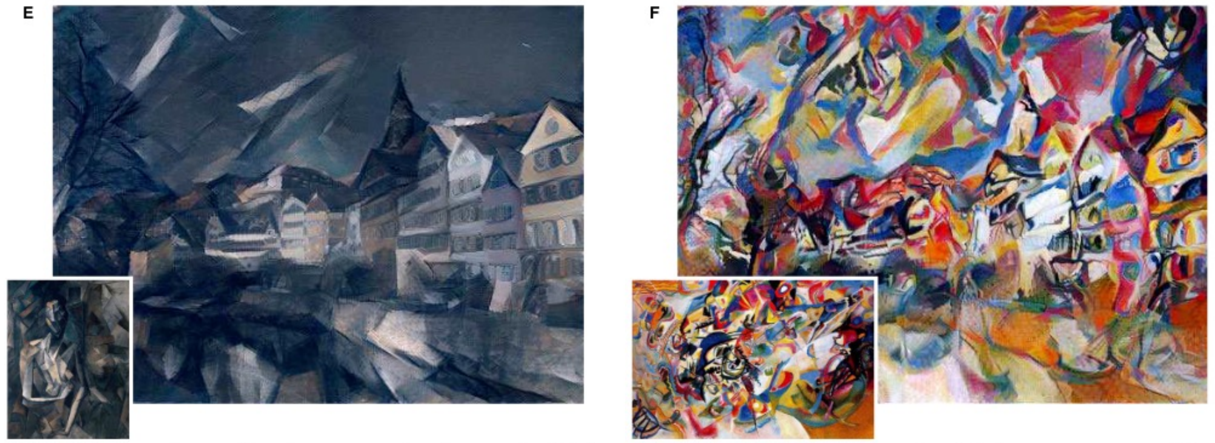


Figure 3. Images that combine the content of a photograph with the style of several well-known artworks. The images were created by finding an image that simultaneously matches the content representation of the photograph and the style representation of the artwork. The original photograph depicting the Neckarfront in Tübingen, Germany, is shown in **A** (Photo: Andreas Praefcke). The painting that provided the style for the respective generated image is shown in the bottom left corner of each panel. **B** *The Shipwreck of the Minotaur* by J.M.W. Turner, 1805. **C** *The Starry Night* by Vincent van Gogh, 1889. **D** *Der Schrei* by Edvard Munch, 1893. **E** *Femme nue assise* by Pablo Picasso, 1910. **F** *Composition VII* by Wassily Kandinsky, 1913.

Figure from Gatys et al. (2016)

Text-to-Image Generation

- Given a text description, sample an image that depicts the prompt
- From Google Gemini 2.5 Flash Image (Nano Banana)

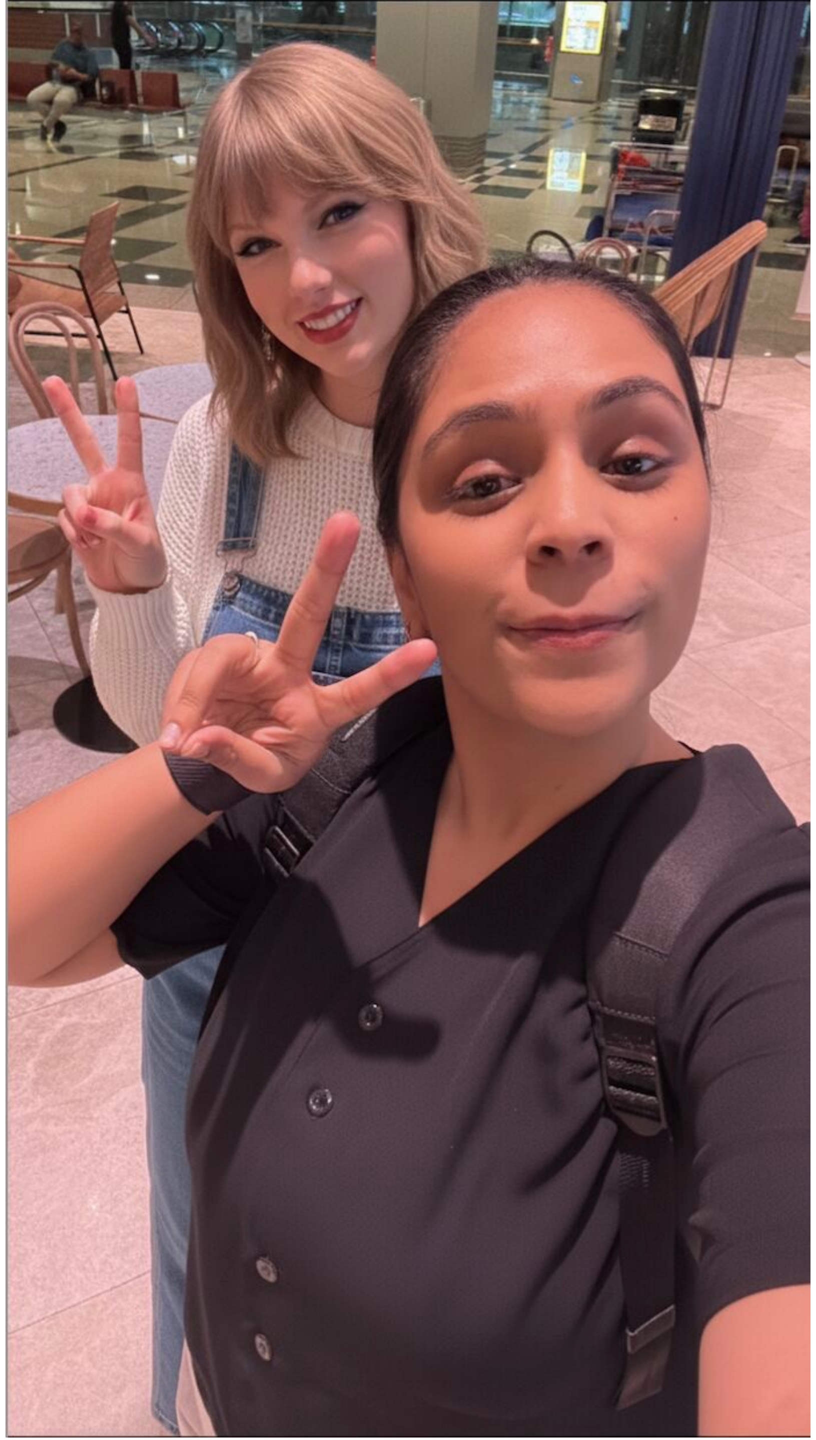


Figure from: https://www.linkedin.com/posts/alka-gupta-ag_so-i-bumped-into-taylor-swift-at-changi-airport-activity-7369534745061322753-296P

Text-to-Image Generation

- Given a text description, sample an image that depicts the prompt
- The following images are samples from SDXL with refinement

Prompt: close up headshot, futuristic **old man**, wild hair sly smile in front of gigantic UFO, dslr, sharp focus, dynamic composition, **rule of thirds**

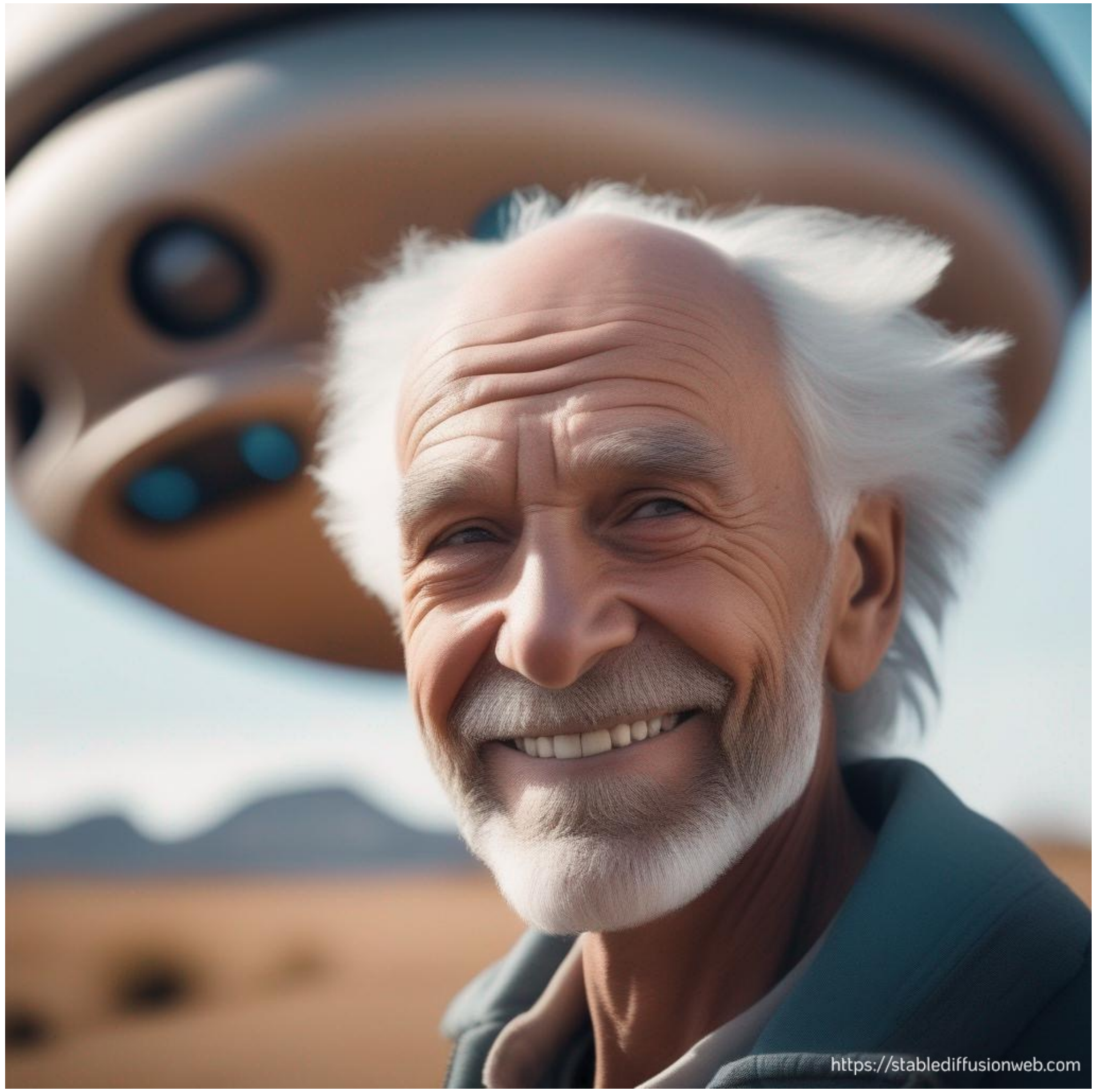


Figure from https://stablediffusionweb.com/

MODEL: GENERATIVE ADVERSARIAL NETWORK (GAN)

Stable Diffusion (Spring 2025) still can't explain GANs

Prompt: slide explaining Generative Adversarial Networks (GANs) for a Machine Learning course, carefully designed, easy to follow

Negative Prompt: boring, unclear, nontechnical

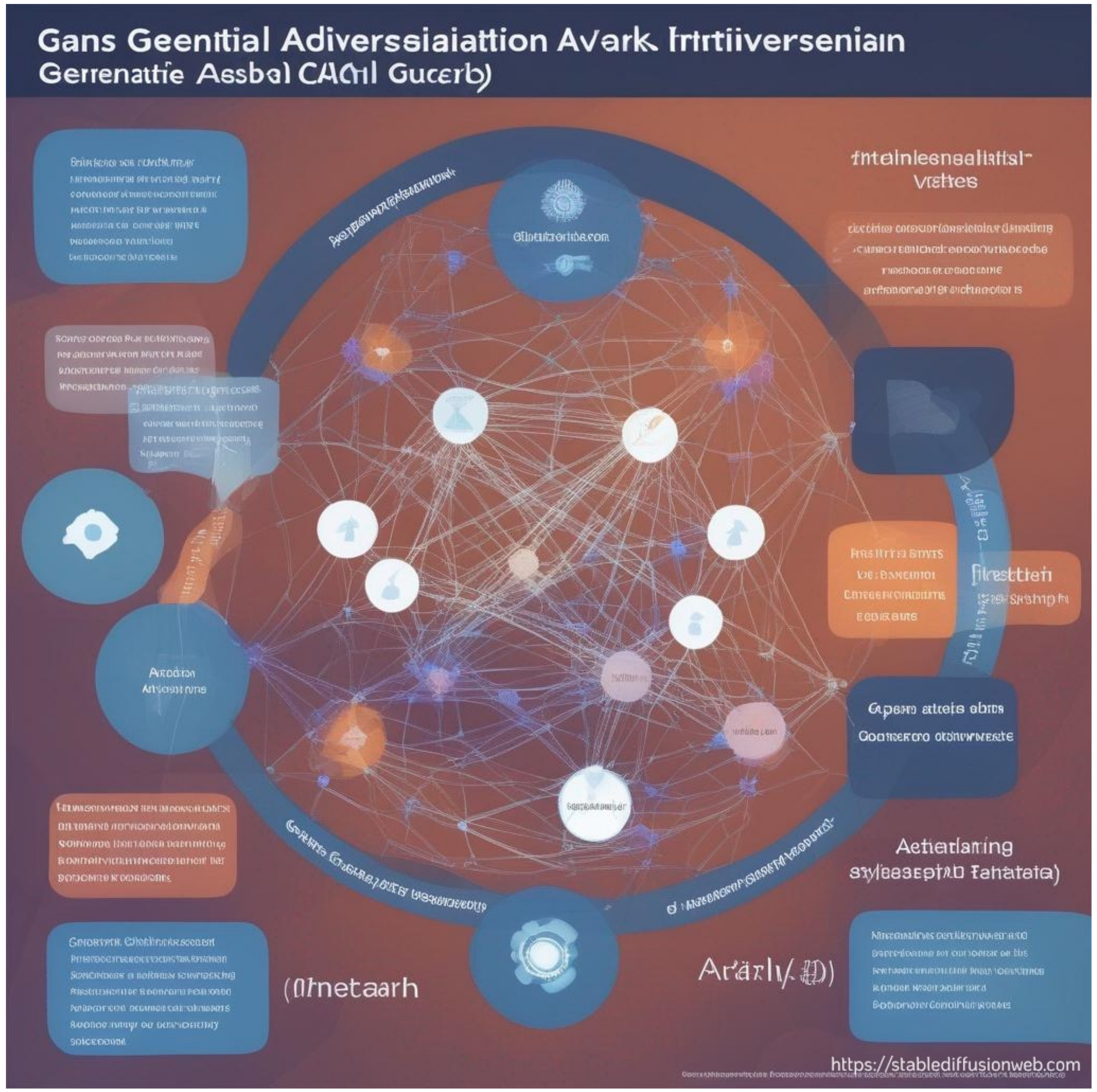
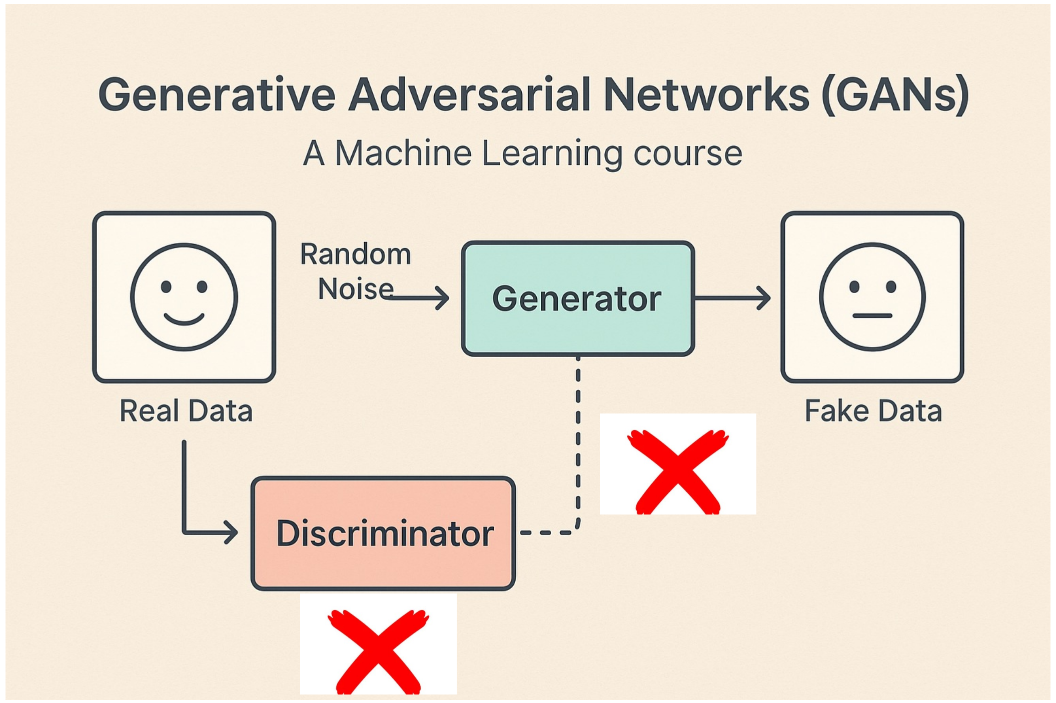

Gans Geentitial Adiversaiattion Avark Irrtiiversenian
Gerrenattie Assbal CACtl Gucerb)
(Ilnetaarh
Arärly.(D)
https://stablediffusionweb.com

Figure from https://stablediffusionweb.com/

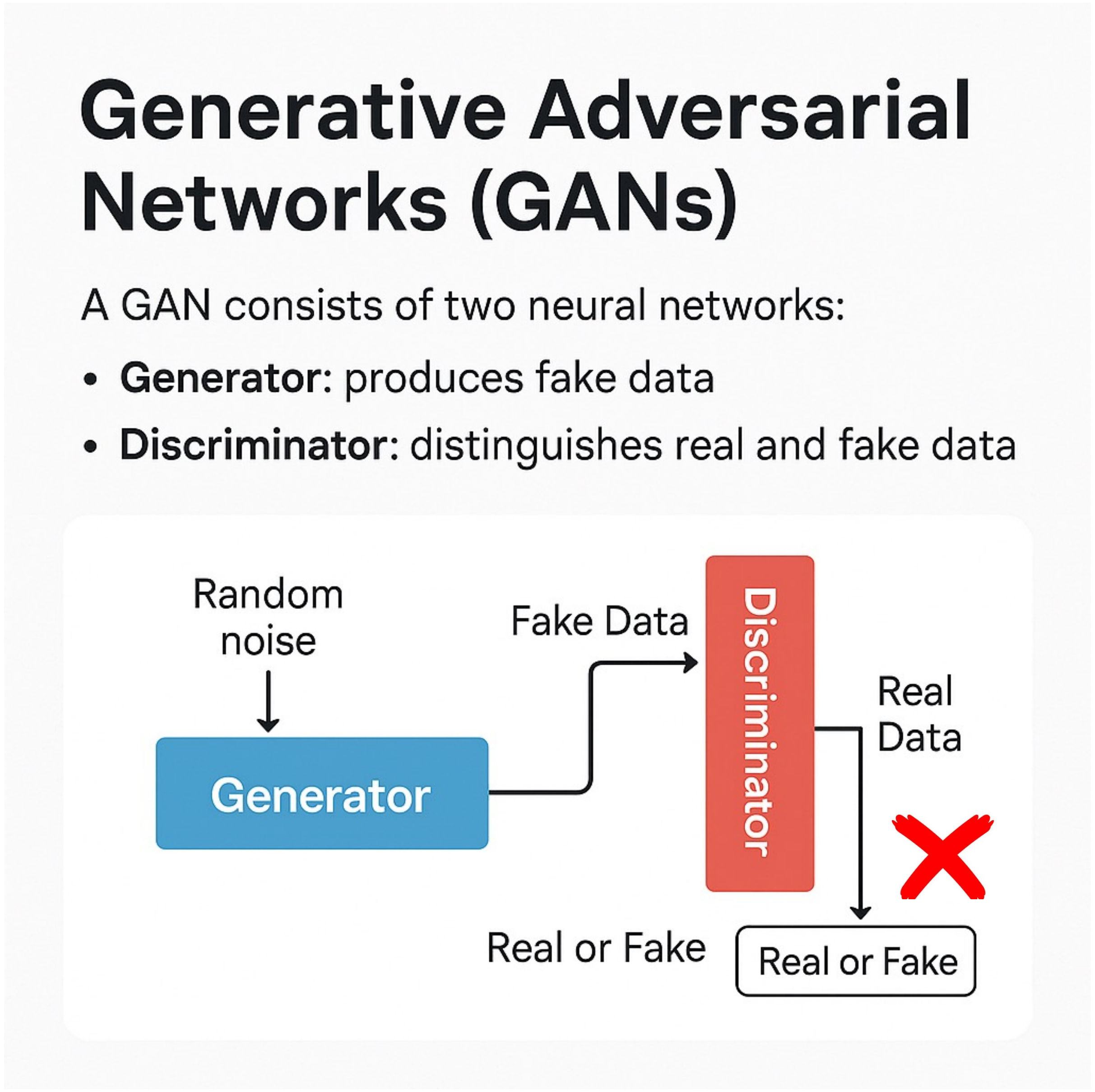
GPT5 (Fall 2025) does a better (but incorrect!) job at explaining GANs



Generative Adversarial Networks (GANs)
A Machine Learning course
Random Noise
Generator
Real Data
Fake Data
Discriminator

Prompt: Generate a slide (as an image) explaining Generative Adversarial Networks (GANs) for a Machine Learning course, carefully designed, easy to follow

Negative Prompt: boring, unclear, nontechnical



Generative Adversarial Networks (GANs)

A GAN consists of two neural networks:

- **Generator**: produces fake data
- **Discriminator**: distinguishes real and fake data

Random noise
Generator
Fake Data
Discriminator
Real Data
Real or Fake
Real or Fake

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

Generator Model

1) the Generator

takes a vector of random noise as input, and generates an image

Example Generator: DCGAN

- An inverted CNN with four **fractionally-strided** convolution layers (not deconvolution)
- These fractional strides grow the size of the image from layer to layer
- The final layer has three channels for red/green/blue

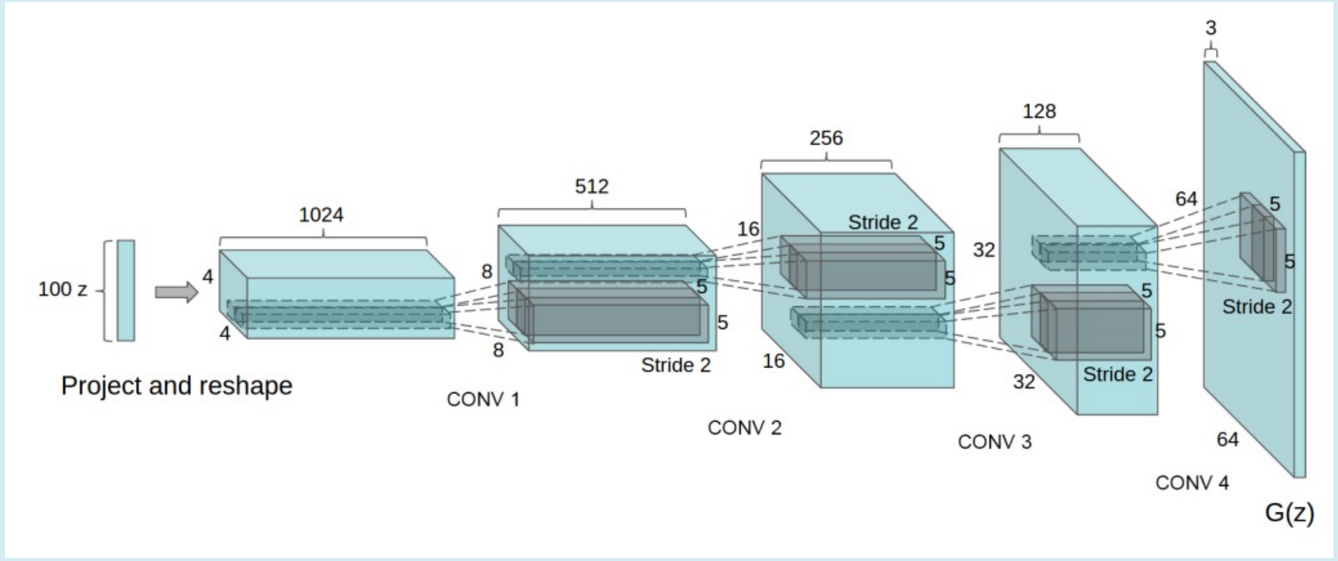


Figure from Radford et al. (2016)

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

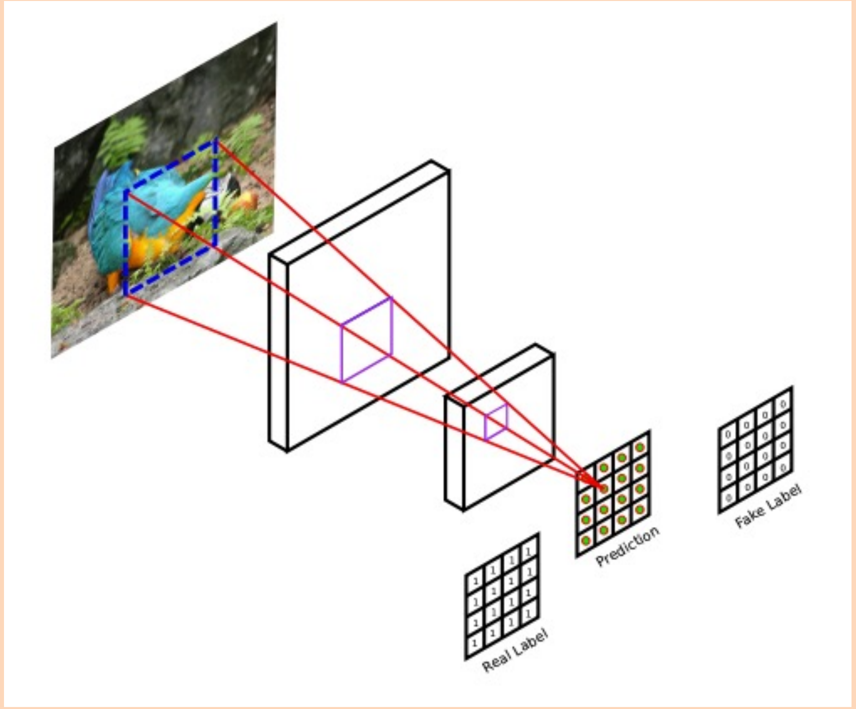
2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

Discriminator Model

Example Discriminator: PatchGAN

- Convolutional neural network
- Looks at each patch of the image and tries to predict whether it is real or fake
- Helps avoid producing blurry images



2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

Figure from Demir & Unal (2018)

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

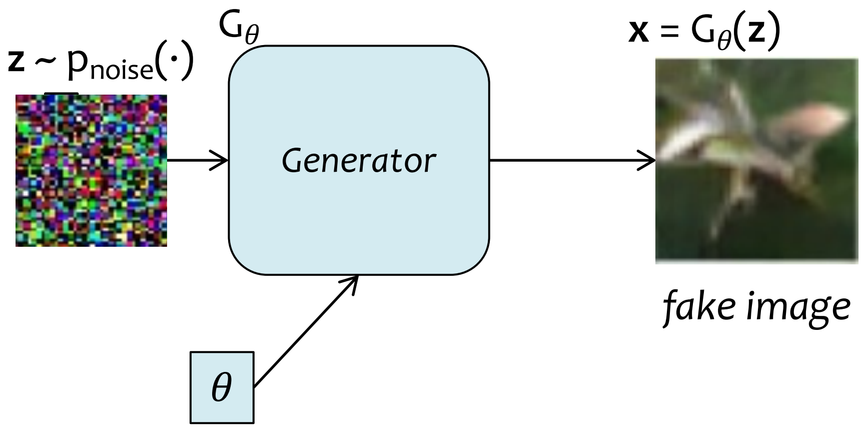
2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

In training, the GAN plays a two player minimax game:

1. the Generator tries to create realistic images to fool the Discriminator into thinking they are real
2. the Discriminator tries to identify the real images from the fake

Generative Adversarial Networks (GANs)

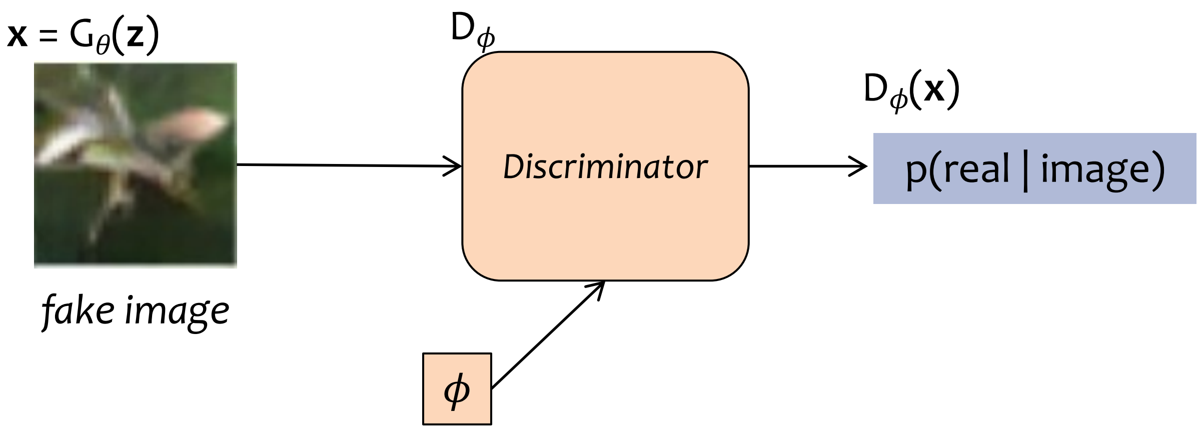


$\mathbf{z} \sim p_{noise}(\cdot)$
G_θ
Generator
θ
$\mathbf{x} = G_\theta(\mathbf{z})$
fake image

Real/fake images from Huang et al. (2017)

Gaussian noise from https://physbam.stanford.edu/cs448x/old/Noise_Review.html

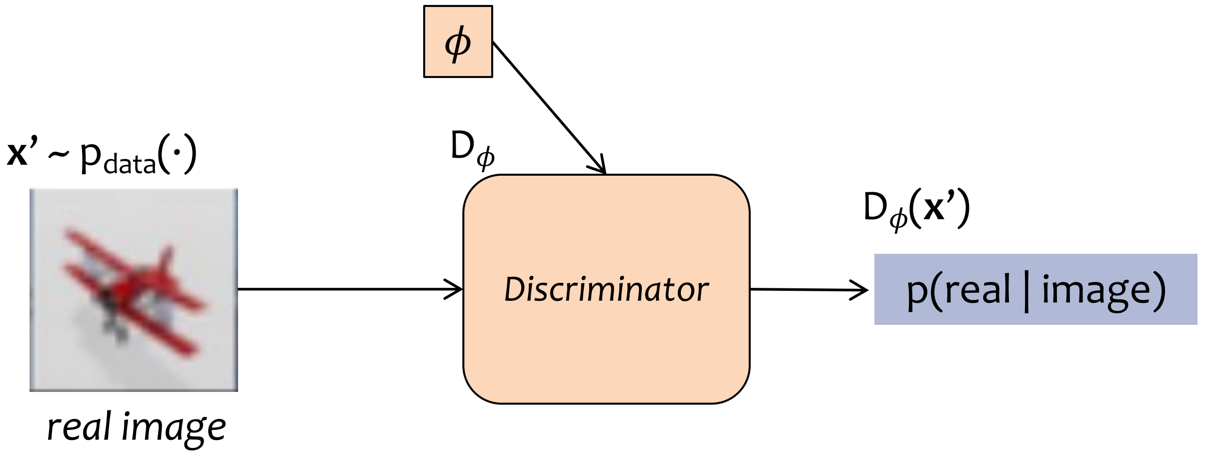
Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

Gaussian noise from https://physbam.stanford.edu/cs448x/old/Noise_Review.html

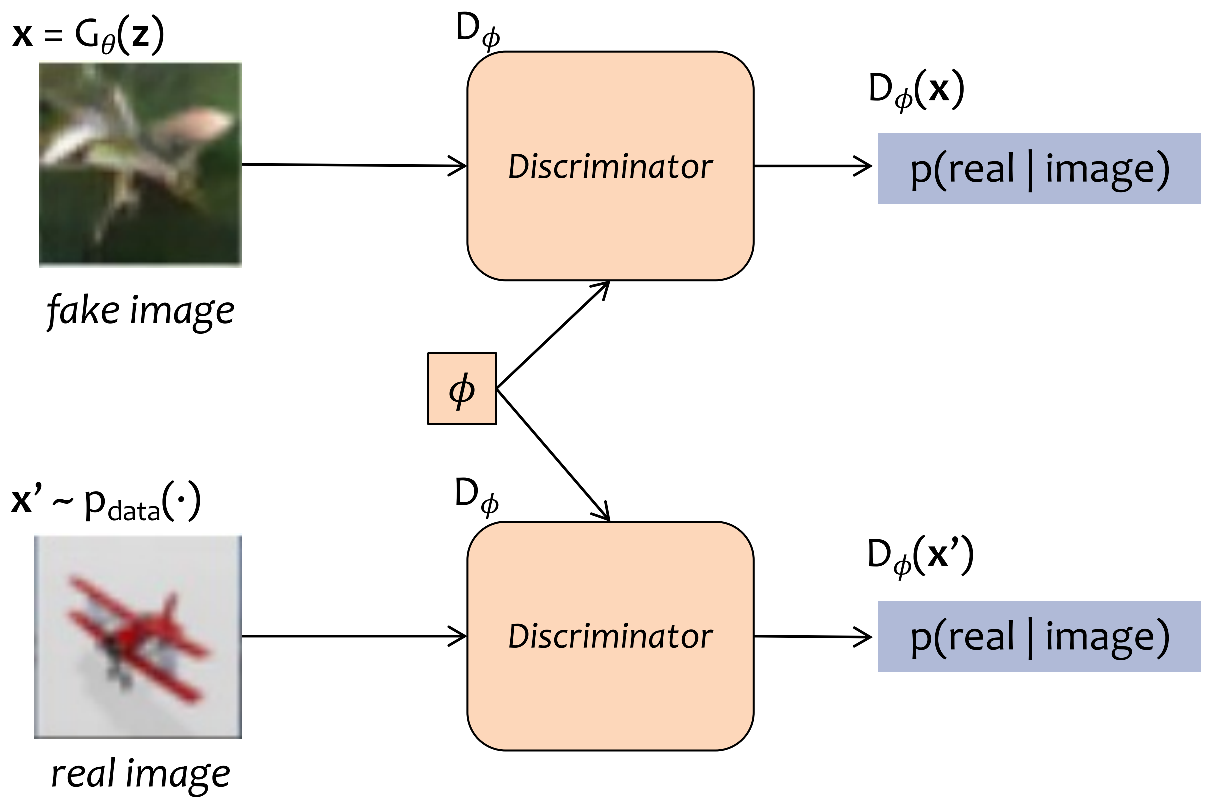
Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

Gaussian noise from https://physbam.stanford.edu/cs448x/old/Noise_Review.html

Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

Gaussian noise from https://physbam.stanford.edu/cs448x/old/Noise_Review.html

LEARNING FOR GANS

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

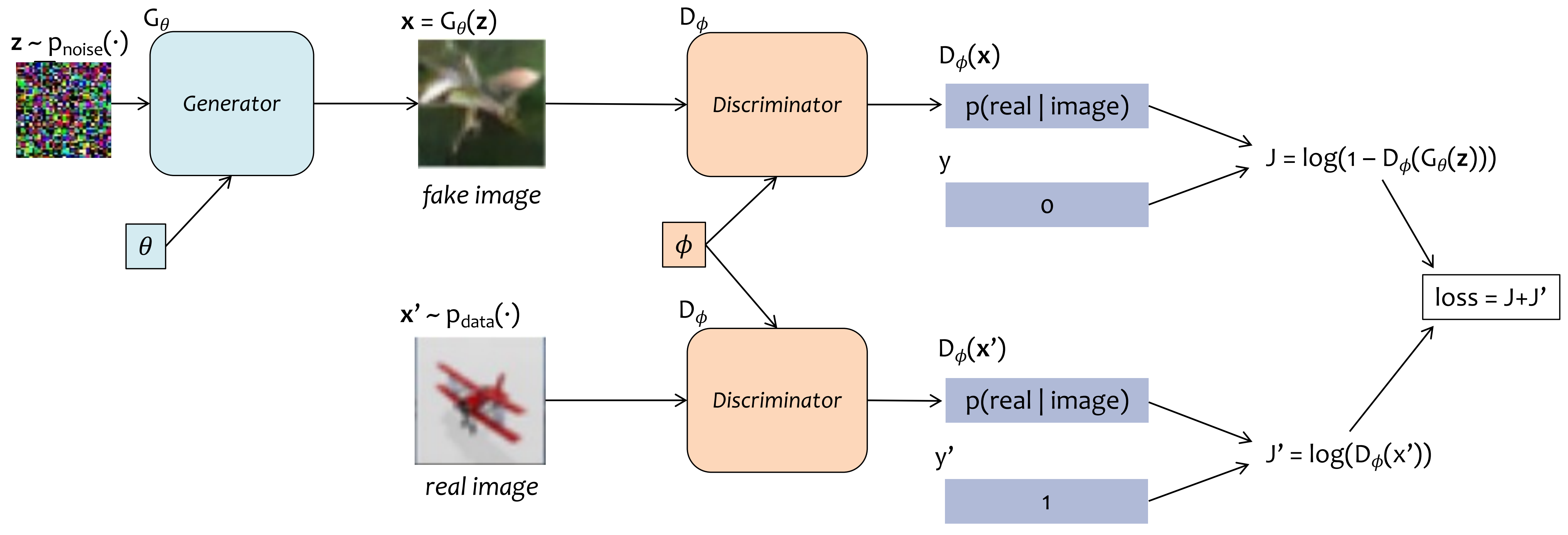
2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

In training, the GAN plays a two player minimax game:

1. the Generator tries to create realistic images to fool the Discriminator into thinking they are real
2. the Discriminator tries to identify the real images from the fake

Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

Gaussian noise from https://physbam.stanford.edu/cs448x/old/Noise_Review.html

Generative Adversarial Networks (GANs)

$$\max_{\phi} J(\theta, \phi)$$

$$\max_{\phi} \log\left(D_\phi(\mathbf{x}^{(i)})\right) + \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

The discriminator is trying to maximize the likelihood of a binary classifier with labels {real = 1, fake = 0}, on the fixed output of the generator

$$\min_{\theta} \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

The generator is trying to minimize the likelihood of its generated (fake) image being classified as fake, according to a fixed discriminator

$$\min_{\theta} J(\theta, \phi)$$

In training, the GAN plays a two player minimax game:

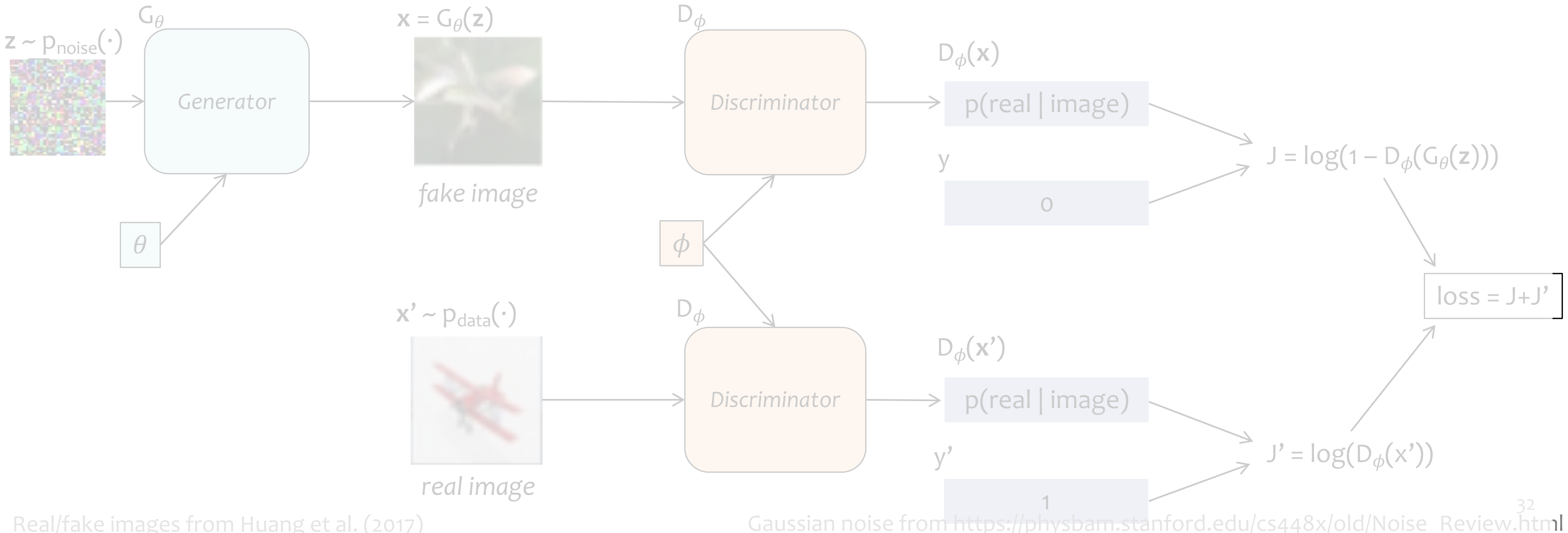
1. the Generator tries to create realistic images to fool the Discriminator into thinking they are real
2. the Discriminator tries to identify the real images from the fake

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- Keep D_ϕ fixed and backprop through G_θ



Real/fake images from Huang et al. (2017)

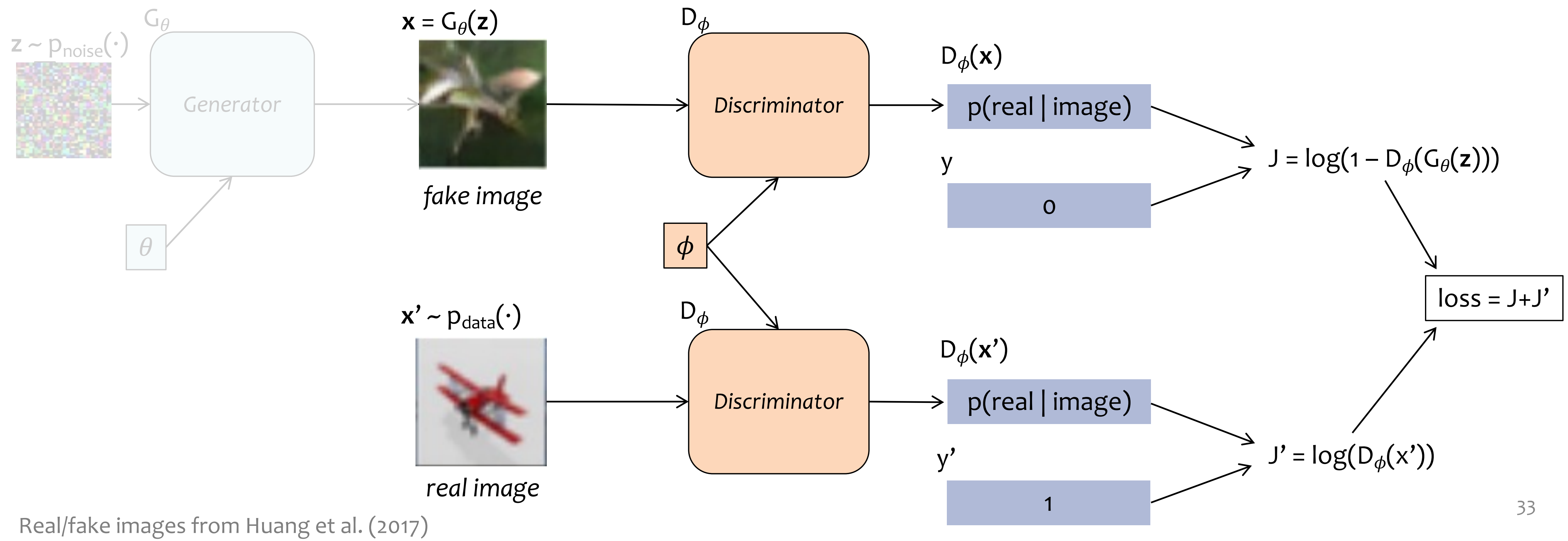
Gaussian noise from https://physbam.stanford.edu/cs448x/old/Noise_Review.html

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- **Keep G_θ fixed and backprop through D_ϕ**
- Keep D_ϕ fixed and backprop through G_θ



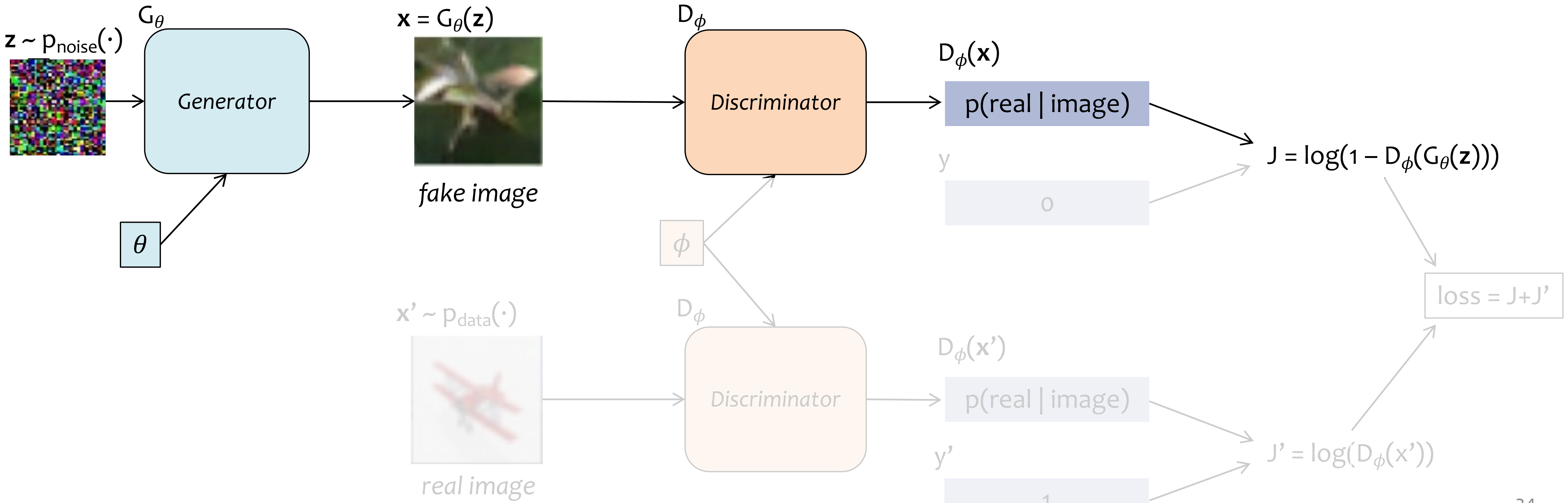
Real/fake images from Huang et al. (2017)

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- **Keep D_ϕ fixed and backprop through G_θ**



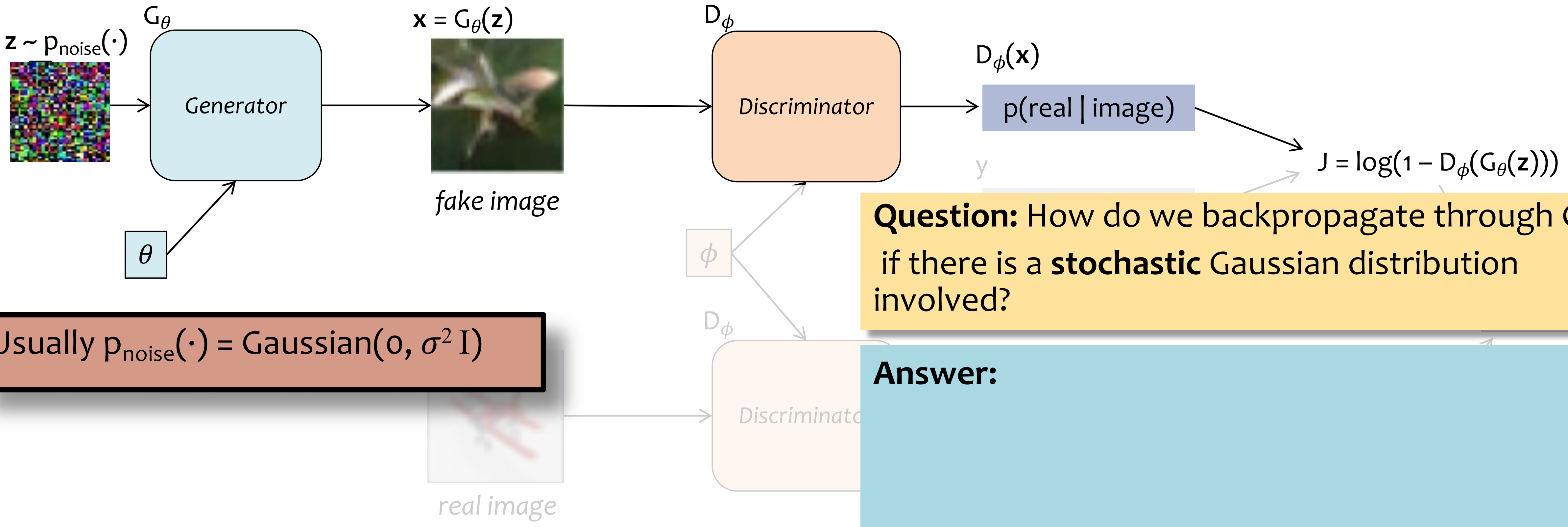
Real/fake images from Huang et al. (2017)

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- **Keep D_ϕ fixed and backprop through G_θ**



Usually $p_{noise}(\cdot)$ = Gaussian(0, σ^2 I)

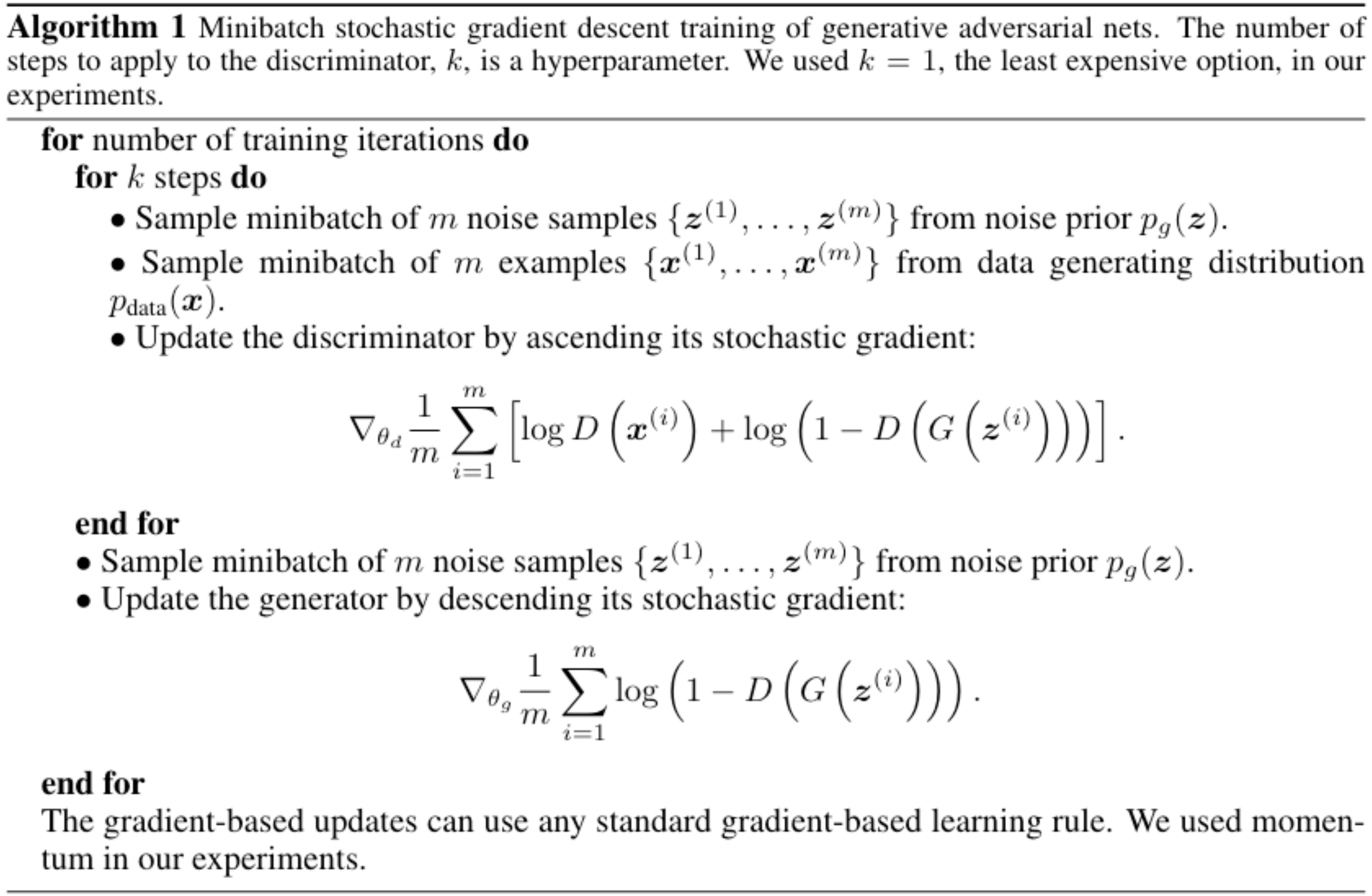
Question: How do we backpropagate through G_θ if there is a **stochastic** Gaussian distribution involved?

Answer:

Real/fake images from Huang et al. (2017)

Learning a GAN

- Training data consists of a collection of m unlabeled images $x^{(1)}, \ldots, x^{(m)}$
- Optimization is similar to block coordinate descent
- But instead of exactly solving the min/max problem, we take a step of mini-batch SGD



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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
        • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
        • Update the discriminator by ascending its stochastic gradient:
            $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$
    end for
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Update the generator by descending its stochastic gradient:
        $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$
end for
```

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

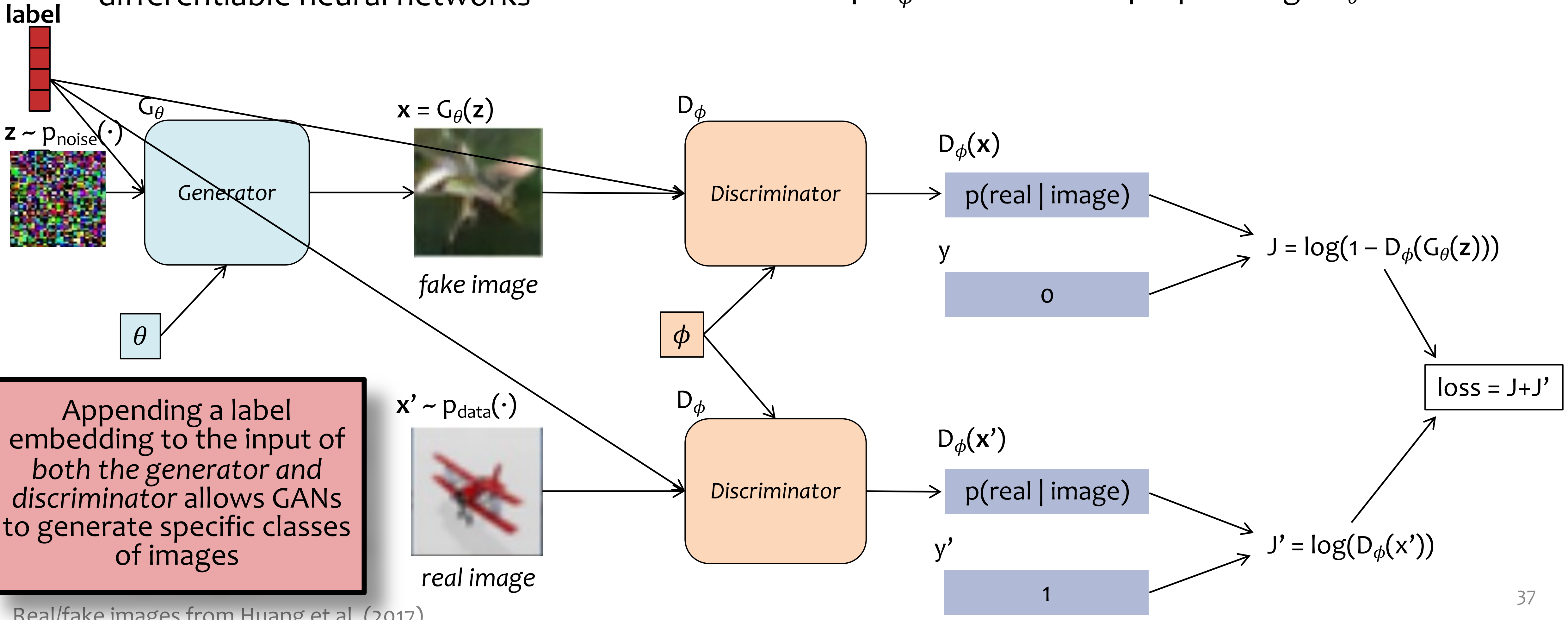
Figure from https://arxiv.org/pdf/1406.2661.pdf

Class-conditional GANs

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- Keep D_ϕ fixed and backprop through G_θ

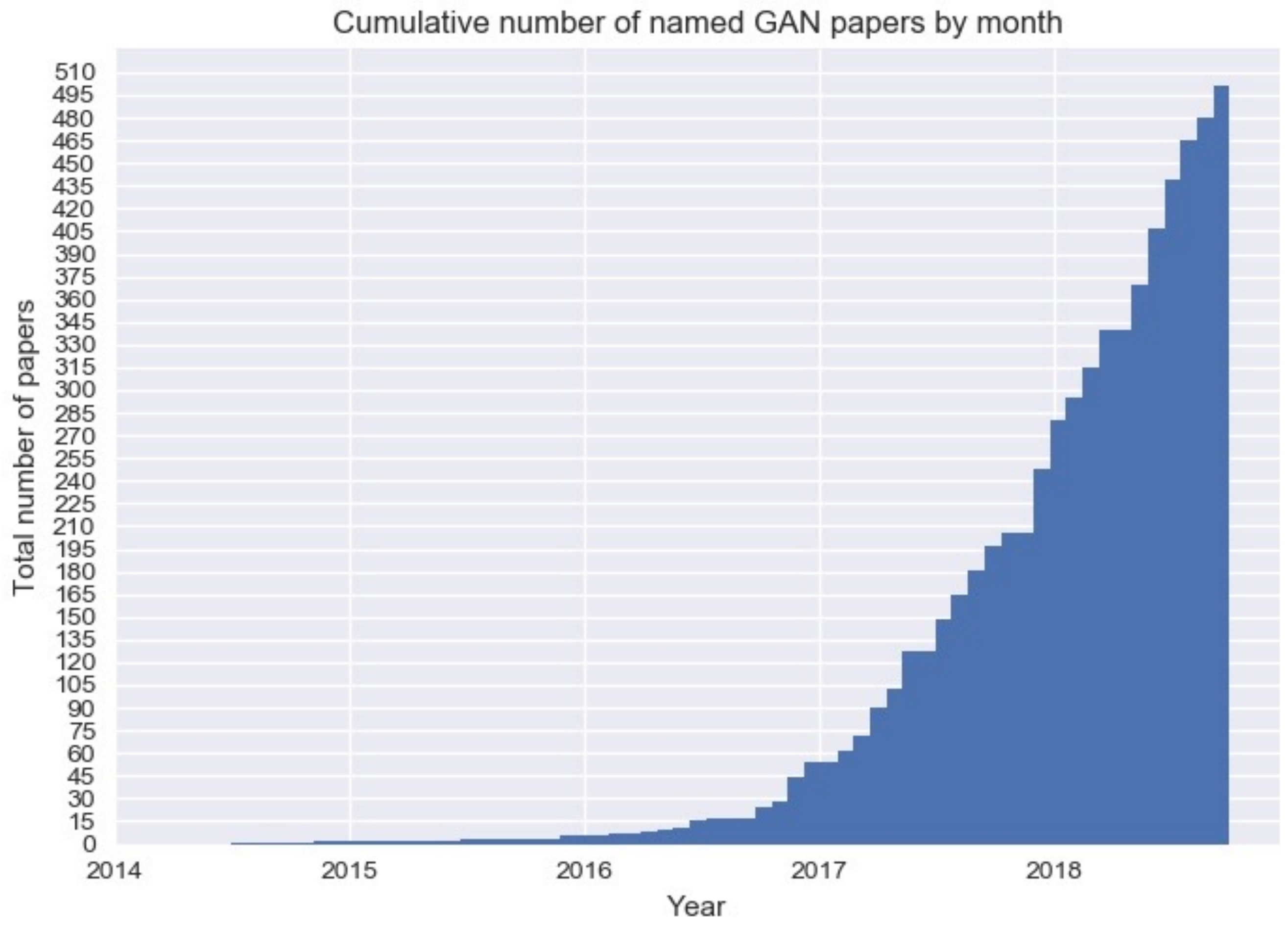


Appending a label embedding to the input of *both the generator and discriminator* allows GANs to generate specific classes of images

Real/fake images from Huang et al. (2017)

SCALING UP THE MODEL SIZE

GANs Everywhere!

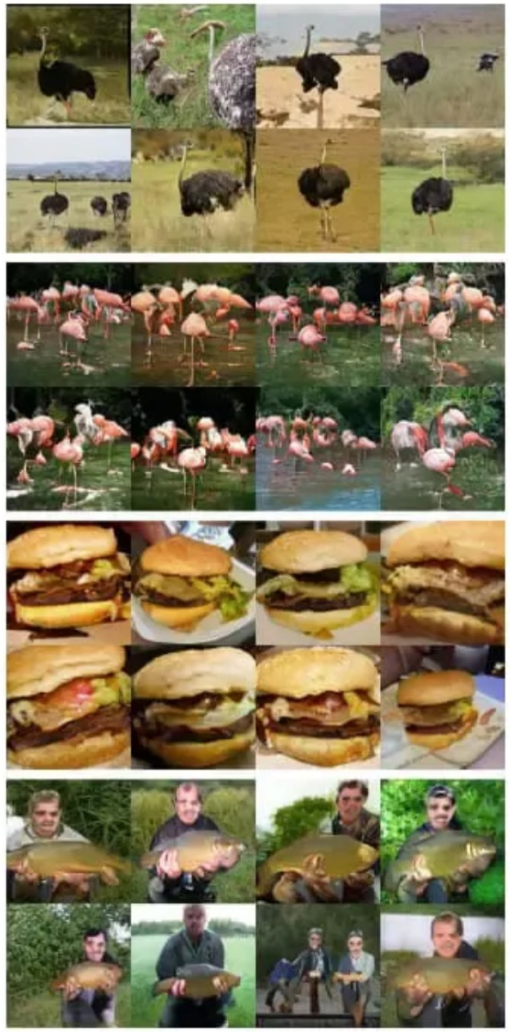


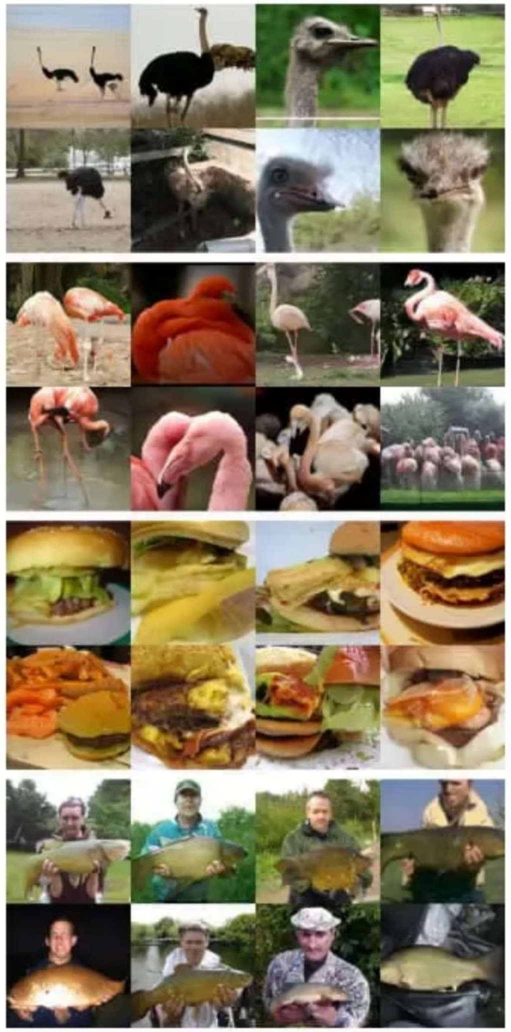
Source: https://github.com/hindupuravinash/the-gan-zoo/tree/master

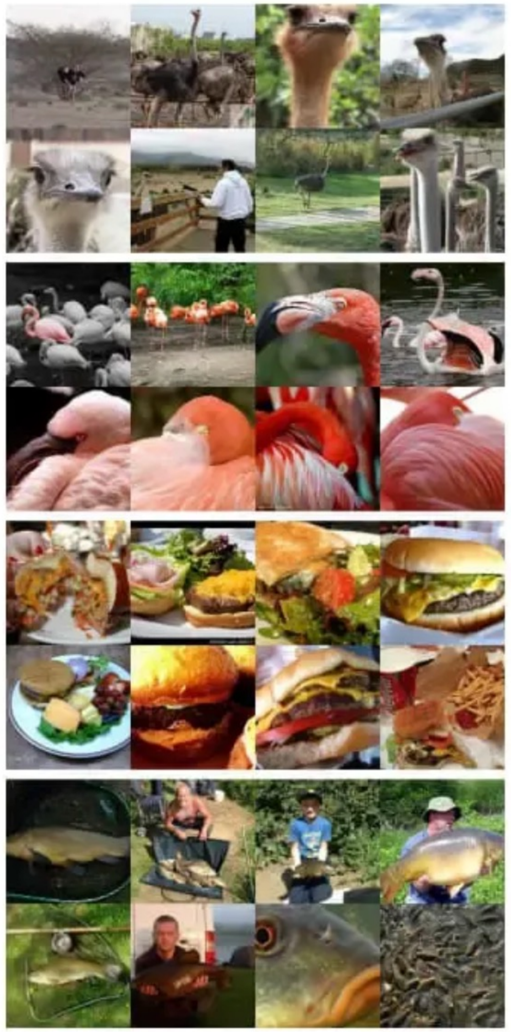
Recall: Computer Vision Timeline

1998	LeNet
2009	ImageNet
2012	AlexNet
2013	VAEs
2014	VGG R-CNN **GANs**
2015	Diffusion models ResNet
2017	Transformer
2020	DDPM
2021	ViT Dall-E CLIP
2022	Dall-E 2 Imagen Stable diffusion
2023	SDXL SDXL Turbo

GANs vs. Diffusion


GAN generated images


Training images


Diffusion generated images

Source: https://medium.com/thedeephub/what-is-the-gan-generative-adversarial-networks-2ed6965c13fb

Scaling Up the Model Size

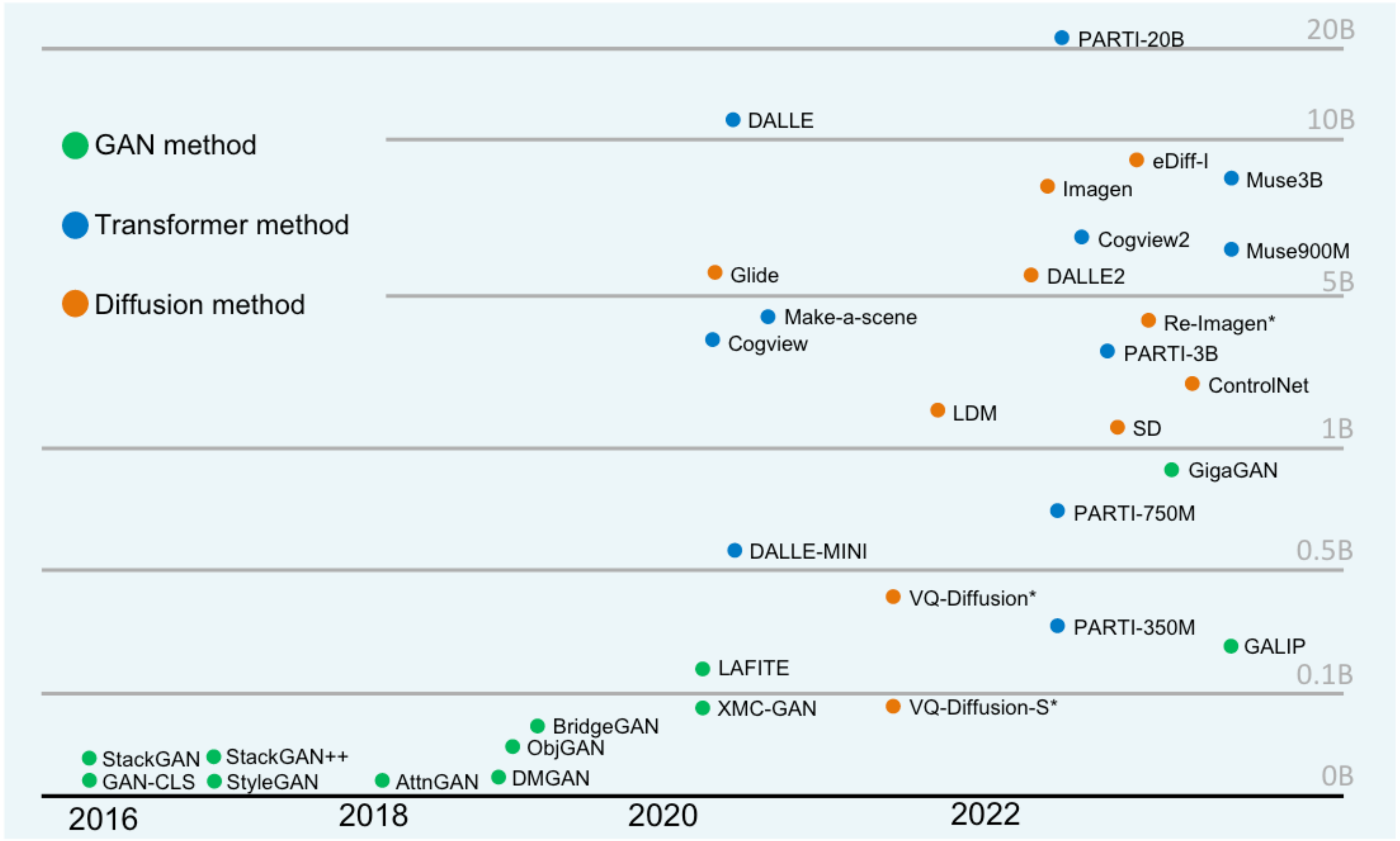


Fig. 5. Timeline of TTI model development, where green dots are GAN TTI models, blue dots are autoregressive Transformers and orange dots are Diffusion TTI models. Models are separated by their parameter, which are in general counted for all their components. Models with asterisk are calculated without the involvement of their text encoders.

Figure from Bie et al. (2023)

Scaling Up the Model Size

Prompt: A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

Parti with different model sizes

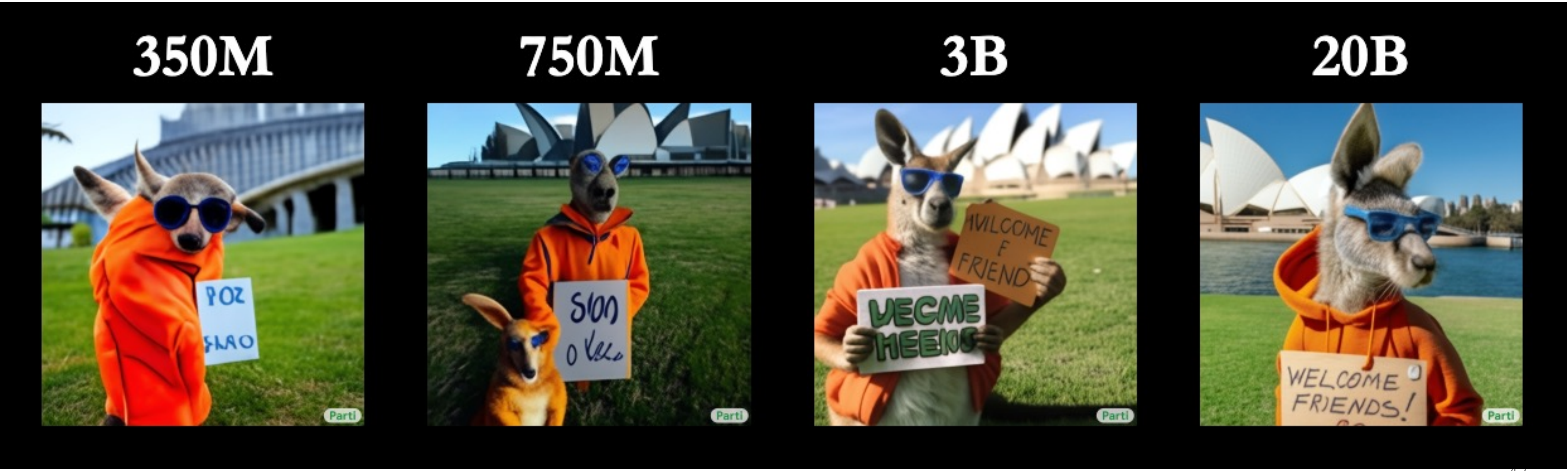


Figure from https://sites.research.google/parti/

Watermarking & Attribution

- **Watermarking**
  - A digital watermark allows one to identify when an image has been created by a model
  - Most methods for image generation (GANs, VAEs, stable diffusion) can be augmented with watermarking
- **Fake-image Detection**
  - Goal: identify fakes even without a watermark
- **Model Attribution**
  - Identify which generative model created an image (e.g. Dalle-2 vs. SDXL)
  - Very successful (natural watermarks)
- **Image Attribution**
  - Goal: identify the source images that led to the generation of a new image
  - Extremely challenging

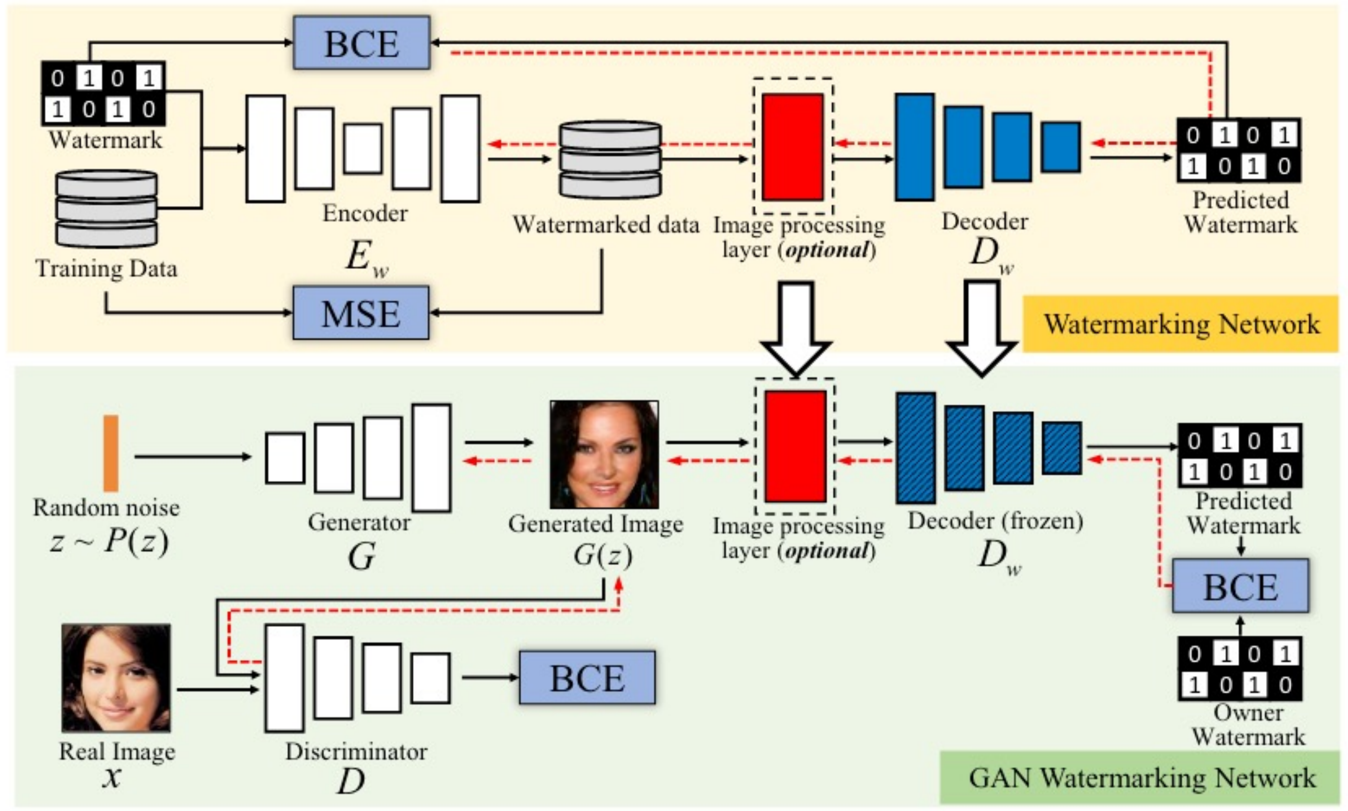


Figure from Fei et al. (2022)

Societal Impacts of Image Generation

Pros

- New tools for artists
- Faster creation of memes

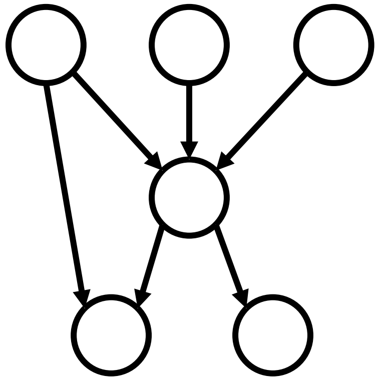
Cons

- Copyright infringement / loss of work for artists
- Societal decrease in creativity
- Potential to create dehumanizing content
- Fake news / false realities / increased difficulty of fact checking
- Not rooted in reality

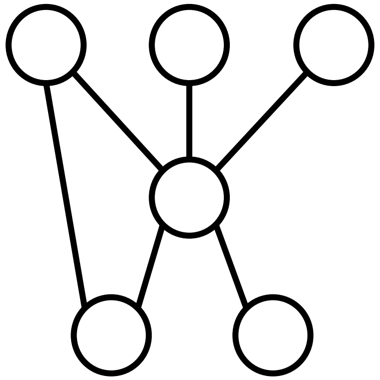
DIRECTED GRAPHICAL MODEL

Three Types of Graphical Models

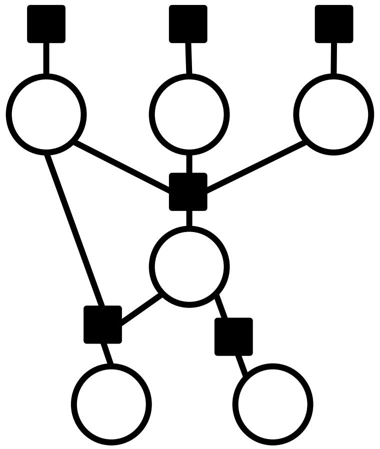
Directed Graphical Model



Undirected Graphical Model

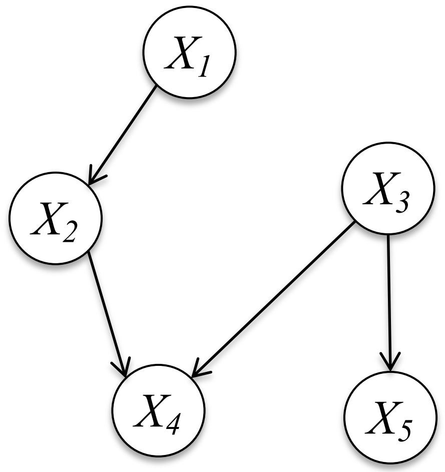


Factor Graph



Directed Graphical Model

Example



$$P(X_1, X_2, X_3, X_4, X_5) = P(X_5 \mid X_3)P(X_4 \mid X_2, X_3) P(X_3)P(X_2 \mid X_1)P(X_1)$$

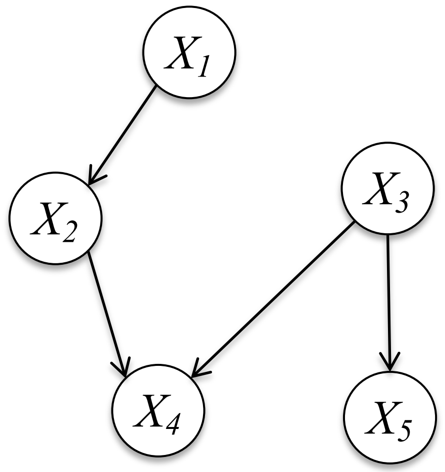
Definition

- A directed graphical model (aka. Bayesian network) is a **directed acyclic graph** that represents the conditional independencies of a set of variables $X_1, \ldots, X_T$
- Each **node** is variable X_t and each **edge** implies a directional influence between a pair of variables
- The DGM factorizes the joint distribution over the variables as a product of conditional probabilities:

$$P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$$

Directed Graphical Model

Example



$$p(x_1, x_2, x_3, x_4, x_5) = p(x_5 \mid x_3)p(x_4 \mid x_2, x_3) p(x_3)p(x_2 \mid x_1)p(x_1)$$

the graph (*qualitative specification*) could be:

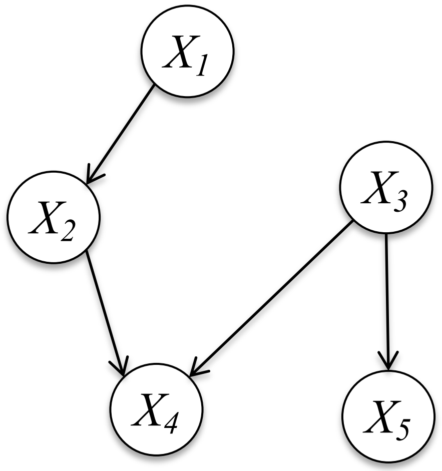
- specified using domain expertise about causal relationships
- learned from data
- chosen because of nice computational properties

the conditional probabilities (*quantitative specification*) is:

- depends on the types of variables involved
- typically learned from data

Directed Graphical Model

Example



$$p(x_1, x_2, x_3, x_4, x_5) =$$
$$p(x_5 \mid x_3)p(x_4 \mid x_2, x_3)$$
$$p(x_3)p(x_2 \mid x_1)p(x_1)$$

Quantitative Specification

x_1	$p(x_1)$
red	0.4
blue	0.6

x_2	x_1	$p(x_2 \mid x_1)$
red	red	0.7
blue	red	0.3
red	blue	0.5
blue	blue	0.5

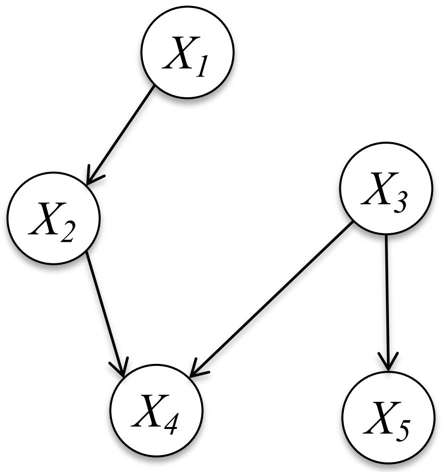
x_3	$p(x_3)$
red	0.6
blue	0.4

x_4	x_2	x_3	$p(x_4 \mid x_2, x_3)$
red	red	red	0.5
blue	red	red	0.5
red	red	blue	0.8
blue	red	blue	0.2
red	blue	red	0.6
blue	blue	red	0.4
red	blue	blue	0.3
blue	blue	blue	0.7

x_5	x_3	$p(x_5 \mid x_3)$
red	red	0.7
blue	red	0.3
red	blue	0.4
blue	blue	0.6

Directed Graphical Model

Example



$$p(x_1, x_2, x_3, x_4, x_5) =$$
$$p(x_5 \mid x_3) p(x_4 \mid x_2, x_3)$$
$$p(x_3) p(x_2 \mid x_1) p(x_1)$$

Quantitative Specification

If each variable is continuous, we could define Gaussian conditional distributions

$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1)$$
$$x_2 \sim \mathcal{N}(x_1, \sigma_2)$$
$$x_3 \sim \mathcal{N}(\mu_3, \sigma_3)$$
$$x_4 \sim \mathcal{N}(x_2 + x_3, \sigma_4)$$
$$x_5 \sim \mathcal{N}(x_3, \sigma_5)$$

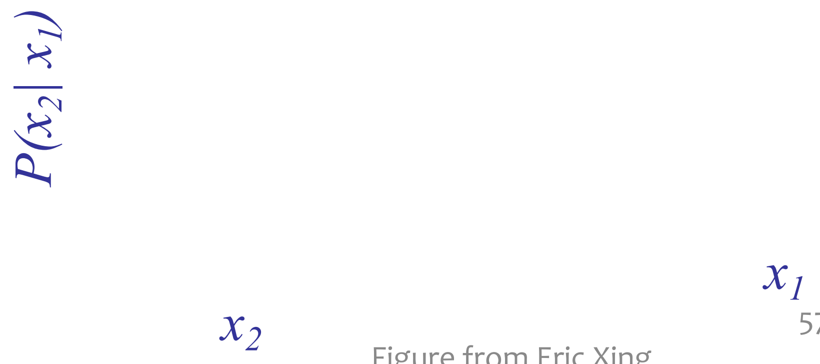


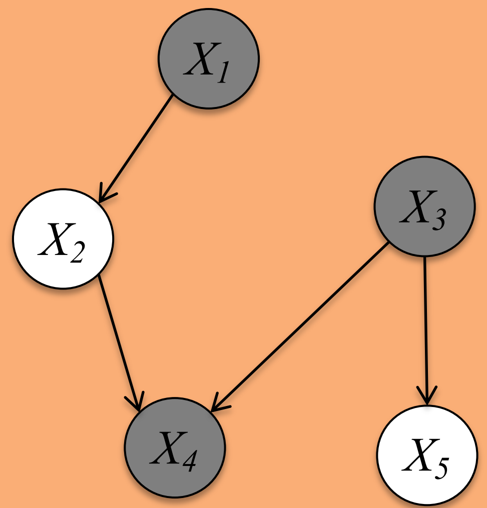
Figure from Eric Xing

Observed Variables

- In a graphical model, **shaded nodes** are "**observed**", i.e. their values are given

Example:

$$P(X_2, X_5 \mid X_1 = 0, X_3 = 1, X_4 = 1)$$



MARKOV MODEL

Markov Model

1. **1st-order Markov assumption:** for a sequence of random variables, the probability distribution over x_t random variables is conditionally independent of $x_1, \ldots, x_{t-2}$ given x_{t-1}

$$p(x_t \mid x_1, \ldots, x_{t-1}) = p(x_t \mid x_{t-1})$$

2. **1st-order Markov model:** defines a joint distribution over a sequence of variables using a Markov assumption

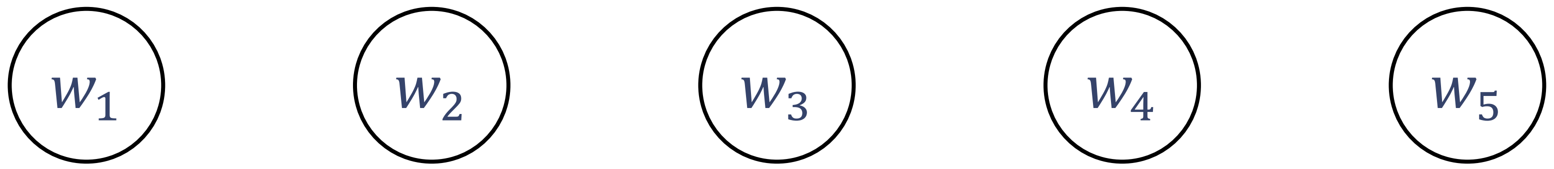
$$p(x_1, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1})$$

3. We can represent the Markov model as a **directed graphical model**

$$\begin{tikzcd} x_1 \arrow[r] & x_2 \arrow[r] & x_3 \arrow[r] & \cdots \arrow[r] & x_{T-1} \arrow[r] & x_T \end{tikzcd}$$

In-class Exercise: RNN-LM as a DGM

Given a five-word sequence, $[w_1, w_2, w_3, w_4, w_5]$, how could we represent the implied probability distributions of an RNN-LM as a directed graphical model?



In-class Exercise: RNN-LM as a DGM

Given a five-word sequence, $[w_1, w_2, w_3, w_4, w_5]$, how could we represent the implied probability distributions of an RNN-LM as a directed graphical model?

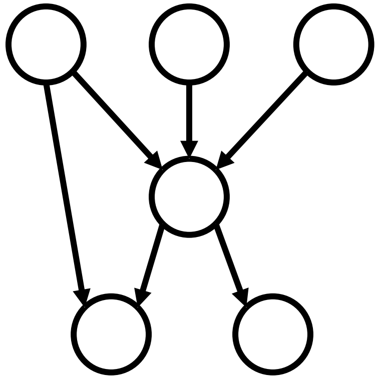
$$\begin{tikzcd}
w_1 \arrow[r] \arrow[rr, bend left] \arrow[rrr, bend left] \arrow[rrrr, bend left] & w_2 \arrow[r] \arrow[rr, bend left] \arrow[rrr, bend left] & w_3 \arrow[r] \arrow[rr, bend left] & w_4 \arrow[r] & w_5
\end{tikzcd}$$

$$p(w_1, \ldots, w_5) = p(w_5 \mid w_4, \ldots, w_1)\, p(w_4 \mid w_3, \ldots w_1) \cdots p(w_1)$$

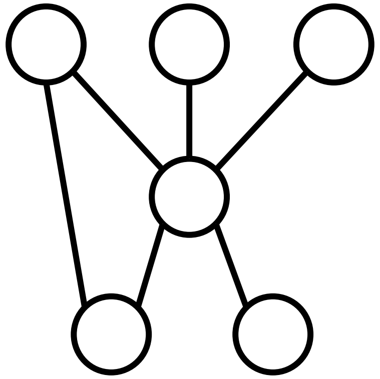
UNDIRECTED GRAPHICAL MODELS

Three Types of Graphical Models

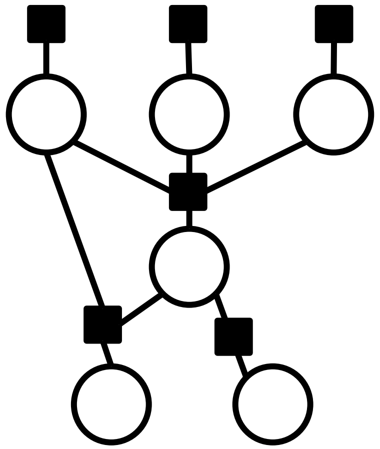
Directed Graphical Model



Undirected Graphical Model



Factor Graph



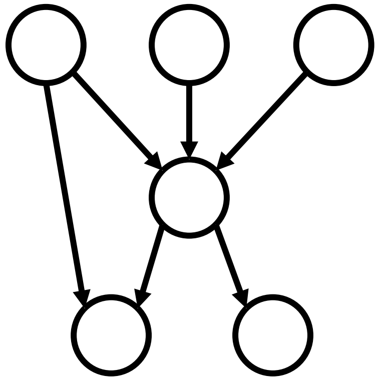
Undirected Graphical Models

Representation of both directed and undirected graphical models

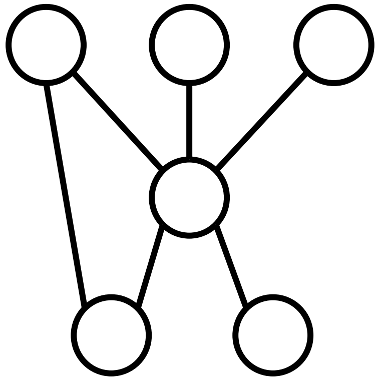
FACTOR GRAPHS

Three Types of Graphical Models

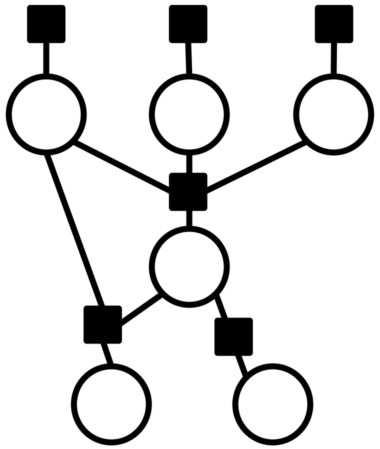
Directed Graphical Model



Undirected Graphical Model



Factor Graph



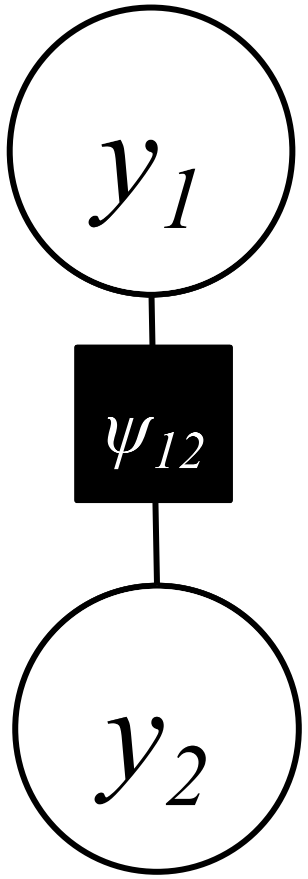
Mathematical Modeling

Factor Graphs

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)



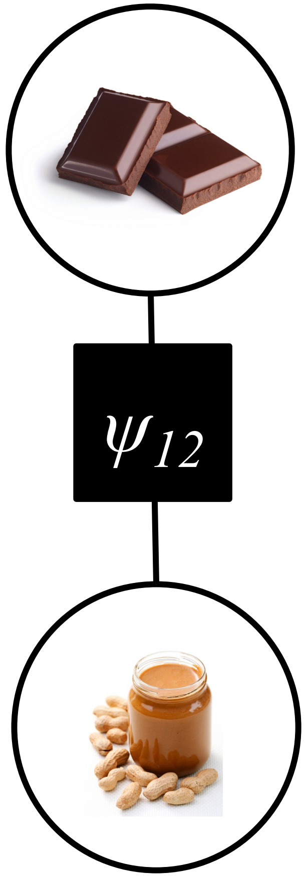
Mathematical Modeling

Factor Graphs

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)



Each **random variable** can be assigned a **value**

The collection of values for all the random variables is called an **assignment.**

Mathematical Modeling

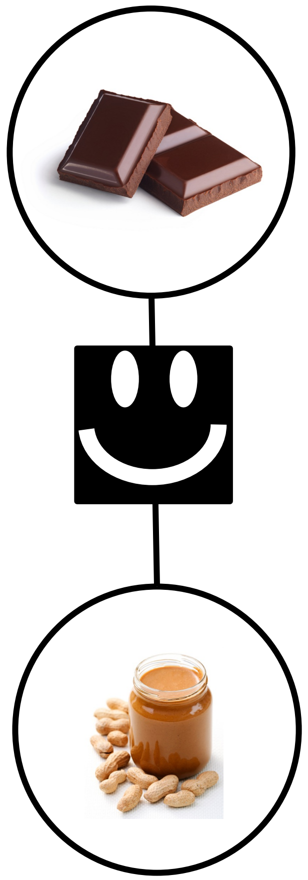
Factor Graphs

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)

Factors have local opinions about the assignments of their neighboring variables



Mathematical Modeling

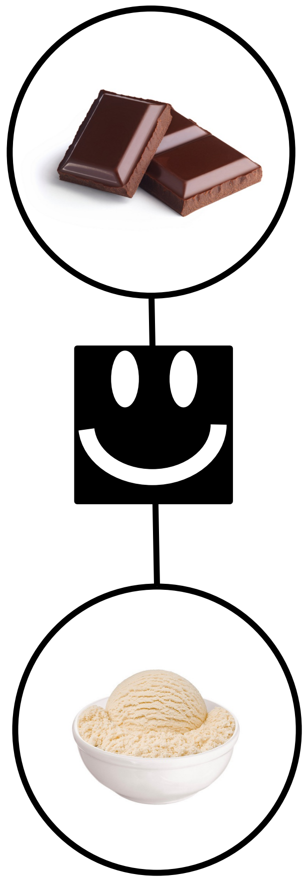
Factor Graphs

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)

Factors have local opinions about the assignments of their neighboring variables



Mathematical Modeling

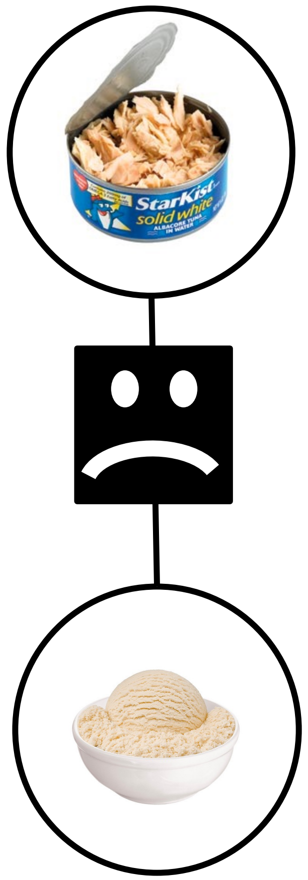
Factor Graphs

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)

Factors have local opinions about the assignments of their neighboring variables



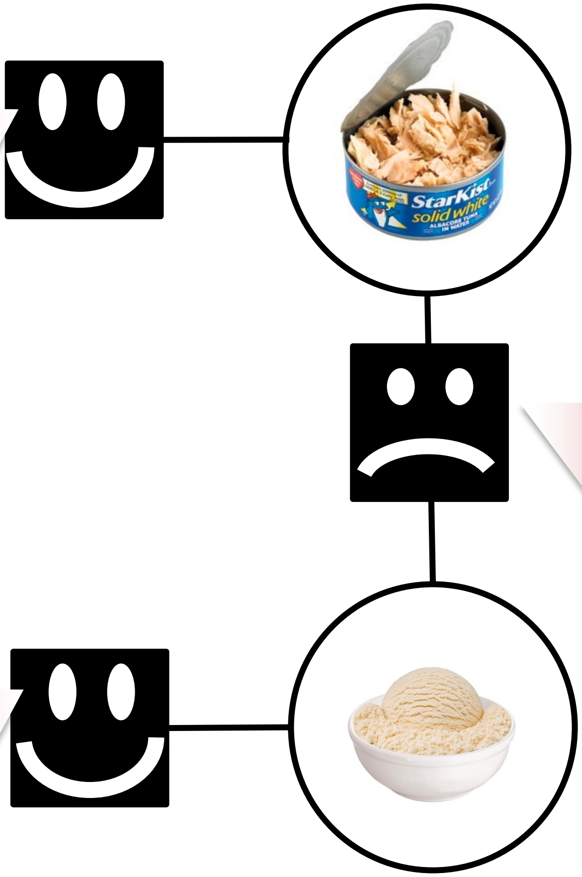
Mathematical Modeling

Factor Graphs

$P(\text{tuna, ice cream}) \;=\; ?$

Those opinions are expressed through **potential tables**

chocolate	0.1
peanut butter	5
ice cream	1
tuna	6
…	



chocolate	4
peanut butter	8
ice cream	7
tuna	3
…	

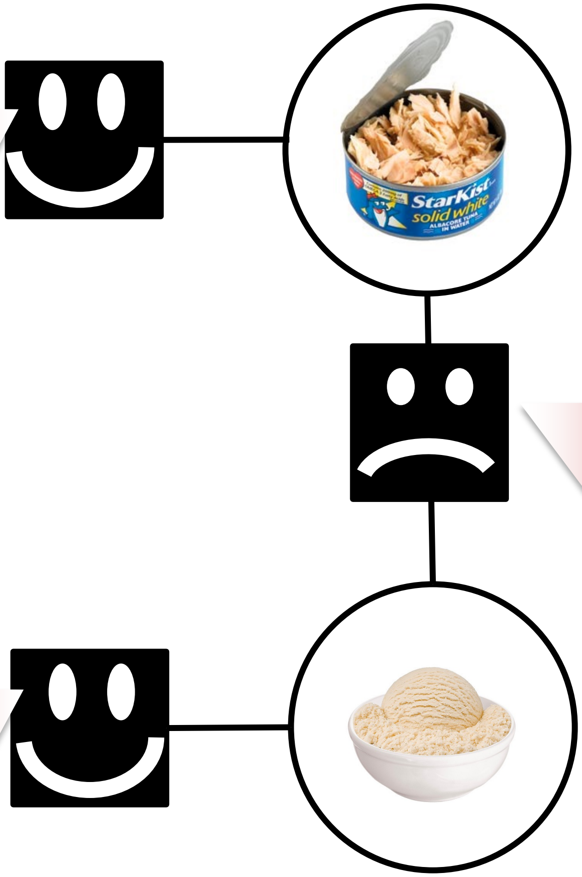
	chocolate	peanut butter	Ice cream	tuna	⋮
chocolate	2	9	7	0.1	
peanut butter	4	2	3	0.2	
ice cream	7	3	2	0.1	
tuna	0.1	0.2	0.1	2	
…					

Mathematical Modeling

Factor Graphs

$$P(\text{tuna, ice cream}) = \frac{1}{Z}(6 * 7 * 0.1)$$

chocolate	0.1
peanut butter	5
ice cream	1
tuna	6
…	



Uh-oh! The probabilities of the various assignments sum up to Z > 1.

So divide them all by Z.

	cho	peanu	Ice	tuna	⋮
chocolate	2	9	7	0.1	
peanut butter	4	2	3	0.2	
ice cream	7	3	2	0.1	
tuna	0.1	0.2	0.1	2	
…					

chocolate	4
peanut butter	8
ice cream	7
tuna	3
…	

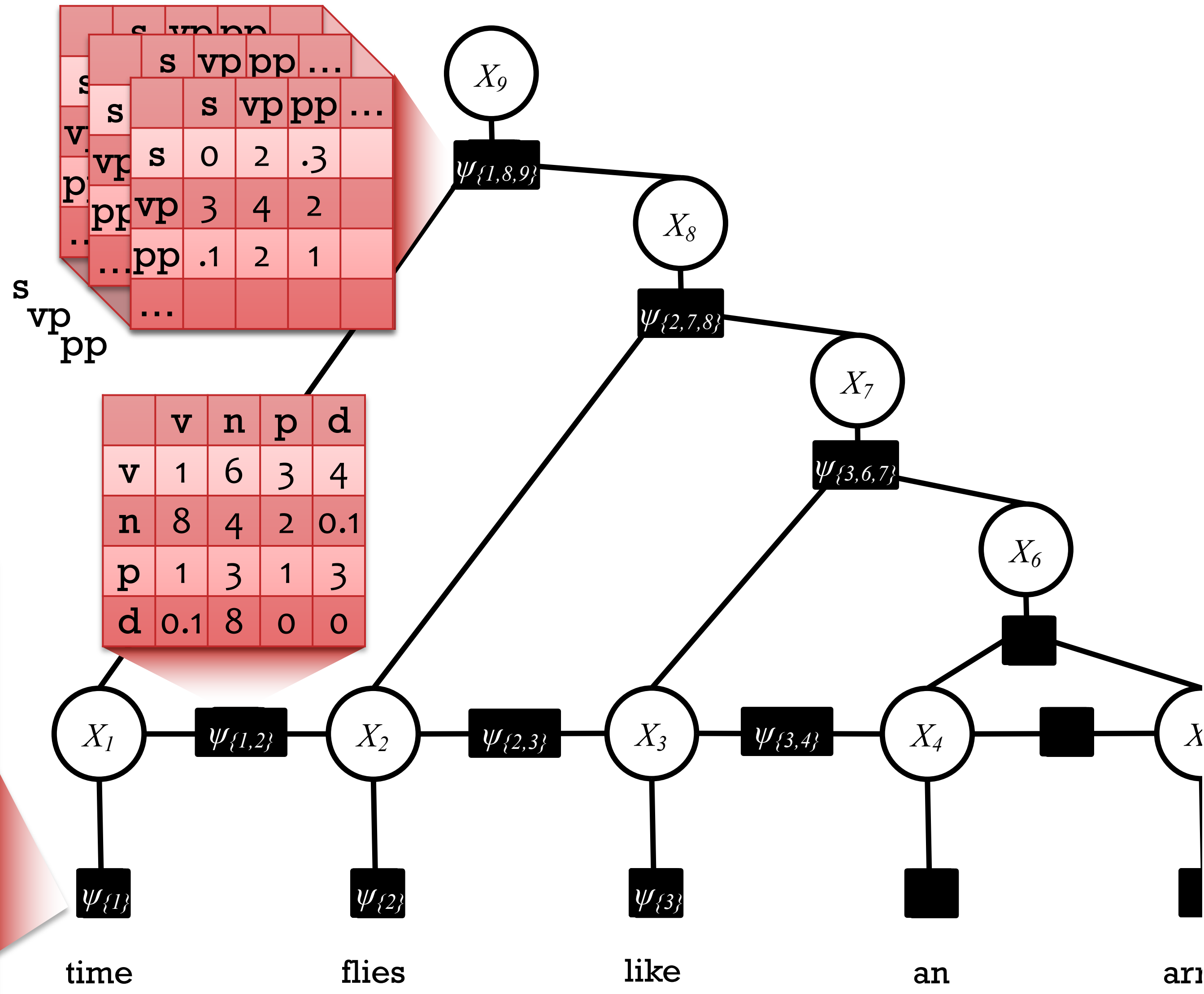
The combined potential tables of all factors defines the probability of an assignment

Factors are Tensors

- Factors must contain **non-negative** values -- this ensures we have a valid probability distribution
- We also sometimes refer to factors as **potential functions** or **potentials** (like UGMs)

Joint Distribution

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$$

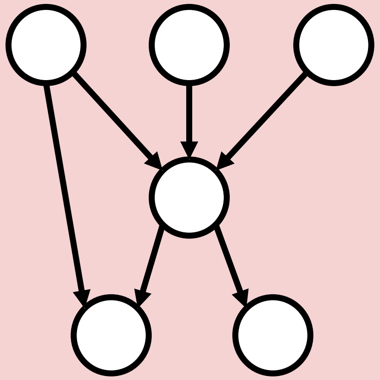


	s	vp	pp	…
s	0	2	.3	
vp	3	4	2	
pp	.1	2	1	
…				

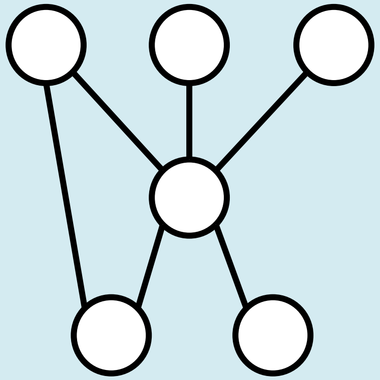
	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

Locally Normalized vs. Globally Normalized

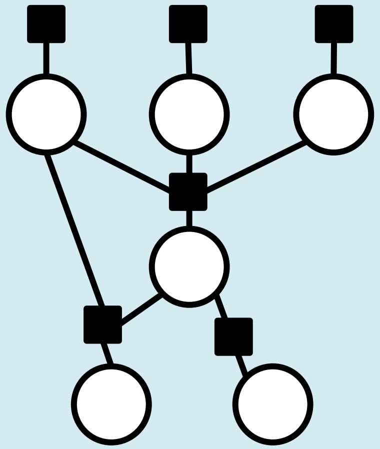
Directed Graphical Model



Undirected Graphical Model



Factor Graph



$$P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$$

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$$

Inference

Three Tasks:

1. Marginal Inference (#P-Hard)
Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad \Bigg| \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function (#P-Hard)
Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference (NP-Hard)
Compute variable assignment with highest probability

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

UNSUPERVISED LEARNING

Unsupervised Learning

Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Unsupervised Learning

Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

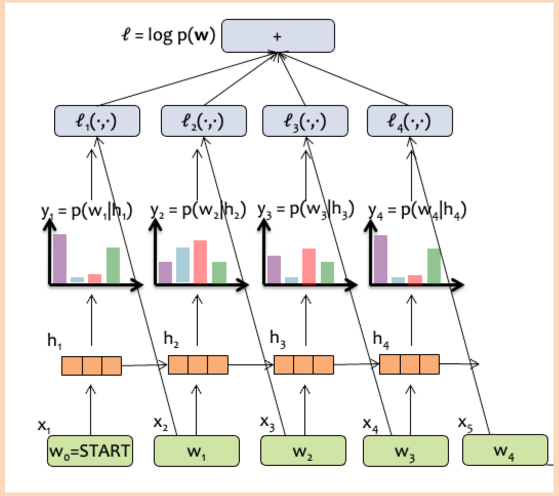
Example: autoregressive LMs

- true $p^*(\mathbf{x}_0)$ is the (human) process that produced text on the web
- choose $p_\theta(\mathbf{x}_0)$ to be an autoregressive language model
  - autoregressive structure means that $p(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) \sim \text{Categorical}(.)$ and ancestral sampling is exact/efficient
- learn by finding

  $$\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$$

  using gradient based updates on

  $$\nabla_\theta \log(p_\theta(\mathbf{x}_0))$$



Unsupervised Learning

Assumptions:

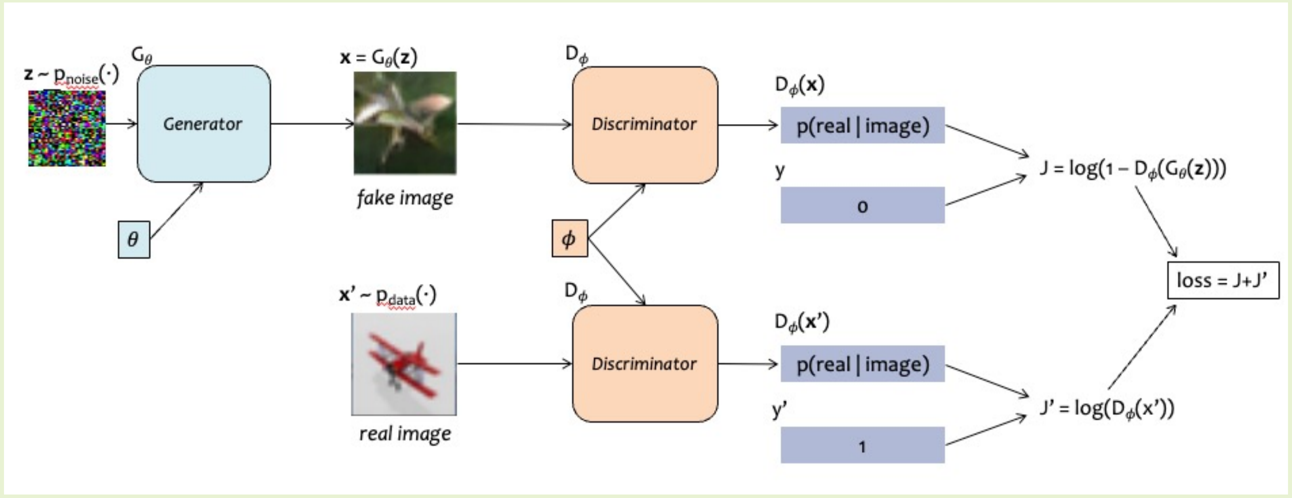
1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Example: GANs

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flikr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
  - sampling is typically easy: $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ and $\mathbf{x}_0 = f_\theta(\mathbf{z})$
- learn by finding $\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
  - No! Because we can't even compute $\log(p_\theta(\mathbf{x}_0))$ or its gradient
  - Why not? Because the integral is intractable even for a simple 1-hidden layer neural network with nonlinear activation

$$p_\theta(\mathbf{x}_0) = \int_z p_\theta(\mathbf{x}_0 \mid z)\, p(z)\, dz$$


$\mathbf{z} \sim p_{\text{noise}}(\cdot)$
G_θ
Generator
θ
$\mathbf{x} = G_\theta(\mathbf{z})$
fake image
D_ϕ
Discriminator
ϕ
$D_\phi(\mathbf{x})$
p(real | image)
y
0
$J = \log(1 - D_\phi(G_\theta(\mathbf{z})))$
loss = J+J'
$\mathbf{x}' \sim p_{\text{data}}(\cdot)$
real image
D_ϕ
Discriminator
$D_\phi(\mathbf{x}')$
p(real | image)
y'
1
$J' = \log(D_\phi(\mathbf{x}'))$

so optimize a minimax loss instead

Unsupervised Learning

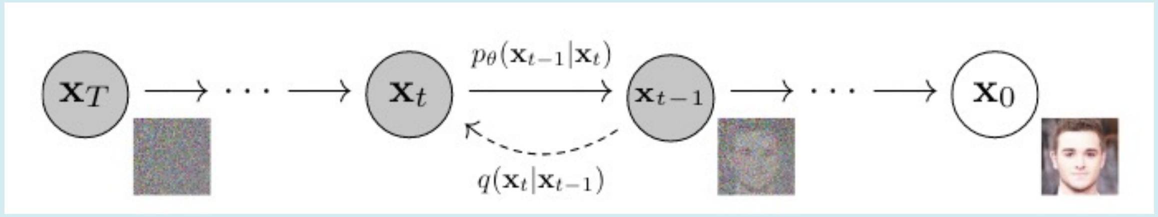
Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Example: VAEs / Diffusion Models

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flikr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
  - sampling is will be easy
- learn by finding $\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
  - Sort of! We can't compute the gradient $\nabla_\theta \log(p_\theta(\mathbf{x}_0))$
  - So we instead optimize a variational lower bound (more on that later)



$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure from Ho et al. (2020)

Latent Variable Models

- For GANs and VAEs, we assume that there are (unknown) **latent variables** which give rise to our observations
- The **vector z** are those latent variables
- After learning a GAN or VAE, we can **interpolate** between images in latent **z** space

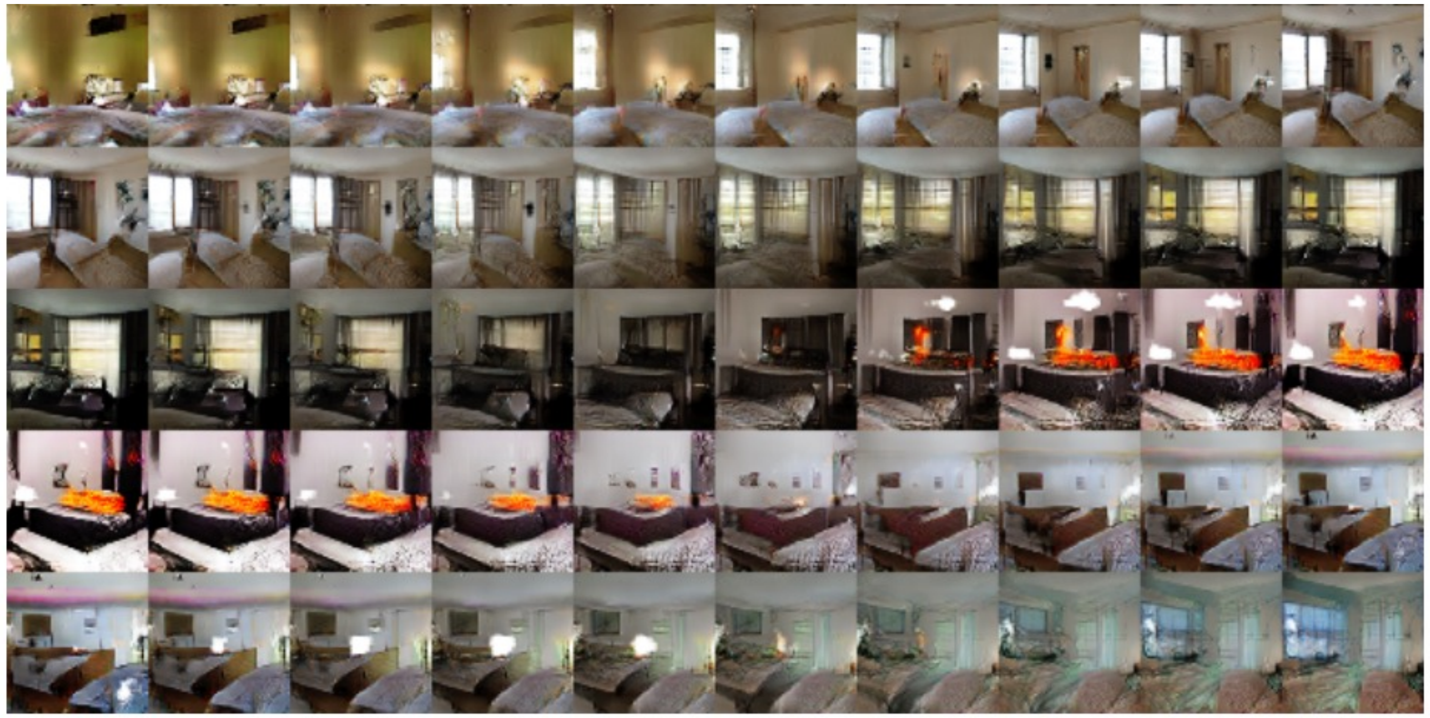


Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

Figure from Radford et al. (2016)

DIFFUSION MODELS AND VARIATIONAL AUTOENCODERS (VAES)

Diffusion Models

- Next we will consider (1) **diffusion models** and (2) **variational autoencoders (VAEs)**
  - Although VAEs came first, we're going to dive into diffusion models since they will receive more of our attention
- The steps in defining these models is roughly:
  - Define a probability distribution involving Gaussian noise
  - Use a variational lower bound as an objective function
  - Learn the parameters of the probability distribution by optimizing the objective function
- So what is a variational lower bound?

Diffusion Models

- Next we will consider (1) **diffusion mo**
  variational autoencoders (VAEs)
  - Although VAEs came first, we're going to
    models since they will receive more of ou
- The steps in defining these models is
  - Define a probability distribution involvin
  - Use a variational lower bound as an obje
  - Learn the parameters of the probabi
    the objective function
- So what is a variational lower bound?

The standard presentation of diffusion models requires an understanding of variational inference. (we'll do that next time)

Next time, we'll do an alternate presentation without variational inference!