

**Spring 2023**

# ADVANCED TOPICS IN COMPUTER VISION

---

**Atlas Wang**

Assistant Professor, The University of Texas at Austin

**Visual Informatics Group@UT Austin**  
<https://vita-group.github.io/>



# Outline



Basics: Why GAN and what it is?



The Optimization Problem of GANs



How to Train GANs: An Odyssey



How to Evaluate GANs?



An Application Tour of GANs



Summary and Open Challenges



Basics: Why GAN and  
what it is?

# Generative vs. Discriminative Models

---

- Given a distribution of inputs  $X$  and labels  $Y$ 
  - Discriminative: model the conditional distribution  $P(Y | X)$ .
  - Generative networks model the joint distribution  $P(X, Y)$ .
- If the model understands the joint distribution  $P(X, Y)$ , then it
  - can calculate  $P(Y | X)$  using Bayes rule
  - can perform other tasks like  $P(X | Y)$ , i.e., generating data from the label (called “conditional generation” in GANs)
  - understands the distribution better than a discriminative model, a scientific philosophy called **“analysis by synthesis”**

*“The analysis of the patterns generated by the world in any modality, with all their naturally occurring complexity and ambiguity, with the goal of reconstructing the processes, objects and events that produced them” - David Mumford*



# Generative vs. Discriminative Models

---

- Even if you only have  $X$ , you can still build a generative model
  - ... making generative modeling amendable to unsupervised/semi-supervised representation learning
  - Not every problem is discriminative, but all problems could be generative!
- **However, generative modeling is harder!**
  - Map from  $X$  to  $Y$  is typically many to one
  - Map from  $Y$  to  $X$  is typically one to many
  - Dimensionality of  $X$  typically  $\gg$  dimensionality of  $Y$
  - Hence compared to estimating  $P(Y | X)$ , the estimation of  $P(X, Y)$  usually runs into the “curse of dimensionality”

# Setup of Generative Models

**Discriminative model:** given  $n$  examples  $(x^{(i)}, y^{(i)})$   
learn  $h : X \rightarrow Y$

**Generative model:** given  $n$  examples  $x^{(i)}$ , recover  $p(x)$

**Maximum-likelihood objective:**  $\prod_i p_\theta(x) = \sum_i \log p_\theta(x)$

**Generation:** Sampling from  $p_\theta(x)$

# Autoregressive Models

**Factorize** dimension-wise:

$$p(x) = p(x_1)p(x_2|x_1)\dots p(x_n|x_1, \dots, x_{n-1})$$

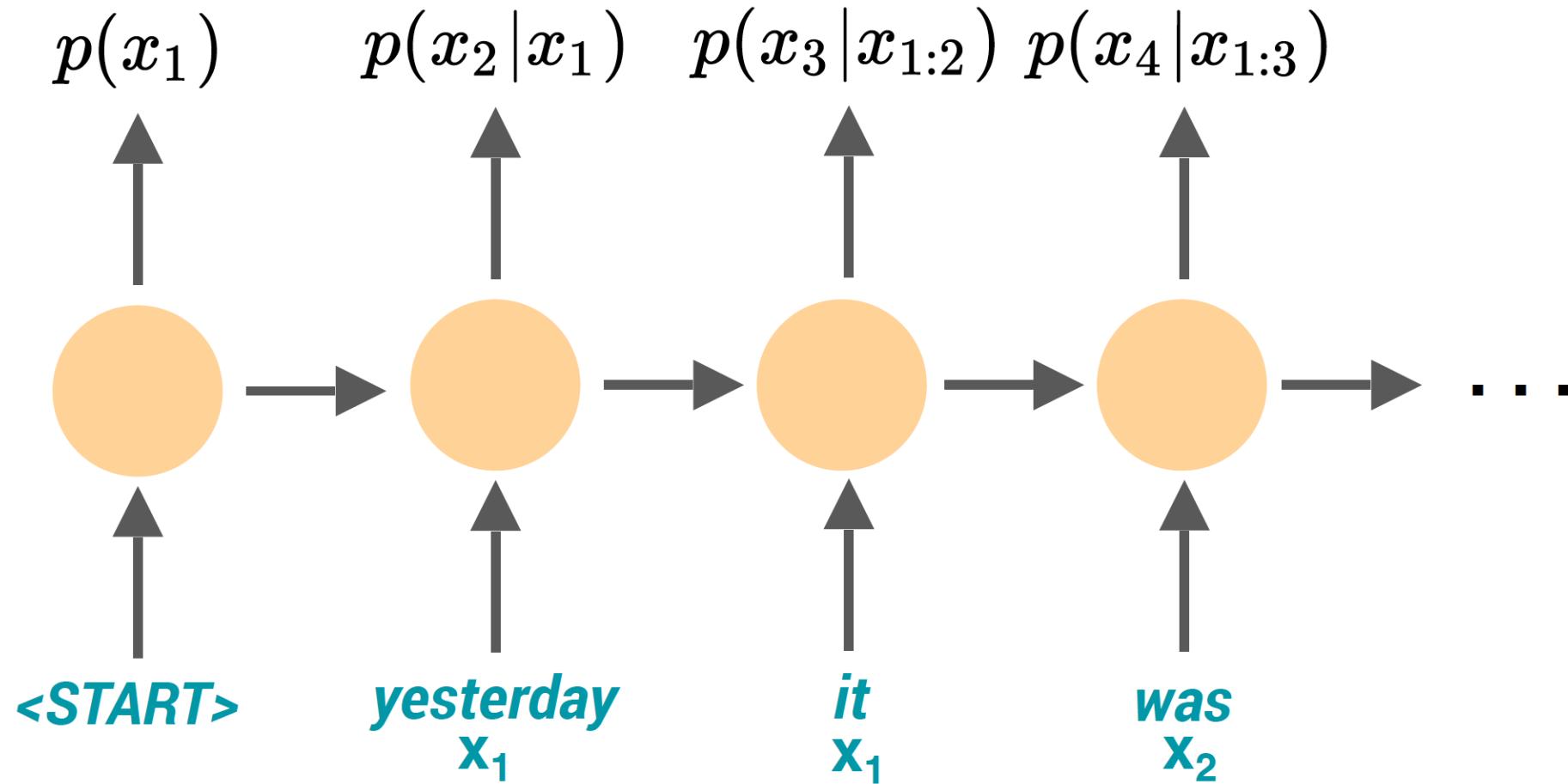
Build a “next-step prediction” model  $p(x_n|x_1, \dots, x_{n-1})$

If  $x$  is **discrete**, network outputs a probability for each possible value

If  $x$  is **continuous**, network outputs parameters of a simple distribution  
(e.g. Gaussian mean and variance)... *or just discretize!*

**Generation:** sample one step at a time, conditioned on all previous steps

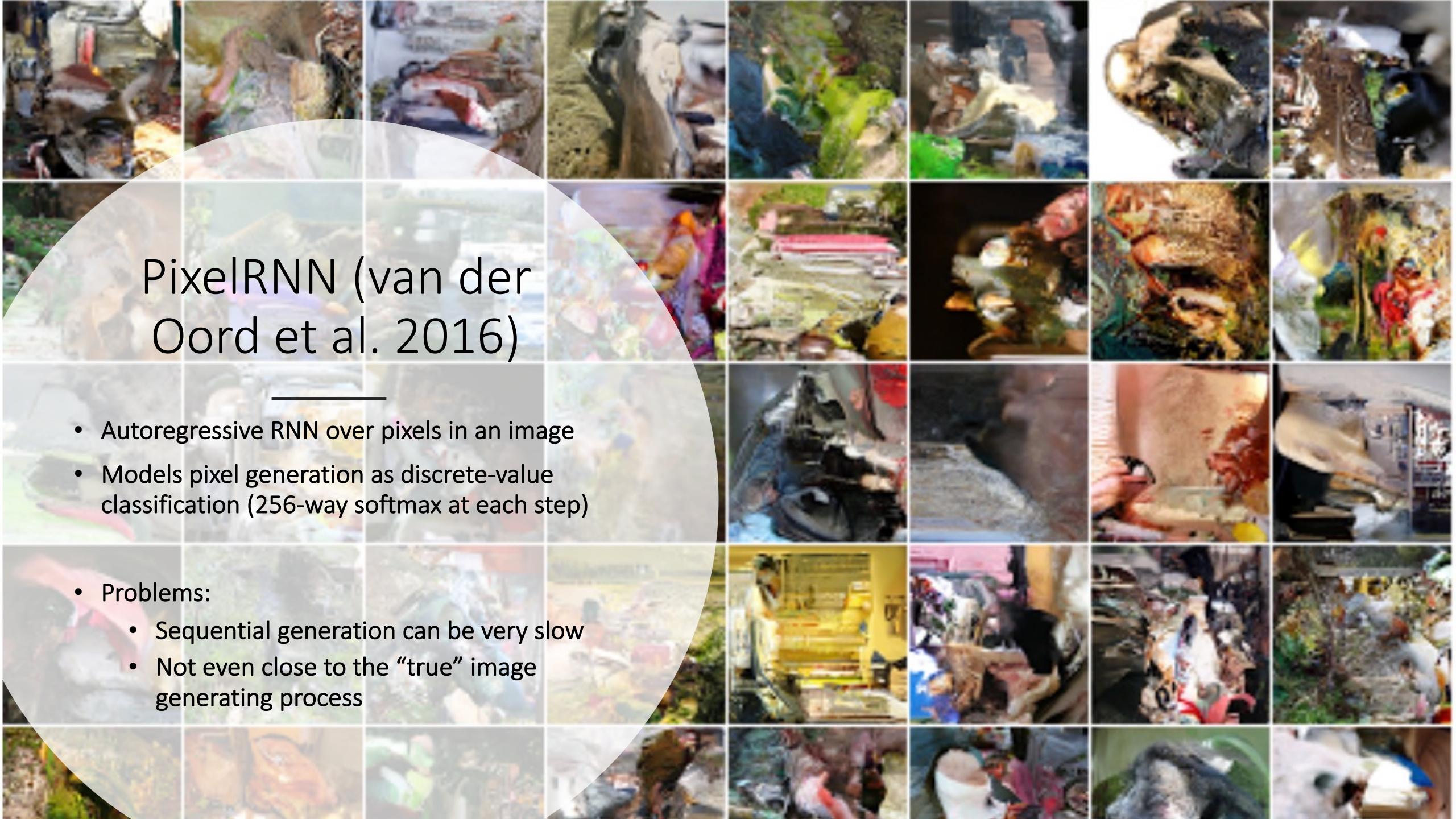
# RNNs for Autoregressive Language Modeling



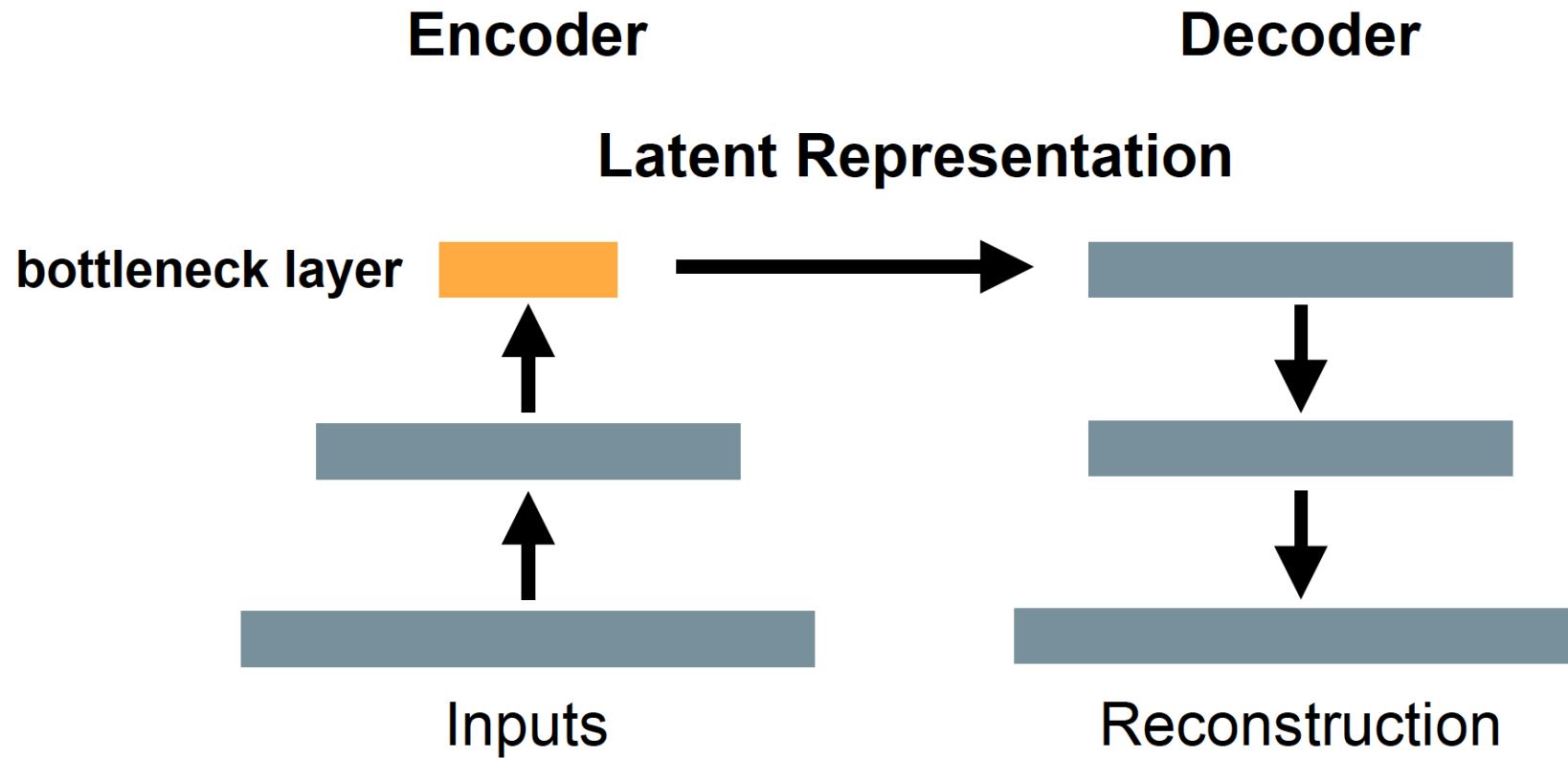
# PixelRNN (van der Oord et al. 2016)

- Autoregressive RNN over pixels in an image
- Models pixel generation as discrete-value classification (256-way softmax at each step)

- Problems:
  - Sequential generation can be very slow
  - Not even close to the “true” image generating process



# Autoencoders for Representation Learning



$$L = (x - \hat{x})^2$$

Idea: extending compression to implicit generative modeling, which allows for sampling!

# Variational Autoencoders (VAEs)

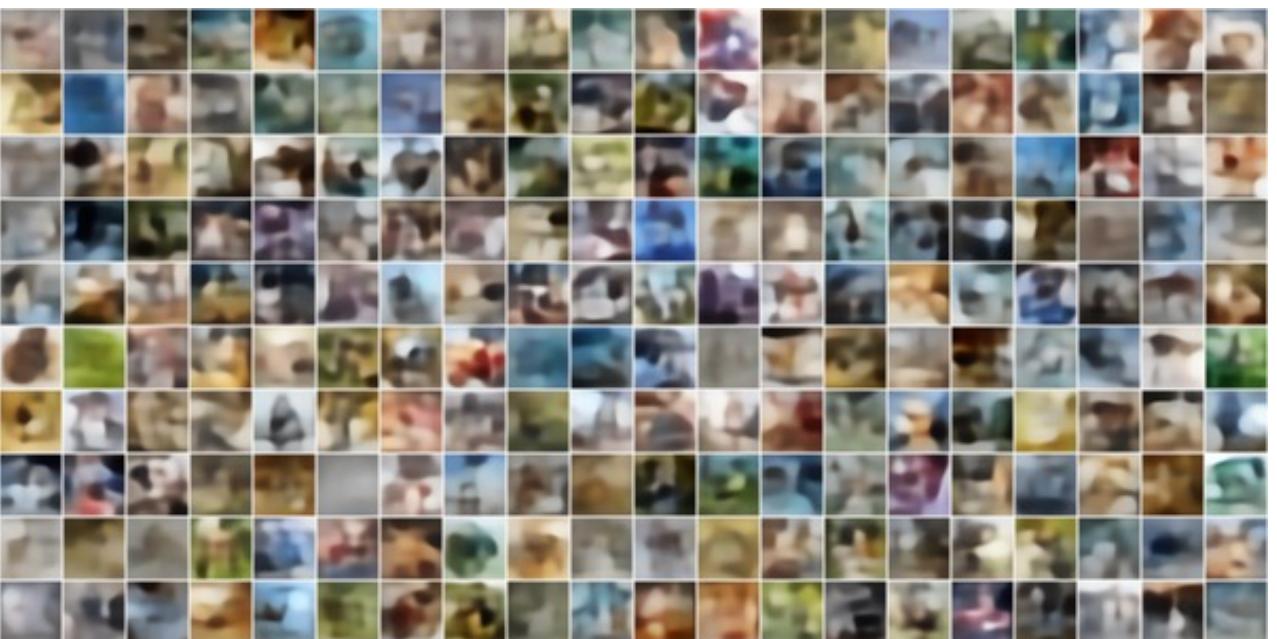
$$\begin{aligned}\log p(X) &= \log \int_Z p(X, Z) \\&= \log \int_Z p(X, Z) \frac{q(Z)}{q(Z)} \\&= \log \left( \mathbb{E}_q \left[ \frac{p(X, Z)}{q(Z)} \right] \right) \\&\geq \mathbb{E}_q \left[ \log \frac{p(X, Z)}{q(Z)} \right] \\&= \mathbb{E}_q [\log p(X, Z)] + H[Z]\end{aligned}$$

- Similar to a typical autoencoder, but allowing for sampling!
  - **Goal:** generative model of  $P(X)$
  - **Loss function:** reconstruct inputs  $X$  (in probabilistic sense)
  - **Encoder** models  $P(Z | X)$
  - **Decoder** models  $P(X | Z)$
- Hidden representation  $Z$  is to be learned by the model
  - We encourage the marginal distribution over  $Z$  to match **a prior  $Q(Z)$  (needs to be pre-chosen)**
  - During training: generated by encoder
  - During testing: sampled from  $Q(Z)$ , assuming  $\mathbb{E}_X P(Z | X) \approx Q(Z)$
- **Algorithm:** maximizing  $\log P(X)$   
-> maximizing its **evidence lower bound (ELBO)**
  - can also be re-expressed as minimizing  $\text{KL}(Q(Z) || P(Z | X))$
  - VAEs require an analytical understanding of the prior  $Q(Z)$

# VAE Results

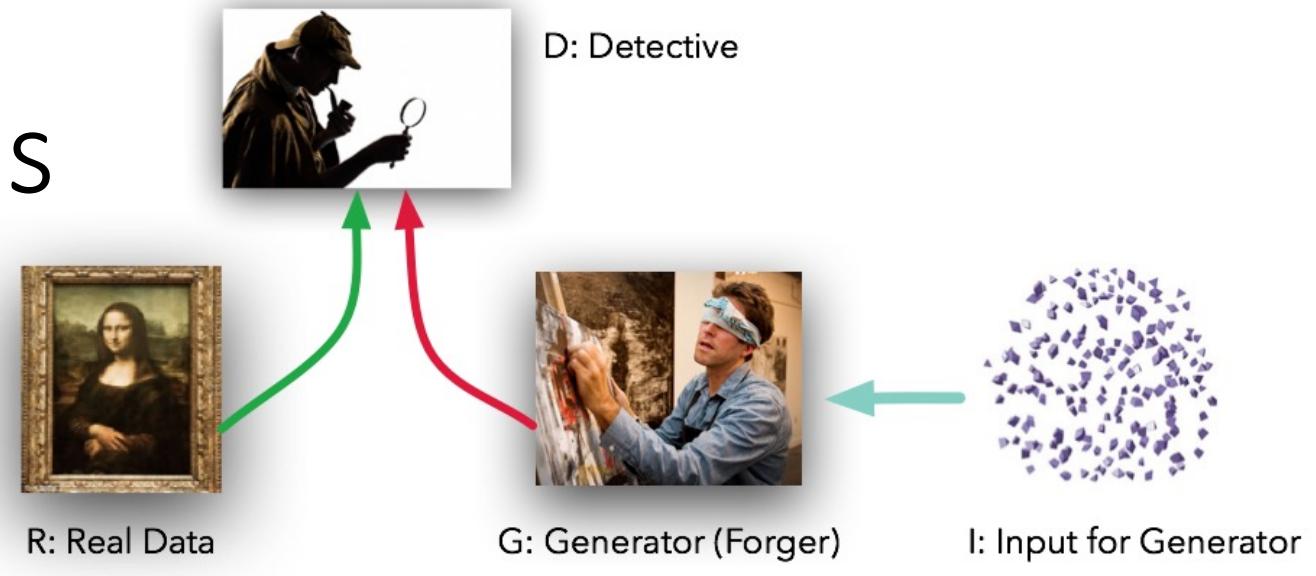
## Problems:

- Encoder and decoder's output distributions are typically limited (diagonal-covariance Gaussian or similar)
- This prevents the model from capturing fine details and leads to blurry generations



# GANs: NEW WORLD of generative models

**GAN** (generative adversarial network)  
= two competing modules:  
**G** (generator) + **D** (discriminator)



Ian Goodfellow  
@goodfellow\_ian

4.5 years of GAN progress on face generation.  
[arxiv.org/abs/1406.2661](https://arxiv.org/abs/1406.2661) [arxiv.org/abs/1511.06434](https://arxiv.org/abs/1511.06434)  
[arxiv.org/abs/1606.07536](https://arxiv.org/abs/1606.07536) [arxiv.org/abs/1710.10196](https://arxiv.org/abs/1710.10196)  
[arxiv.org/abs/1812.04948](https://arxiv.org/abs/1812.04948)





Donald J. Trump

@realDonaldTrump

You cannot trust CNN! They are FAKE!!!

RETWEETS

7,771

LIKES

2,094



11:07 AM - 21 Nov 2017



364



8K

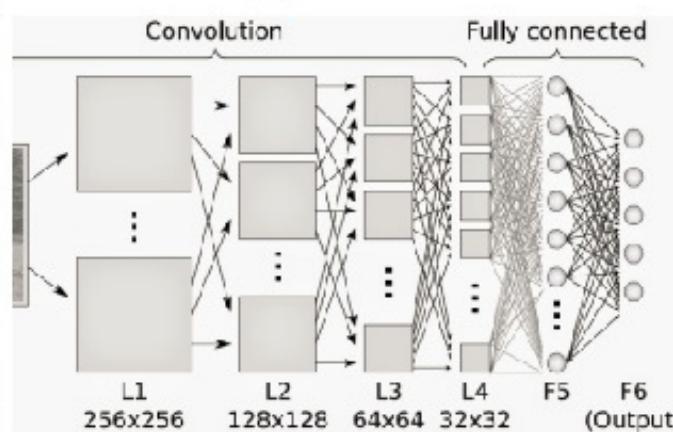


2K

**what people think  
he's referring to**



**what he's actually  
referring to**

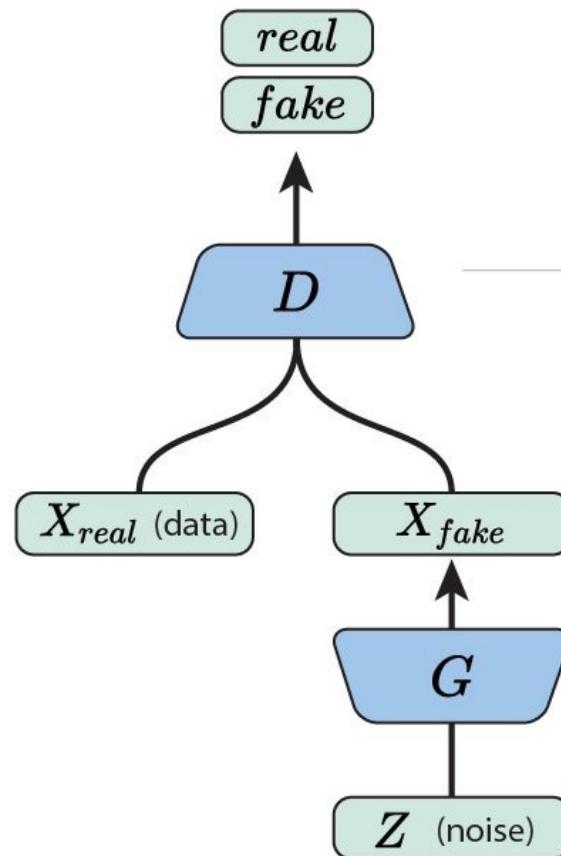


# Conceptual Diagram of GANs

## Quick Notes:

- **Same goal:** model  $P(X)$  !
- **G** learns  $P(X / Z)$
- **Challenge:** no simple loss function available to measure the divergence
- **Solution:** learning it, using **D** !
  - From the perspective of the **G**, **D** is like an adaptive loss function
  - In most applications, **D** is an auxiliary and thrown away after training; only **G** is wanted

**Generative Adversarial Networks** (GANs) are a way to make a generative model by having two neural networks compete with each other.



The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

The **generator** turns random noise into imitations of the data, in an attempt to fool the discriminator.

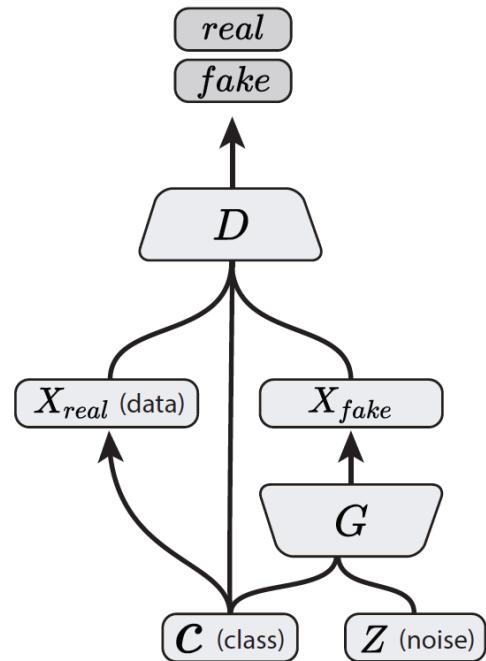
# GANs versus VAEs

- GANs minimize the divergence between the generated distribution and an unknown target distribution, in an actor-critic fashion
- Noisy, difficult and notoriously unstable optimization
- GANs only require the “black box” ability to sample from a prior
- GANs produce “sharper” results while VAE results are often blurry
- VAEs minimize a bound on the divergence between the generated distribution and a pre-specified target distribution
- Faster, reliable and theoretically justified optimization
- VAEs needs know the prior form as “white box”
- VAEs learn an encoder-decoder pair but GANs do not (or only decoder...)

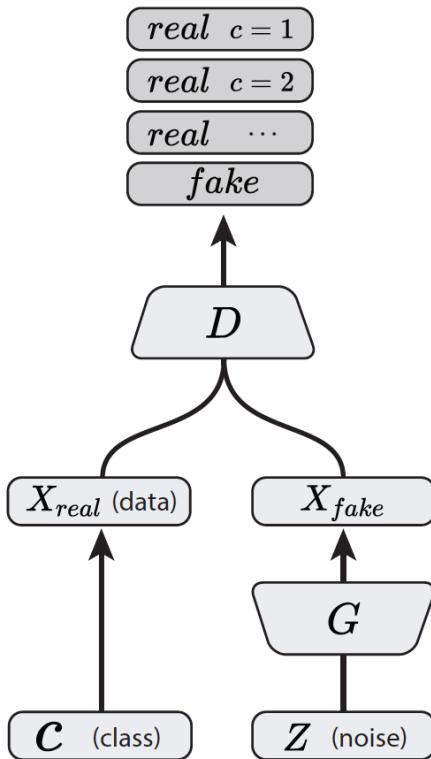
# GANs are Implicit Probabilistic Models

- Generator implicitly learns a target distribution  $P(X)$ , but has no explicit specification of the density function, which is different from VAEs
- Generator models  $P(X | Z)$ , and the implicitly learned distribution is defined naturally in terms of a sampling procedure: sampling from  $P(X)$  by drawing samples from  $P(Z)$  as input
  - If  $Z$  is a random noise: **noise-to-image GAN**
  - If  $Z$  is another natural image: **image-to-image GAN**
- It is not easy to marginalize over all  $Z$  and to calculate  $E_Z P(X | Z)$  explicitly
  - If you really want the explicit likelihood estimation: Bayesian GAN (extra MC sampling), flow-GAN (flow-based generator), etc.

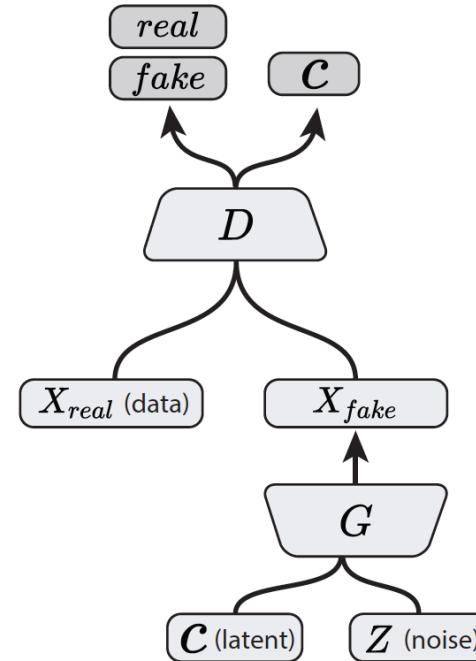
# From Unsupervised to Supervised: *Conditional GAN and Beyond*



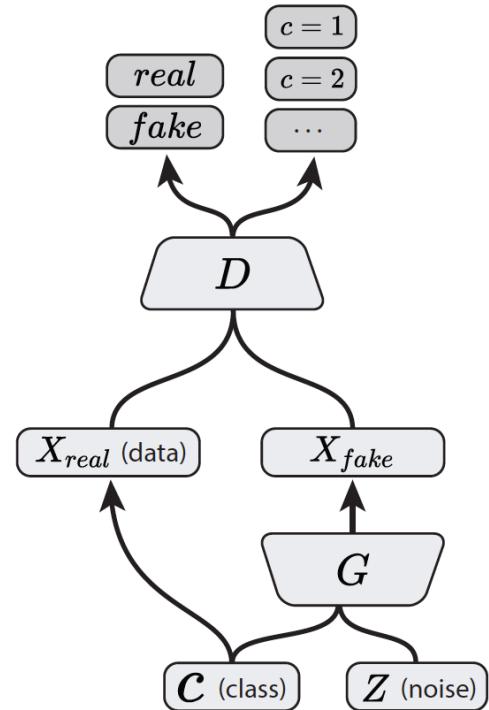
Conditional GAN  
(Mirza & Osindero, 2014)



Semi-Supervised GAN  
(Odena, 2016; Salimans, et al., 2016)



InfoGAN  
(Chen, et al., 2016)



AC-GAN  
(Present Work)



# The Optimization Problem of GANs

# Formulation: Min-Max Game

A GAN is defined by the following min-max game

$$\min_G \max_D V(D, G) = \mathbb{E}_X \log D(X) + \mathbb{E}_Z \log(1 - D(G(Z)))$$

- $D$  wants  $D(X) = 1$  and  $D(G(Z)) = 0$
- $G$  wants  $D(G(Z)) = 1$

# Min-Max Optimal Discriminator

---

What is the optimal discriminator?

$$\begin{aligned} f &:= \mathbb{E}_{X \sim P_D} \log D(X) + \mathbb{E}_{X \sim P_G} \log(1 - D(X)) \\ &= \int_X [P_D(X) \log D(X) + P_G(X) \log(1 - D(X))] dX \end{aligned}$$

Assuming we have an ideal function for the discriminator, it can output a different value for every  $X$ . So we optimize the following for each  $X$ .

$$[P_D(X) \log D(X) + P_G(X) \log(1 - D(X))]$$

# Min-Max Optimal Discriminator

$$\frac{\partial f}{\partial D(X)} = \frac{P_D(X)}{D(X)} - \frac{P_G(X)}{1 - D(X)} = 0$$

$$\frac{P_D(X)}{D(X)} = \frac{P_G(X)}{1 - D(X)}$$

$$(1 - D(X))P_D(X) = D(X)P_G(X)$$

$$D(X) = \frac{P_D(X)}{P_G(X) + P_D(X)}$$

# What does the optimal discriminator look like?

$$\begin{aligned} f &:= \mathbb{E}_{X \sim P_D} \log D(X) + \mathbb{E}_{X \sim P_G} \log(1 - D(X)) \\ &= \mathbb{E}_{P_D} \log \frac{P_D(X)}{P_G(X) + P_D(X)} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{P_G(X) + P_D(X)} \\ &= \mathbb{E}_{P_D} \log \frac{P_D(X)}{2m(X)} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{2m(X)} \\ &= \mathbb{E}_{P_D} \log \frac{P_D(X)}{m(X)} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{m(X)} - \log 4 \\ &= KL(P_D \| m) + KL(P_G \| m) - \log 4 \\ &= 2 \left( \frac{1}{2} KL(P_D \| m) + \frac{1}{2} KL(P_G \| m) \right) - \log 4 \\ m(X) &:= \frac{P_D(X) + P_G(X)}{2} \end{aligned}$$

# Jensen-Shannon Divergence

- The optimal discriminator will minimize the Jensen-Shannon divergence between the real and generated distributions
  - JS divergence is the averaged KL between A / B and their “averaged distribution”
  - This is called “*virtual training criterion*”, minimized if and only if  $P_D = P_G$

$$J = \min_G 2JSD(P_D \| P_G) - \log 4$$

$$JSD(A \| B) := \frac{1}{2}KL\left(A\middle\| \frac{A+B}{2}\right) + \frac{1}{2}KL\left(B\middle\| \frac{A+B}{2}\right)$$

**Remark:**

An optimally trained **D** just calculates the JS divergence ... but a **real D** calculates something much more complicated, e.g., “perceptual distance” ...

- $P_D = P_G$  makes a **minimax stationary point!**
  - If the generated data exactly matches the real data, **D** should output 0.5 for all inputs.
  - If **D** outputs 0.5 for all inputs, the gradient to the generator is flat, so the generated distribution has no reason to change.

# Stationary point might not be stable point

- The “optimal” GAN solution might be difficult to reach, but easy to slip away ...
- If the generated data is near the real, the discriminator outputs might be arbitrarily large
  - Even when real data and generated data are separated by some minimal distance, a discriminator with unlimited capacity can still assign an arbitrarily large distance between these distributions.
  - **Motivating:** *gradient penalty, Lipchitz constraint, Wasserstein loss* ... (later this talk)
- Generator may “overshoot” some values or “oscillate” around an optimum – a notorious behavior in minimax optimization
  - Whether those oscillations converge or not depends on training details
  - GAN training can be very sensitive to hyperparameters

# Challenge of Minimax Optimization

- The hard part is that both generator and discriminator need to be trained simultaneously to hit “moving targets”
  - **Ideally:** assuming optimal discriminator at any time
  - **Practically:** never going to happen!
  - If the discriminator is under-trained, it provides incorrect information to the generator
  - If the discriminator is over-trained, there is nothing local that a generator can do to marginally improve
  - The correct discriminator changes during training
- Significant research on techniques, tricks, etc. to help stabilize training



# How to Train GANs: An Odyssey

# GAN training: Scaling up & Stabilizing



- Backbone algorithm: gradient ascent & descent
- Often combining many techniques to work well
- As of now, no silver-bullet to kill all training
- Getting better every year

- a) Controlling gradient & Improving loss smoothness
- b) Breaking end-to-end learning into progressive training
- c) Finding better architectures
- d) Other improved techniques and tricks

# Gradient & Loss Smoothness (1)

- Deep Regret Analytic Generative Adversarial Networks (2017)

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0, cI)} [\|\nabla_{\mathbf{x}} D_{\theta}(x + \delta)\| - k]^2$$

- New penalty to minimize the norm of the gradient in a region around real data
- That makes a function smoother
  - Smoothening in a random region around real data to smooth the discriminator

# Gradient & Loss Smoothness (2)

- Energy-based GAN/Hinge Loss (2017)

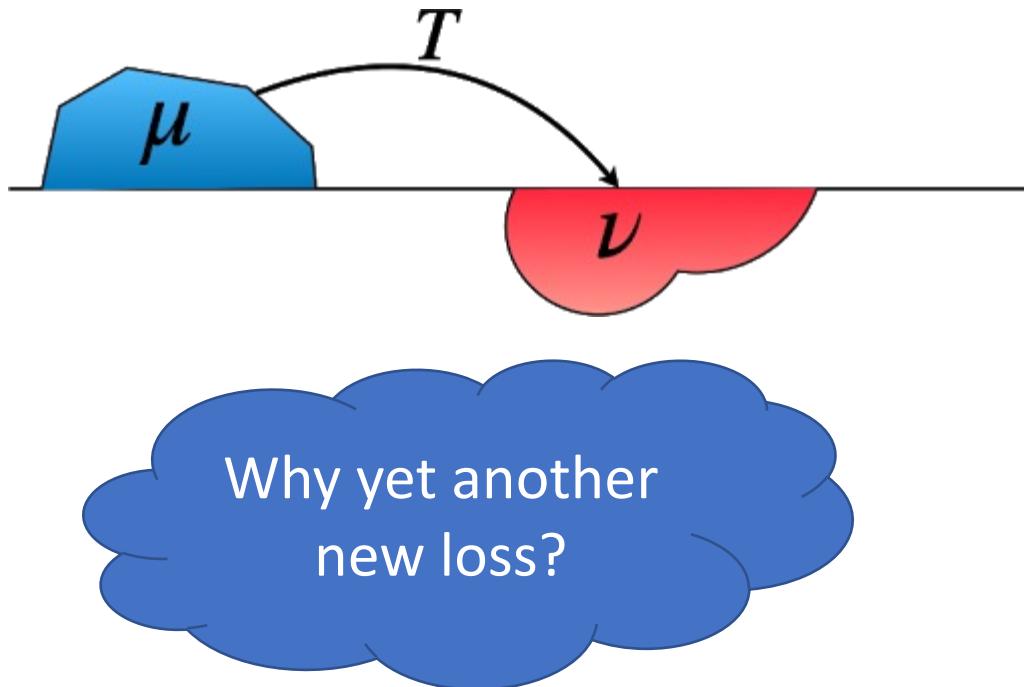
$$L_D = \mathbb{E}_X D(X) + \mathbb{E}_Z [m - D(G(Z))]^+$$

$$L_G = D(G(z))$$

- Simplify the loss function, removing powers and exponents
- No longer easily described using JS divergence or something similar
- Gradients are neither squashed nor exploding

# Gradient & Loss Smoothness (3)

- Wasserstein GAN (2017)

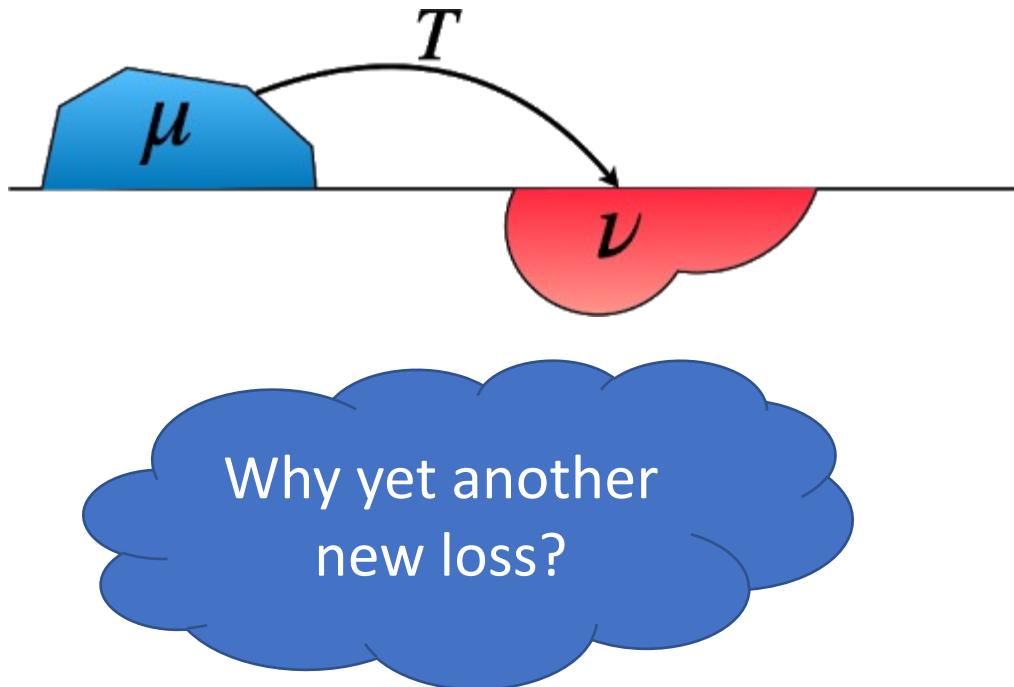


**K-L Divergence:**  $KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)}$

- Real data is a point mass at 0
- Generated data is a point mass at  $\theta$
- If  $\theta \neq 0$ ,  $p(0) \log \frac{p(0)}{q(0)} = 1 \frac{1}{0} = \infty$
- If  $\theta = 0$ ,  $1 \log \frac{1}{1} = 0$
- Not differentiable w.r.t.  $\theta$

# Gradient & Loss Smoothness (3)

- Wasserstein GAN (2017)



## JS Divergence:

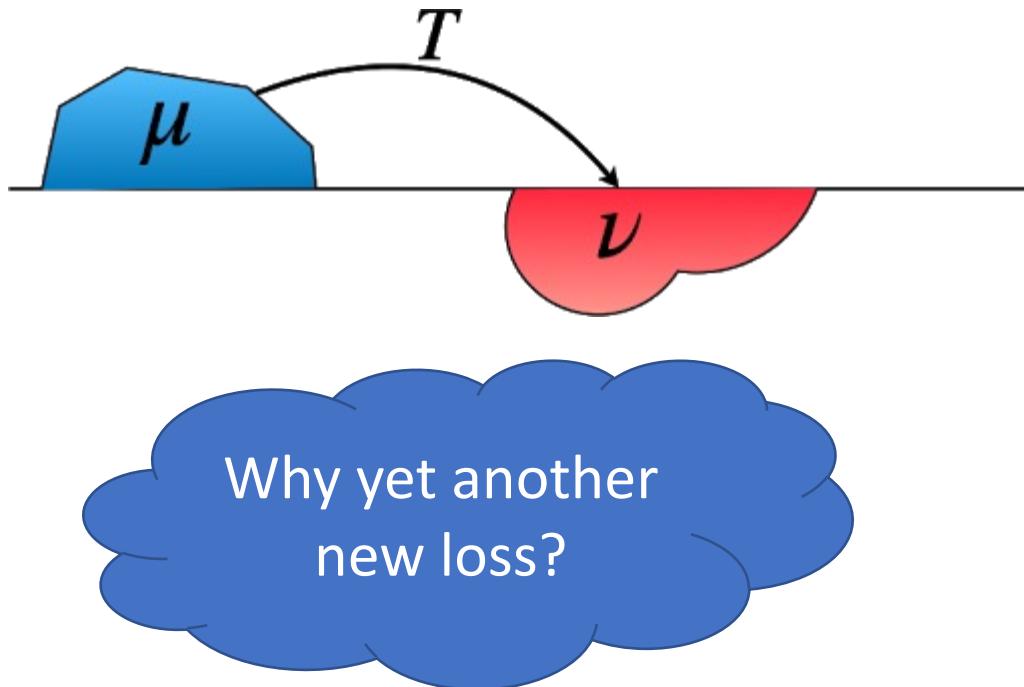
Calculate the average distribution and calculate the average KL  
the average.

$$m(x) = \frac{p(x) + q(x)}{2}$$
$$JS(p\|q)$$

- Real data is a point mass at 0
- Generated data is a point mass at  $\theta$
- If  $\theta \neq 0$ ,  $\frac{1}{2} [1 \log \frac{1}{0.5} + 1 \log \frac{1}{0.5}] = \log 4$
- If  $\theta = 0$ ,  $\frac{1}{2} [1 \log \frac{1}{1} + 1 \log \frac{1}{1}] = 0$
- Not differentiable w.r.t.  $\theta$

# Gradient & Loss Smoothness (3)

- Wasserstein GAN (2017)



## Wasserstein Distance:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

Calculate the mass times the distance in one dimension

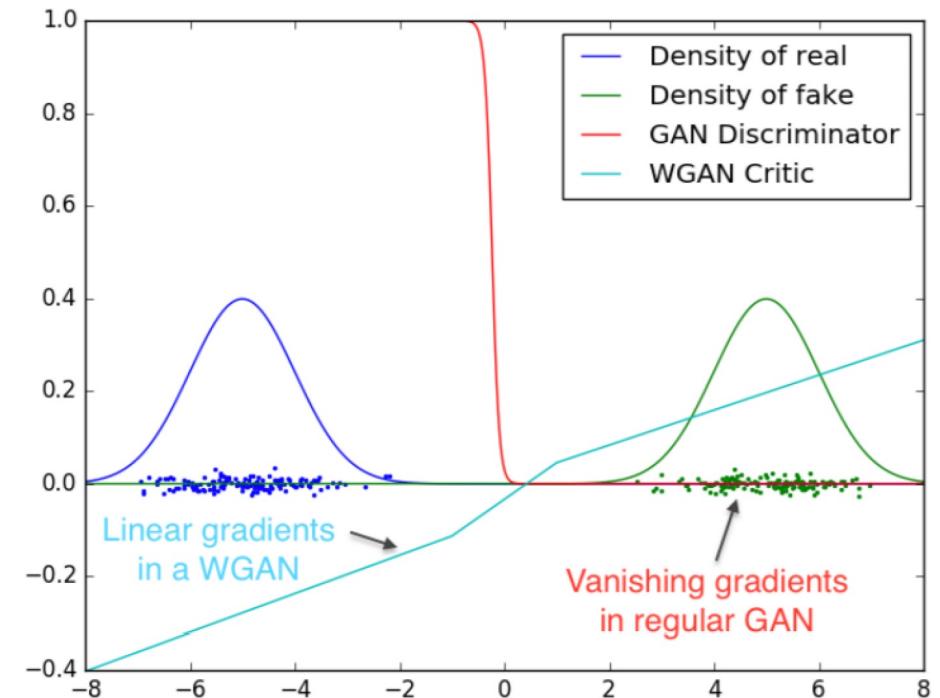
- Real data is a point mass at 0
- Generated data is a point mass at  $\theta$
- EM is  $|\theta|$
- **Differentiable w.r.t.  $\theta$ !**

# Gradient & Loss Smoothness (3)

- In general: Wasserstein metric provides a smooth measure, that stabilizes gradient descent
- Intractable -> Using Kantorovich-Rubinstein duality:

$$L_D = \mathbb{E}_X D(X) - \mathbb{E}_Z D(G(Z))$$

$$L_G = \mathbb{E}_Z D(G(Z))$$



- Further simplified and theoretically grounded.
- Calculating using a dual method needs **constrain the Lipschitz of discriminator**
  - Initially, clipping weights to some value -> but often leading to poor discriminators

# Gradient & Loss Smoothness (4)

- Wasserstein GAN + Gradient Penalty (2017)

$$L = \mathbb{E}_X(D(X)) - \mathbb{E}_Z(D(G(Z))) + \lambda \mathbb{E}_{X'} (\|\nabla D(X')\|_2 - 1)^2$$

- Property: A differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere
- Propose a better alternative of constraining the Lipschitz
- Scale up WGAN to much deeper G, e.g., ResNet-101

$$\sigma(A) := \max_{\mathbf{h}: \mathbf{h} \neq 0} \frac{\|A\mathbf{h}\|_2}{\|\mathbf{h}\|_2} = \max_{\|\mathbf{h}\|_2 \leq 1} \|A\mathbf{h}\|_2$$

- Spectral Normalization GAN (2018)

- Lipschitz of an MLP can be upper bounded by product of each layer's spectral norm
- ... and each layer's Lipschitz approximated by power iteration

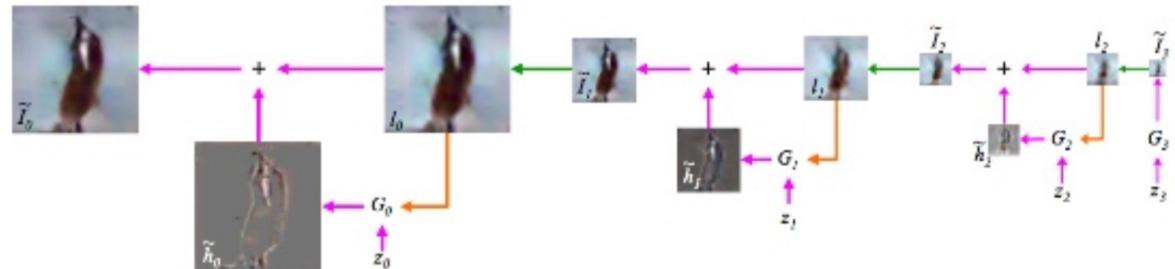
$$\|f\|_{\text{Lip}} \leq \prod_{l=1}^{L+1} \sigma(W^l)$$

$$\bar{W}_{\text{SN}}(W) := W / \sigma(W)$$

# Progressive Training

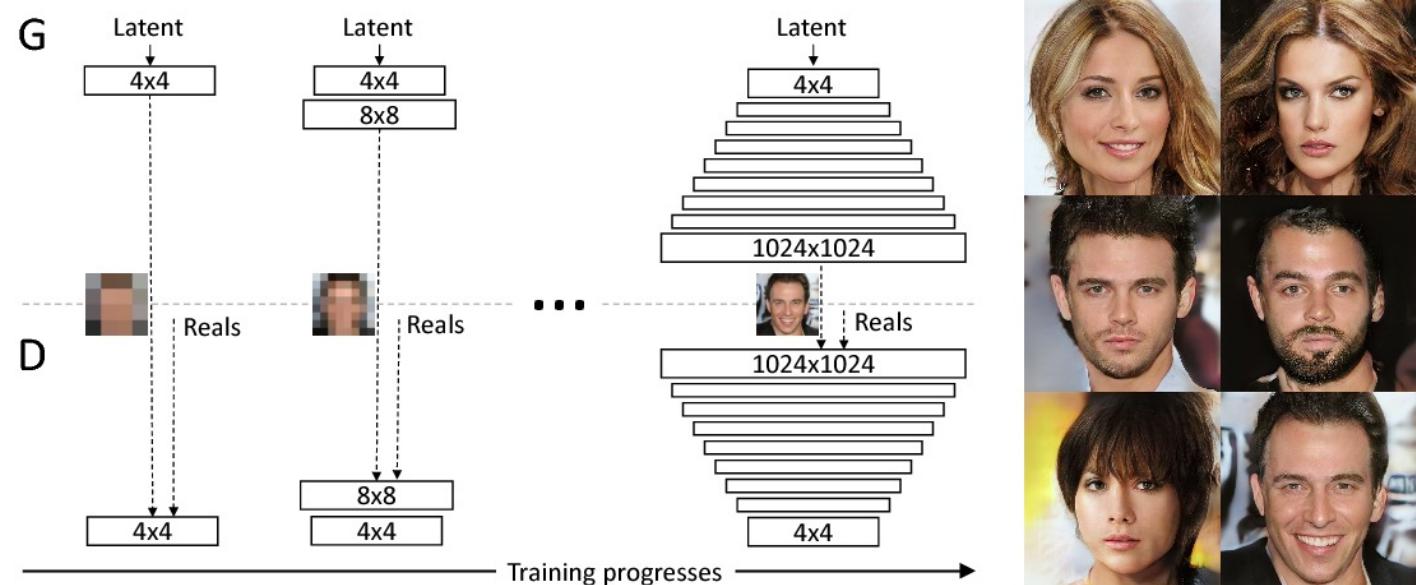
- Laplacian GAN (2014)

- The first GAN block generates small, blurry image
- ... following by conditional GAN blocks, sharpening and enlarging image slightly
- Repeat until desired size. Training is simplified/more stable.



- Progressive GAN (2017)

- Gradually add layers to generator and discriminator to produce larger images
- Always keep the two “balanced”



# Other improved techniques and tricks (1)

- Feature Matching (2016)

- Intuition: High-dimensional statistics of generated images should match statistics of real images
  - Discriminator produces multidimensional output, a “statistic” of the data, that should be more stable than scalar-value output (yes or no)
  - Generator trained to **minimize** L2 between real and generated data
  - Discriminator trained to **maximize** L2 between real and generated data
- $$\left\| \mathbb{E}_X D(X) - \mathbb{E}_Z D(G(Z)) \right\|_2^2$$

- Minibatch Discrimination (2016)

- Discriminator can look at multiple inputs at once and decide if those inputs come from the real or generated distribution
- More confident and less noisy than once per time
- If not, GANs can collapse to a single point

# Other improved techniques and tricks (2)

- **Historical Weight Averaging (2016)**

- Dampen oscillations by encouraging updates to converge to a mean
- Add a decay term that encourages the current parameters to be near a moving average:

$$\left\| \theta - \frac{1}{t} \sum_i^t \theta_i \right\|_2^2$$

- **One-sided label smoothing (2016)**

- Label smoothing is a common technique to avoid over-confident predictions/overfitting
- Smoothening for real targets but not the generated, when training the discriminator

- **Virtual Batch Normalization (2016)**

- Batch normalization accelerates convergence, but hard to apply in GANs (i.e., adding batch norm to the discriminator usually hurts the performance)
- VBN collects statistics on a fixed batch of real data and use to normalize other data



# How to Evaluate GANs?



# GAN Evaluation is HARD

---

- The task of generating realistic-looking images is not as easily quantified as a task like correctly labeling images
- The generator's learned distribution is implicit, and we cannot easily/directly calculate the likelihood of a test set
- **So How?**
  - **Human Evaluation**
  - **Sampling-based Proxy Methods**

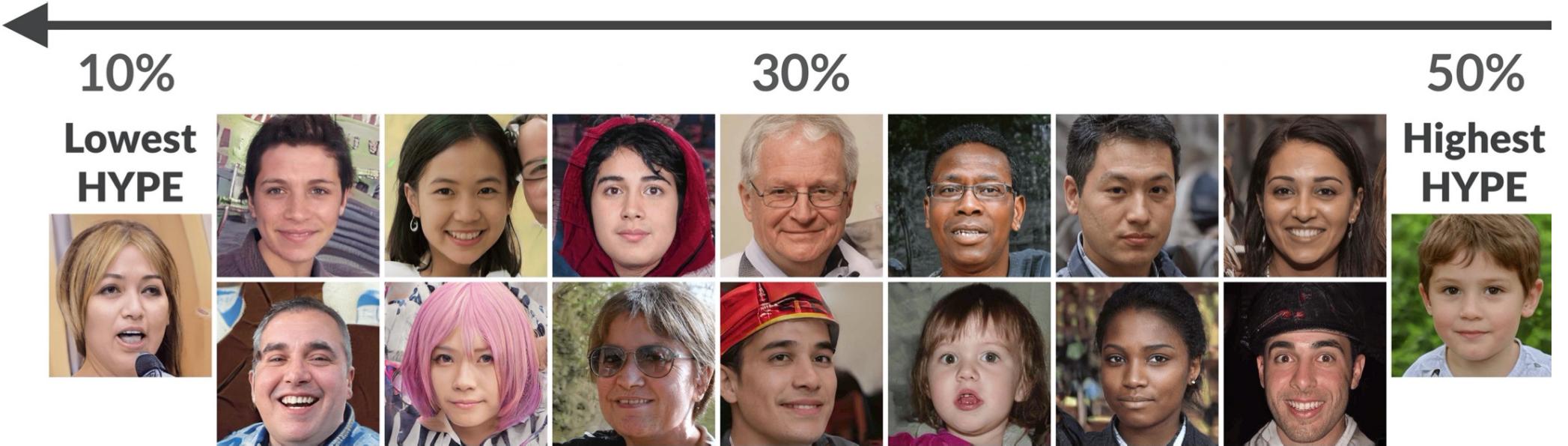


Figure 2: Example images sampled with the truncation trick from StyleGAN trained on F Images on the right exhibit the highest  $HYPE_\infty$  scores, the highest human perceptual fidelity.

*“HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models”, NeurIPS 2019*

## Human Evaluation

- The most direct answer to the question of whether generated data is “realistic-looking”
  - Expensive, time consuming, and maybe not reproducible
  - But perhaps the only way to claim “groundtruth”

# Inception Score (IS)

- **Assumption 1:** We are evaluating sample quality for generative models trained on labelled datasets
- **Assumption 2:** We have a good probabilistic classifier  $c(y|x)$  for predicting the label  $y$  for any point  $x$
- We want samples from a good generative model to satisfy two criteria: sharpness and diversity
- **Sharpness (S)**



$$S = \exp \left( E_{x \sim p} \left[ \int c(y|x) \log c(y|x) dy \right] \right)$$

- High sharpness implies classifier is confident in making predictions for generated images
- That is, classifier's predictive distribution  $c(y|x)$  has low entropy

# Inception Score (IS)

- Diversity (D)



$$D = \exp \left( -E_{\mathbf{x} \sim p} \left[ \int c(y|\mathbf{x}) \log c(y) dy \right] \right)$$

where  $c(y) = E_{\mathbf{x} \sim p}[c(y|\mathbf{x})]$  is the classifier's marginal predictive distribution

- High diversity implies  $c(y)$  has high entropy
- Inception scores (IS) combine the two criteria of sharpness and diversity into a simple metric

$$IS = D \times S$$

- Higher IS corresponds to better quality.
- If classifier is not available, a classifier trained on a large dataset, e.g., Inception Net trained on the ImageNet dataset

# Frechet Inception Distance (FID)

- Inception Scores only require samples from  $p_\theta$  and do not take into account the desired data distribution  $p_{\text{data}}$  directly (only implicitly via a classifier)
- **Frechet Inception Distance (FID)** measures similarities in the feature representations (e.g., those learned by a pretrained classifier) for datapoints sampled from  $p_\theta$  and the test dataset
- Computing FID:
  - Let  $\mathcal{G}$  denote the generated samples and  $\mathcal{T}$  denote the test dataset
  - Compute feature representations  $F_{\mathcal{G}}$  and  $F_{\mathcal{T}}$  for  $\mathcal{G}$  and  $\mathcal{T}$  respectively (e.g., prefinal layer of Inception Net)
  - Fit a multivariate Gaussian to each of  $F_{\mathcal{G}}$  and  $F_{\mathcal{T}}$ . Let  $(\mu_{\mathcal{G}}, \Sigma_{\mathcal{G}})$  and  $(\mu_{\mathcal{T}}, \Sigma_{\mathcal{T}})$  denote the mean and covariances of the two Gaussians
  - FID is defined as the Wasserstein-2 distance between these two Gaussians:

$$\text{FID} = \|\mu_{\mathcal{T}} - \mu_{\mathcal{G}}\|^2 + \text{Tr}(\Sigma_{\mathcal{T}} + \Sigma_{\mathcal{G}} - 2(\Sigma_{\mathcal{T}}\Sigma_{\mathcal{G}})^{1/2})$$

- Lower FID implies better sample quality

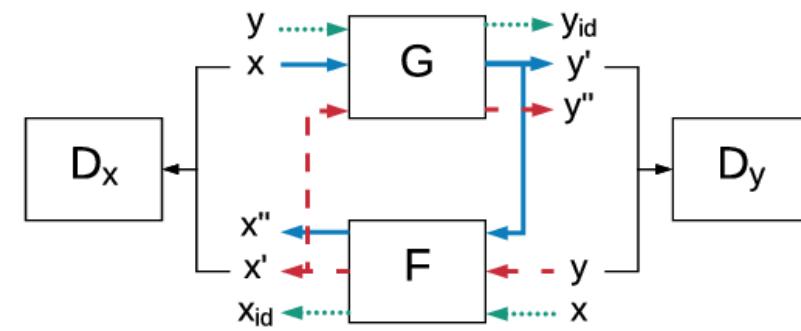
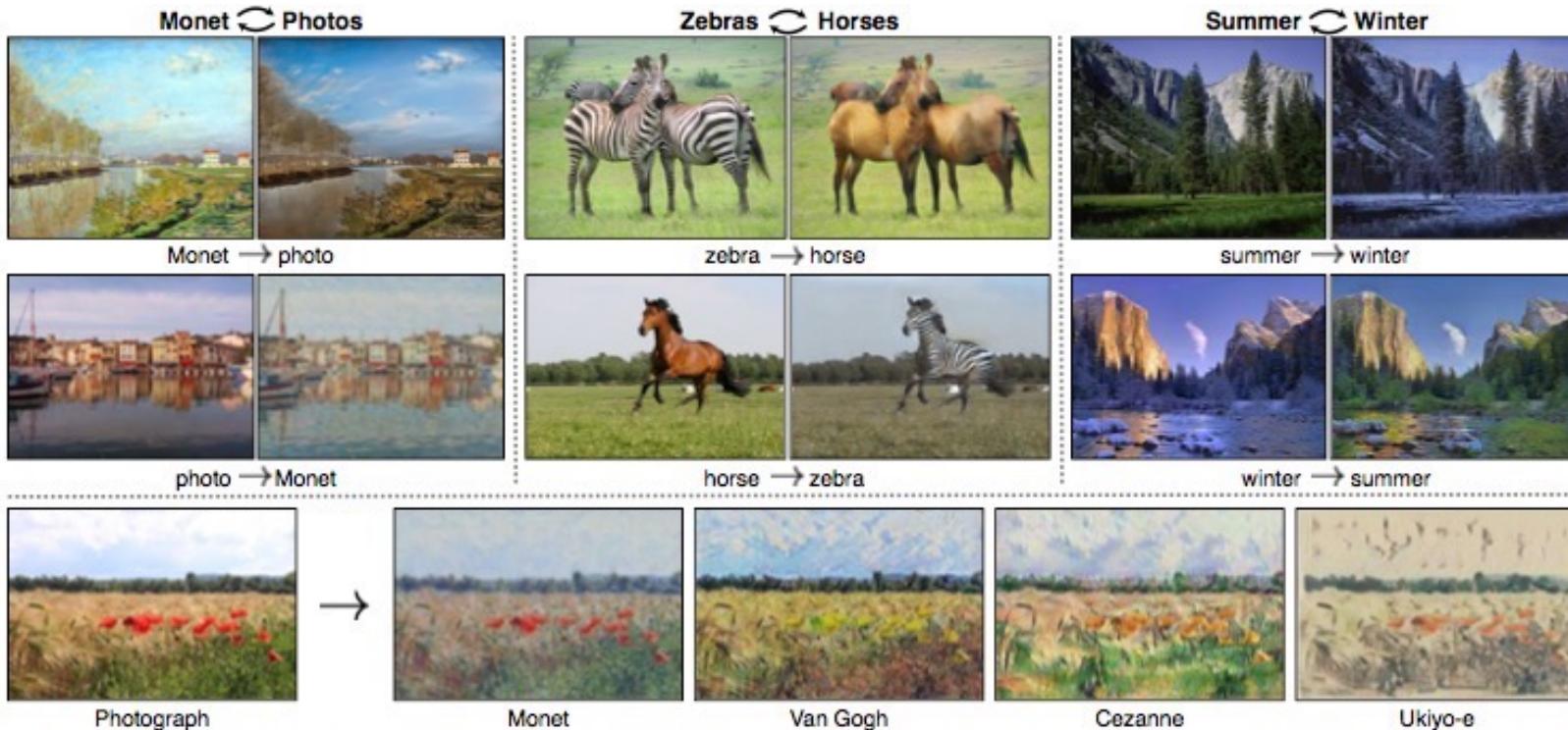
# Sampling-based proxy methods

- **More Fine-Grained Metrics:**
  - Birthday paradox test (detecting severe mode drop)
  - Classification-based metrics (quantifying **inter-class** mode dropping)
  - Black-box diagnosis (detecting **intra-class** mode collapse)
- **Many Aspects remain untouched yet:**
  - Novelty of generated images?
    - Are GANs really “memorizing” or “creating”, and to what extent?
  - Subtle co-variate shifts
    - Privacy, fairness, etc.
- **Comprehensive review:**
  - [https://deepgenerativemodels.github.io/assets/slides/cs236\\_lecture15.pdf](https://deepgenerativemodels.github.io/assets/slides/cs236_lecture15.pdf)

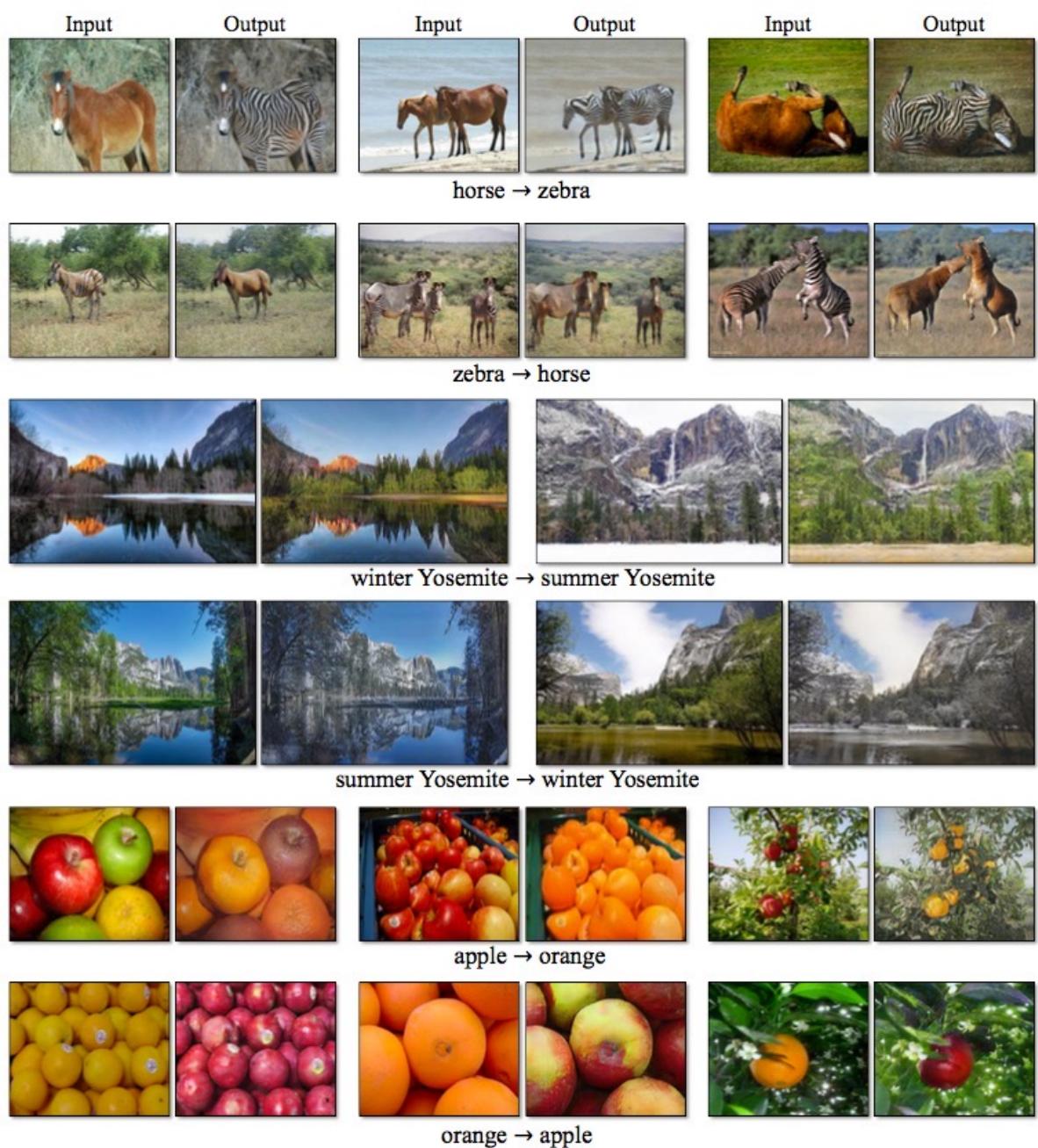


# An Application Tour of GANs

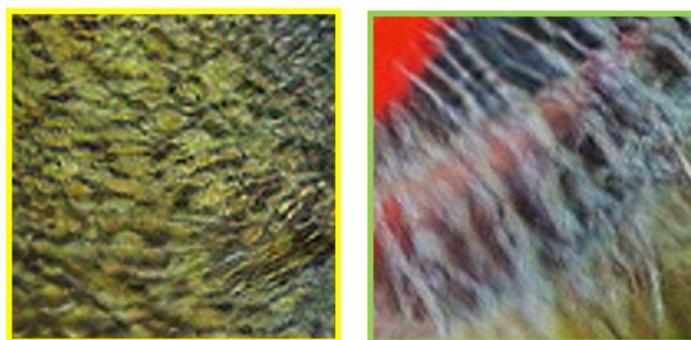
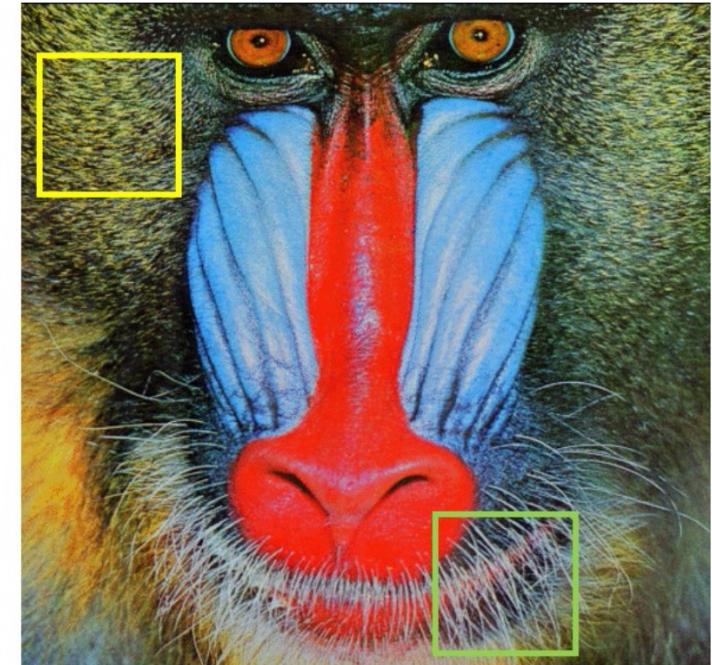
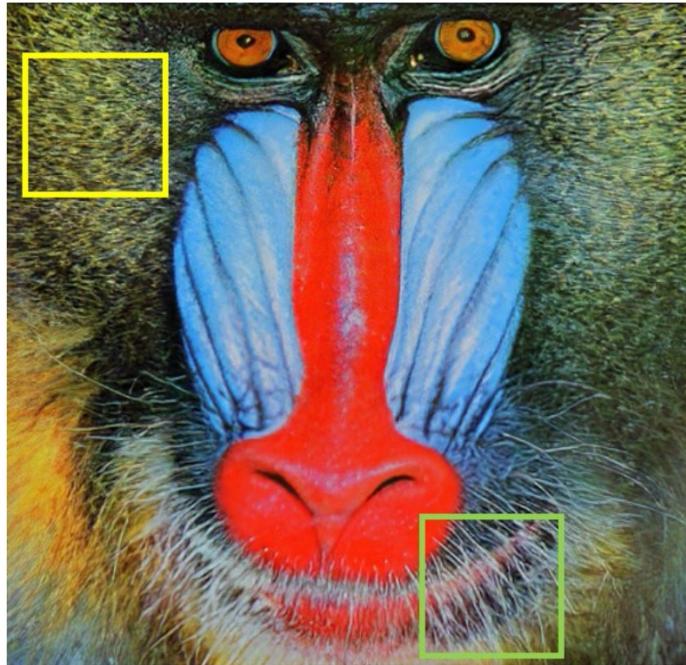
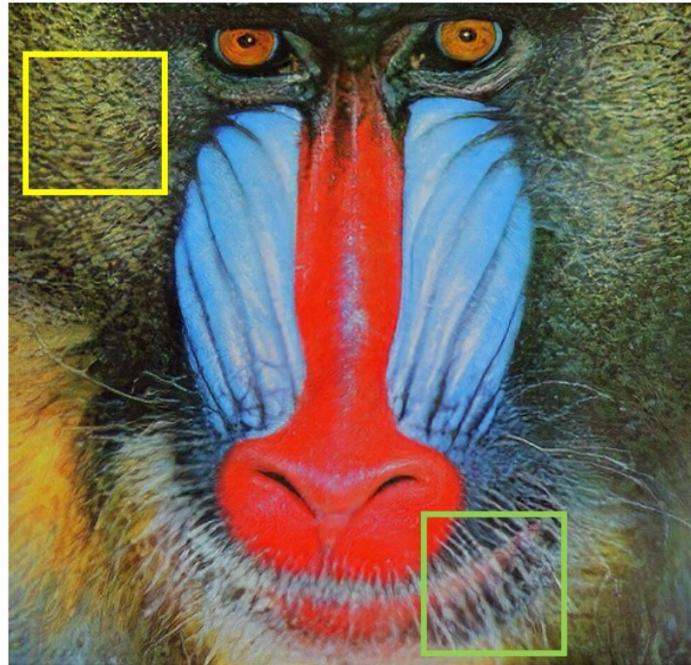
# CycleGAN: Unpaired Image Translation



Given two image collections, CycleGAN learns to translate an image from one collection to the other, without requiring correspondence between images



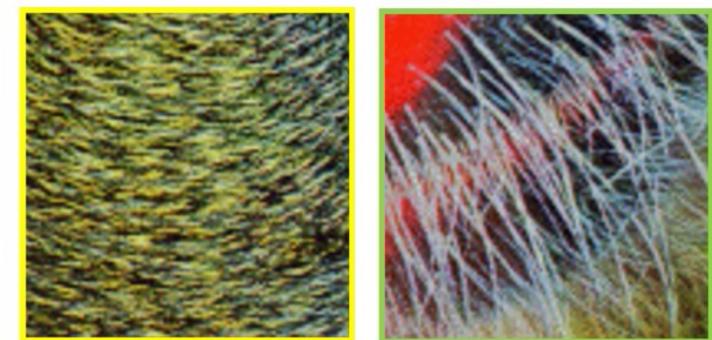
# GAN for Super Resolution: SRGAN/ESRGAN



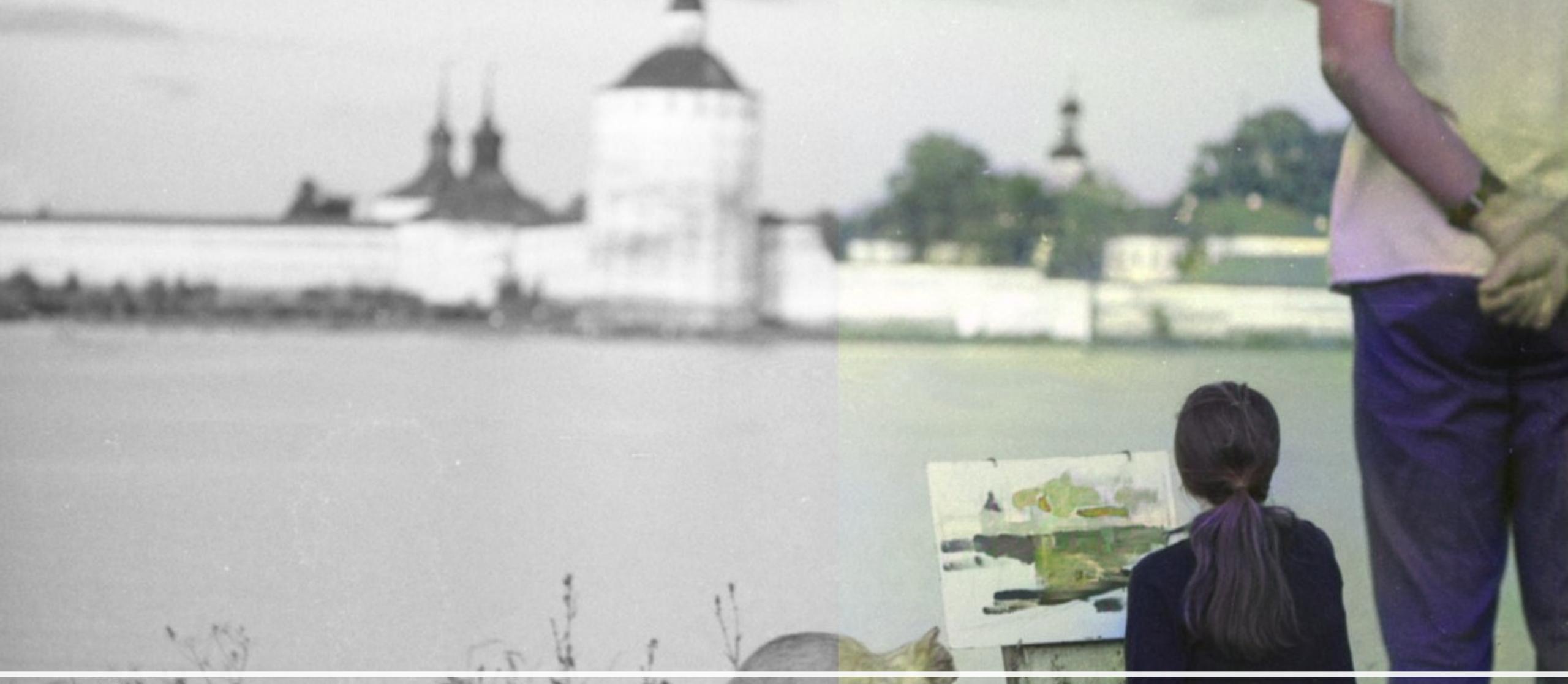
SRGAN



ESRGAN



Ground Truth



## DeOldify: GAN based Image Colorization

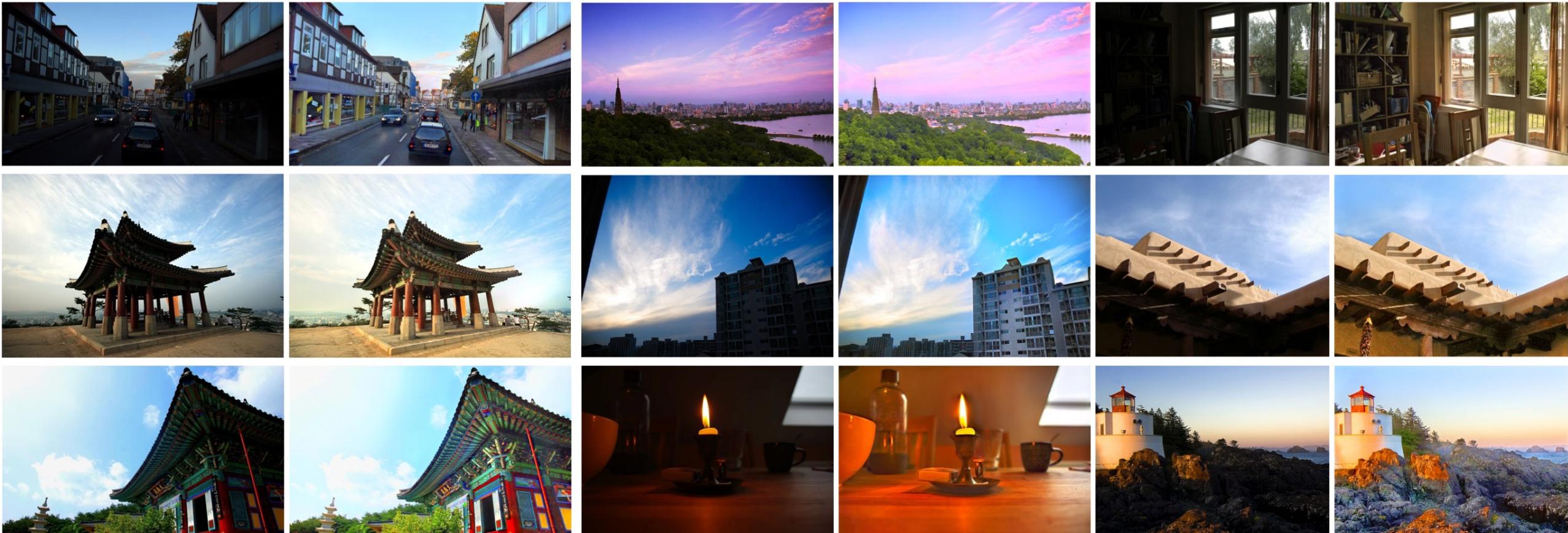
Original

Photo: Vladimir Perelman ©

DeOldify

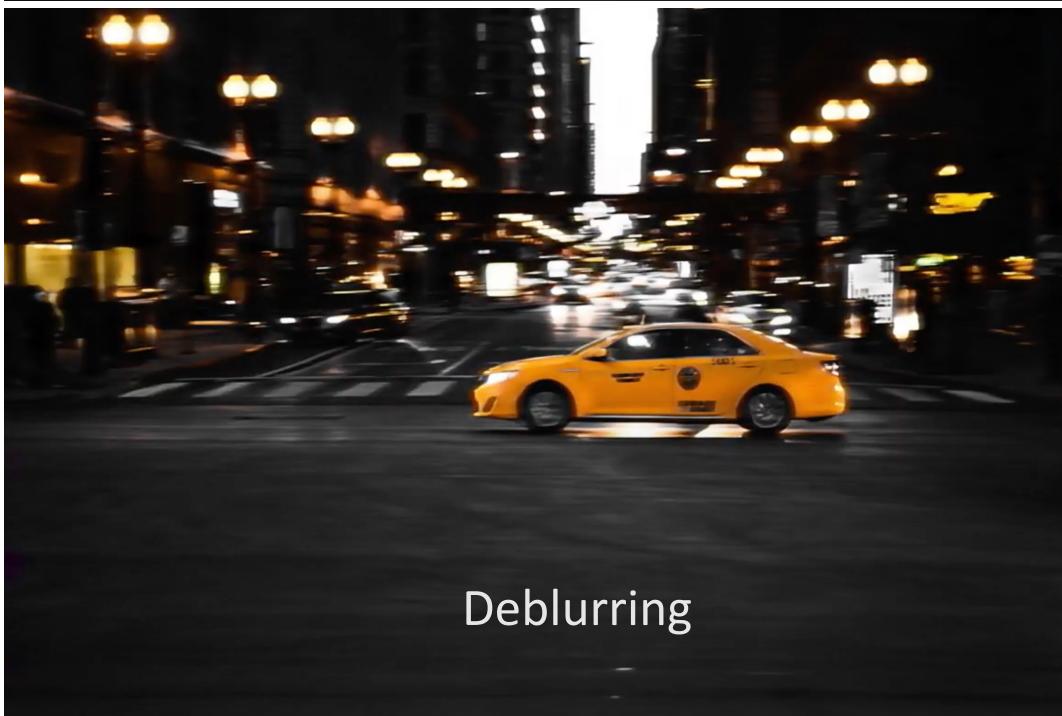
VITA

# Image Enhancement: EnlightenGAN



Now a standard plugin of Python GIMP toolbox, etc.

# VITA



Deblurring

Underwater Enhancement



# VITA

# Font Style Transfer and Animation



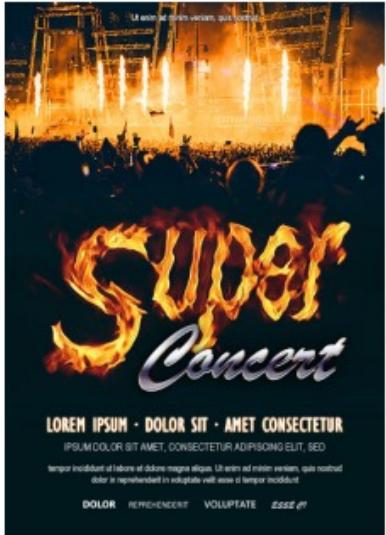
(a) source image



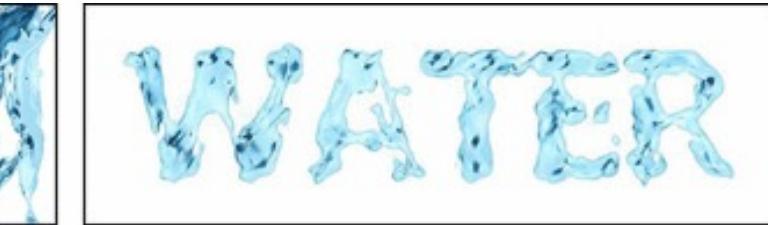
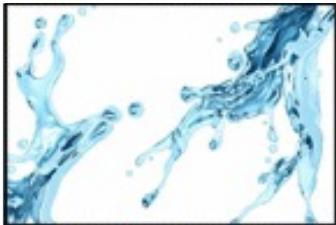
(b) adjustable stylistic degree of glyph



(c) stylized text



(d) application



(e) liquid artistic text rendering



(f) smoke artistic text rendering

# Interactive Image Editing using Sketching

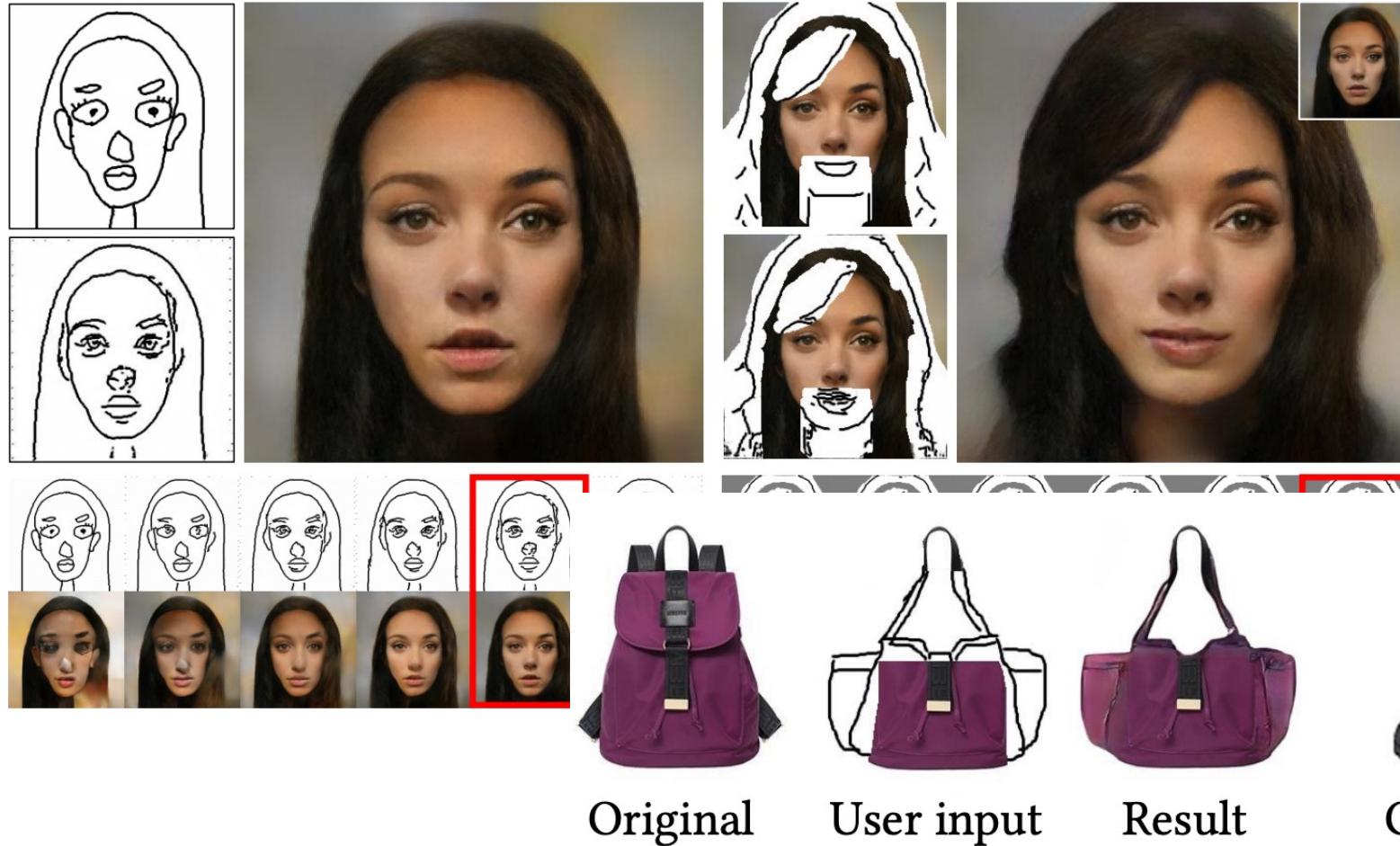
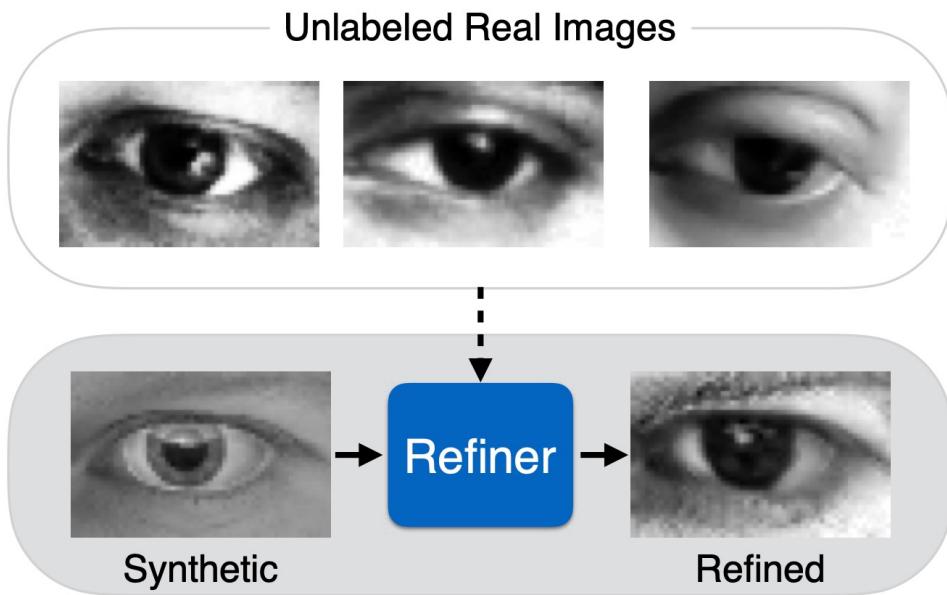


Figure 14. Performance on handbag and shoe datasets.

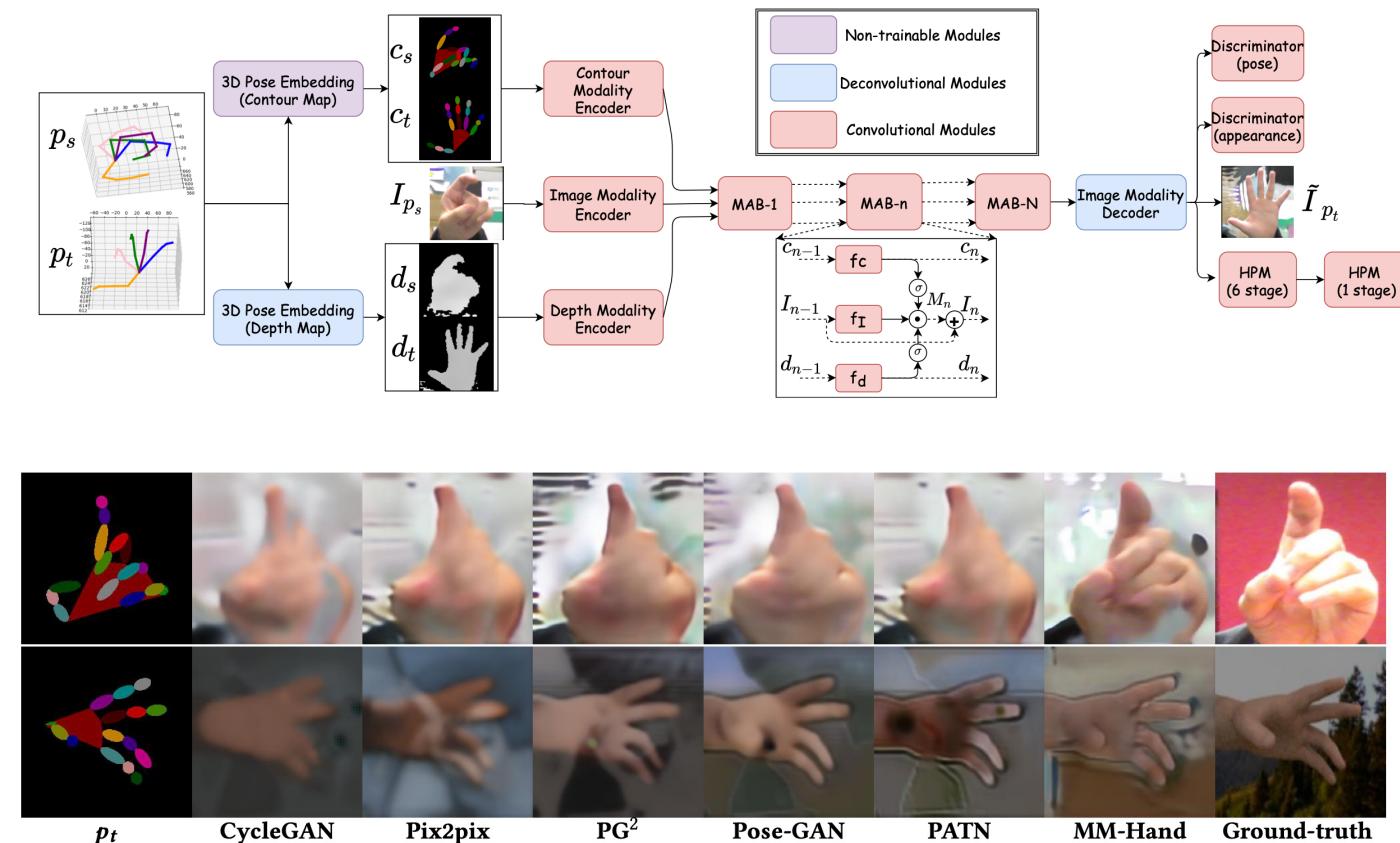
# Synthetic Data Generation

ACM MM 2020, “MM-Hand: 3D-Aware Multi-Modal Guided Hand Generative Network for 3D Hand Pose Synthesis”

VITA



CVPR 2017 Best Paper, “Learning from Simulated and Unsupervised Images through Adversarial Training”



# "DeepFake"

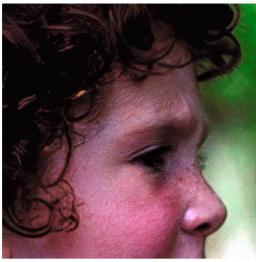
- A person in an existing image or video is replaced with someone else's likeness, usually by GAN (sometimes autoencoders)



<https://github.com/deepfakes/faceswap>

# GAN Compression for Mobile APPs

Original ESRGAN  
1176.61 GFLOPs  
66.8 MB



Pruned ESRGAN  
113.07 GFLOPs  
6.40 MB



SRGAN  
166.66 GFLOPs  
6.08 MB



VDSR  
699.36 GFLOPs  
2.67 MB

Source image

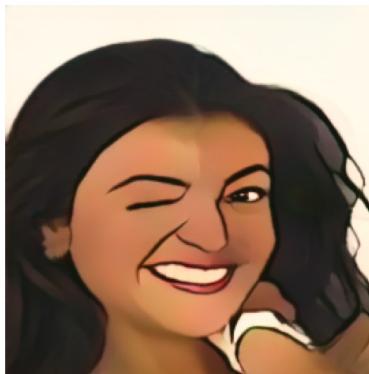
AGD (Proposed)  
108.06 GFLOPs  
1.64 MB

Original result  
56.46 GFLOPs  
42.34 MB

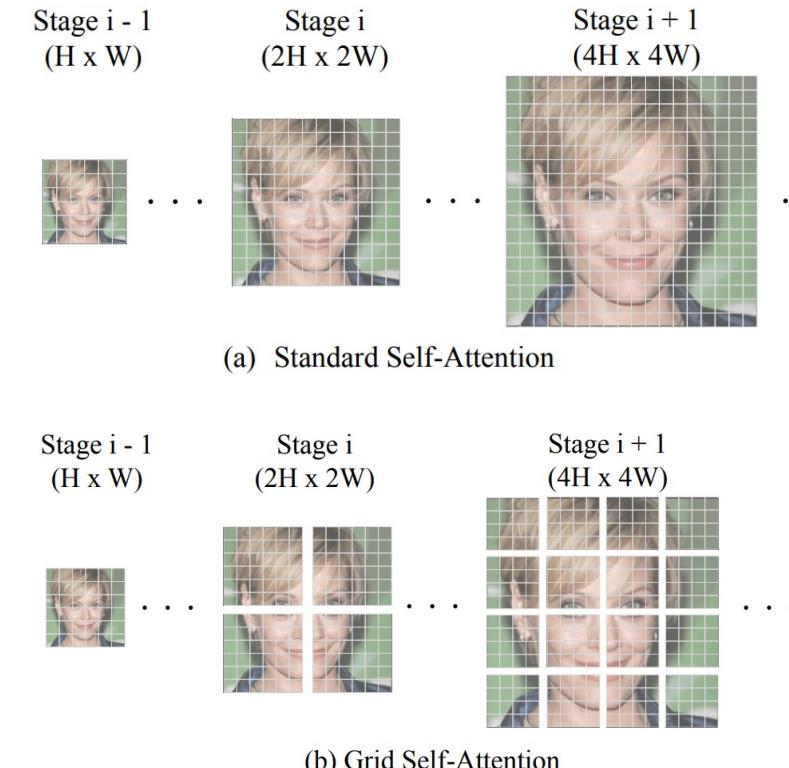
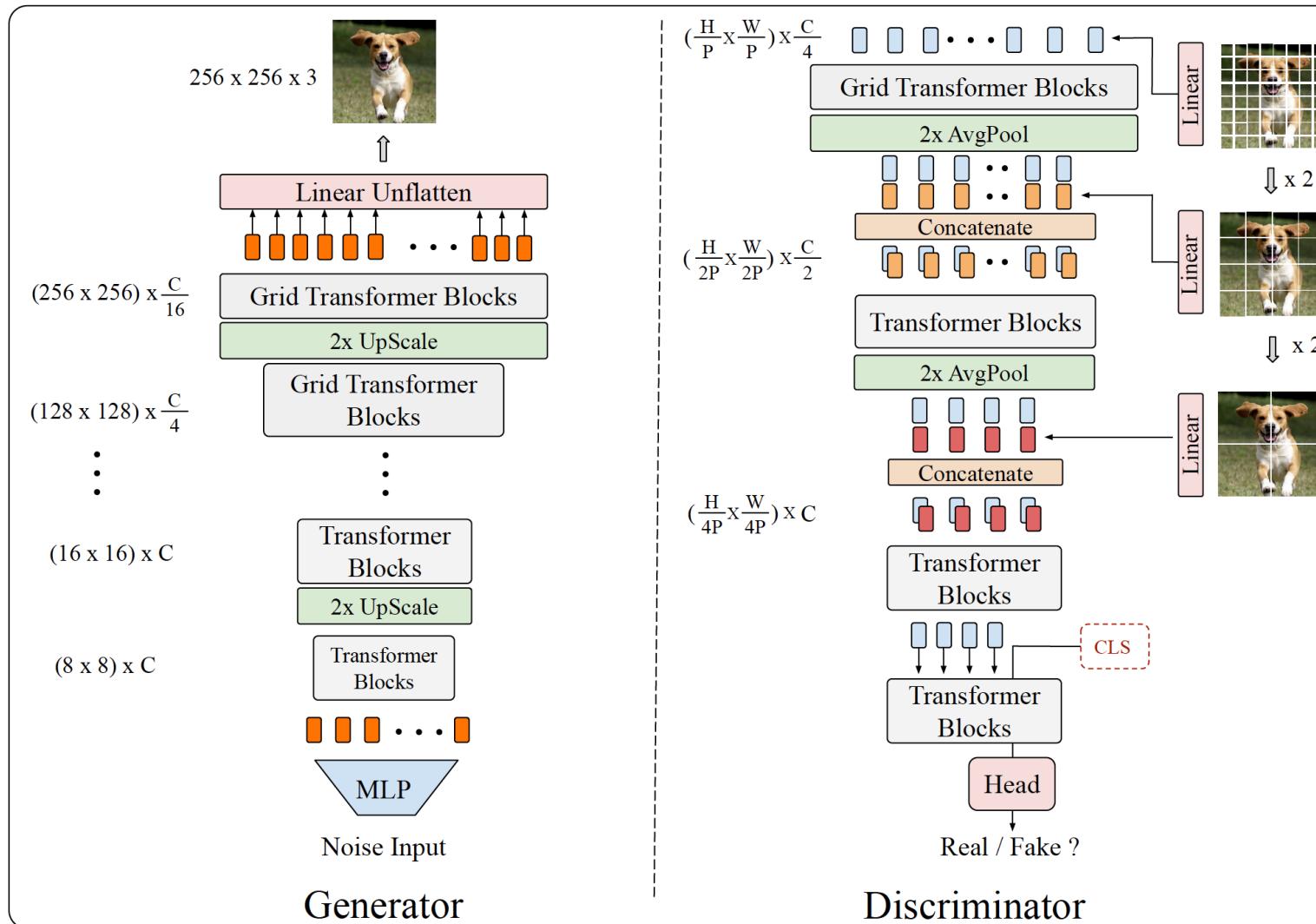
Our result (32bit)  
1.34 GFLOPs  
0.80 MB



Our result (8bit)  
1.20 GFLOPs  
0.18 MB

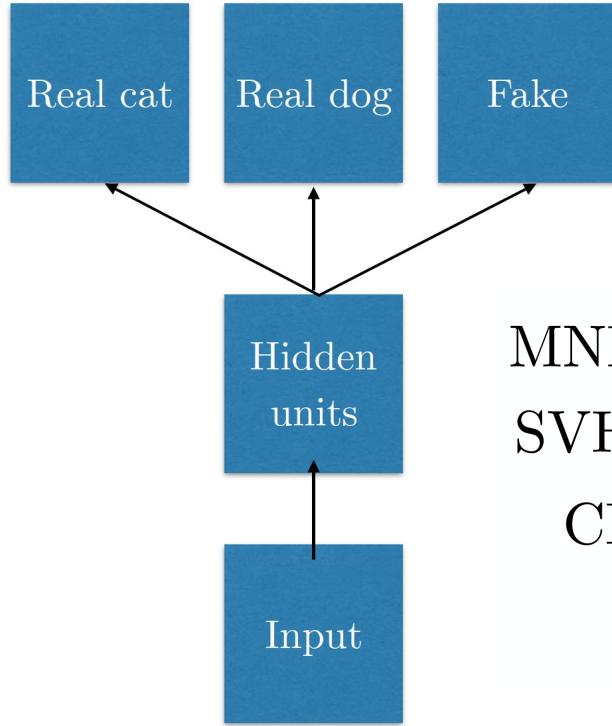
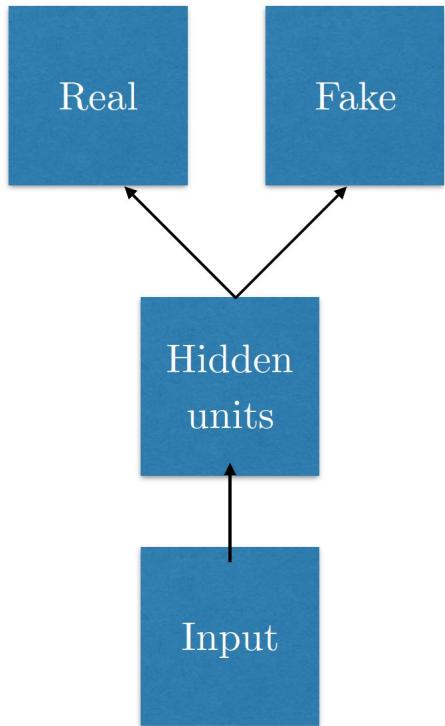


# TransGAN: Two Transformers Make One Strong GAN



VITA

# Semi-supervised Learning



MNIST: 100 training labels  $\rightarrow$  80 test mistakes

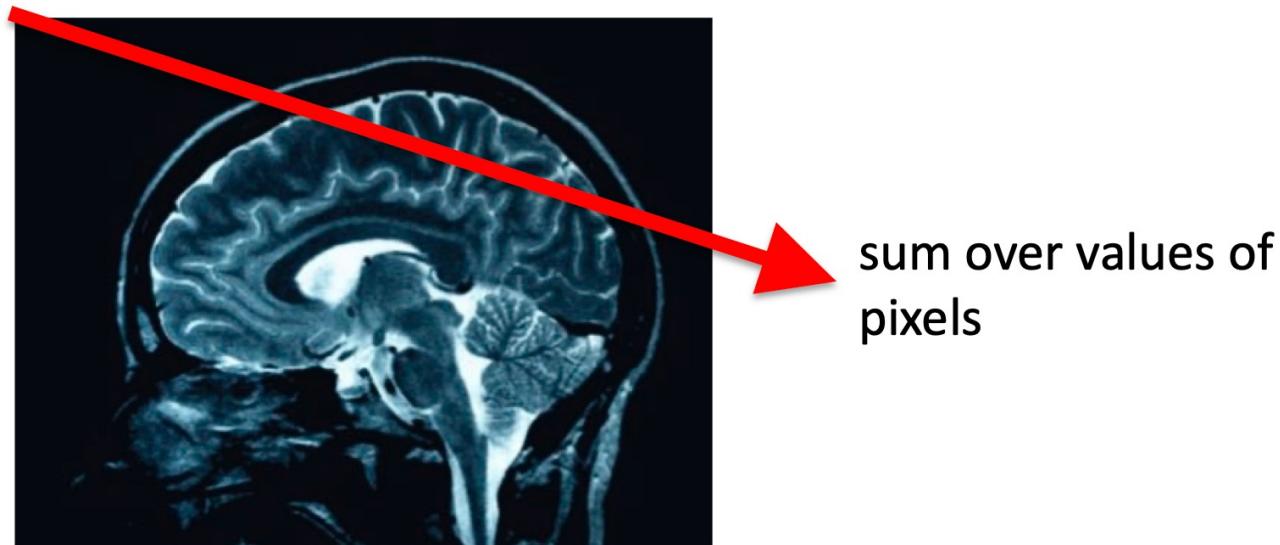
SVHN: 1,000 training labels  $\rightarrow$  4.3% test error

CIFAR-10: 4,000 labels  $\rightarrow$  14.4% test error

(Dai et al 2017)

# GAN can be the “new” image prior too!

- Q1: When do you want to recover some unknown vector by observing linear measurements on its entries?



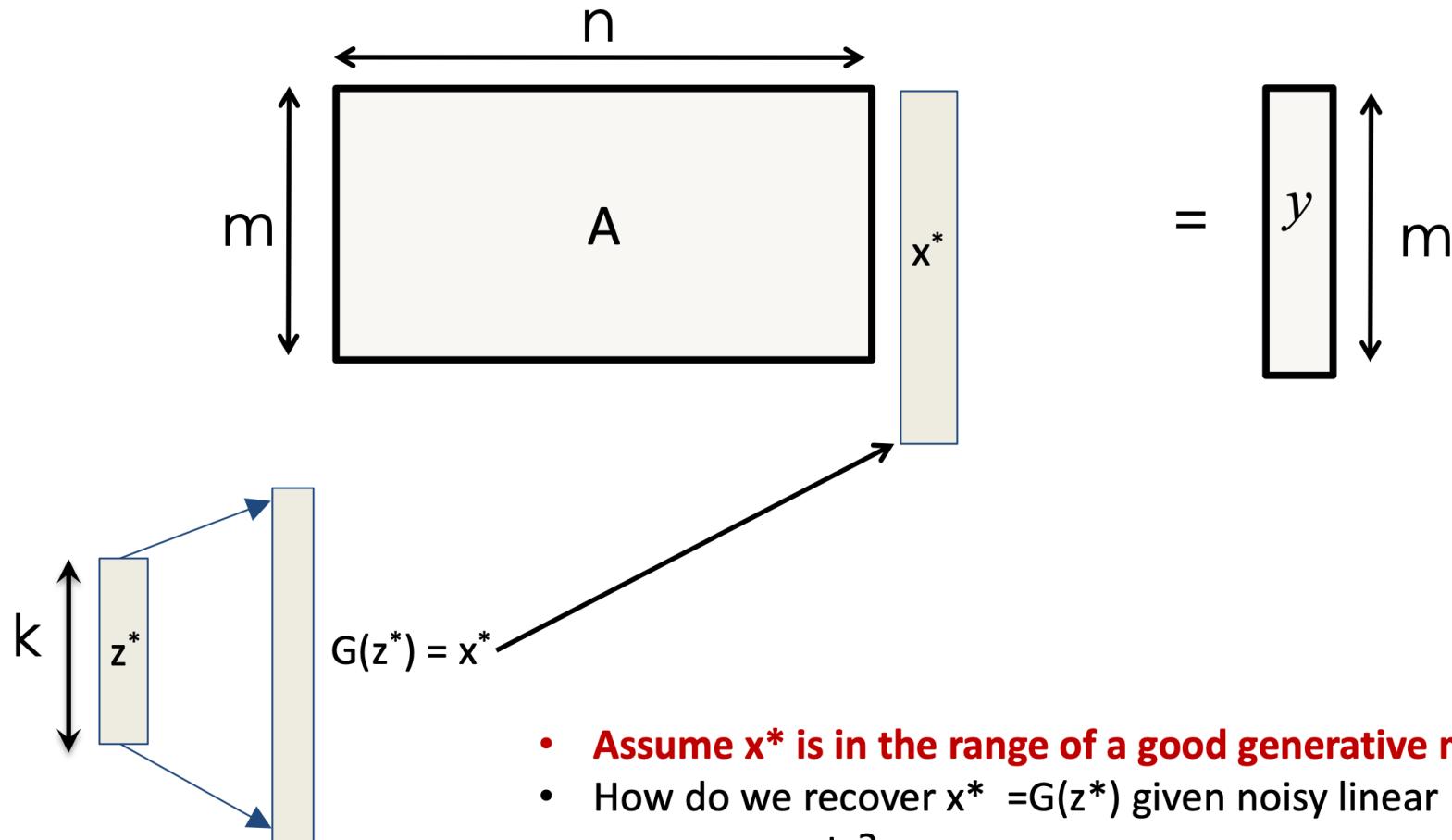
- Real images are not sparse (except night-time sky).
- But they can be sparse in a known basis , i.e.  $x'' = D x^*$
- $D$  can be DCT or Wavelet basis.

Slides credits: Alex Dimakis

# GAN can be the “new” image prior too!

- Q1: What is the “new” image prior?
- 1. Sparsity in a basis is a decent model for natural images (jpg, mpeg, mp3, based on that)
- 1. But now we have much better **data driven** models for natural images: VAEs and GANs
- 2. Idea: Take sparsity out of compressed sensing. Replace with GAN
- 3. Ok. But how to do that?
- Requirements:
  - Reconstruction
  - But what about loss?
  - D can be replaced by G

# GAN can be the “new” image prior too!



- Assume  $x^*$  is in the range of a good generative model  $G(z)$ .
- How do we recover  $x^* = G(z^*)$  given noisy linear measurements?
- $y = A x^* + \eta$

Slides credits: Alex Dimakis

# Few-Shot Generation



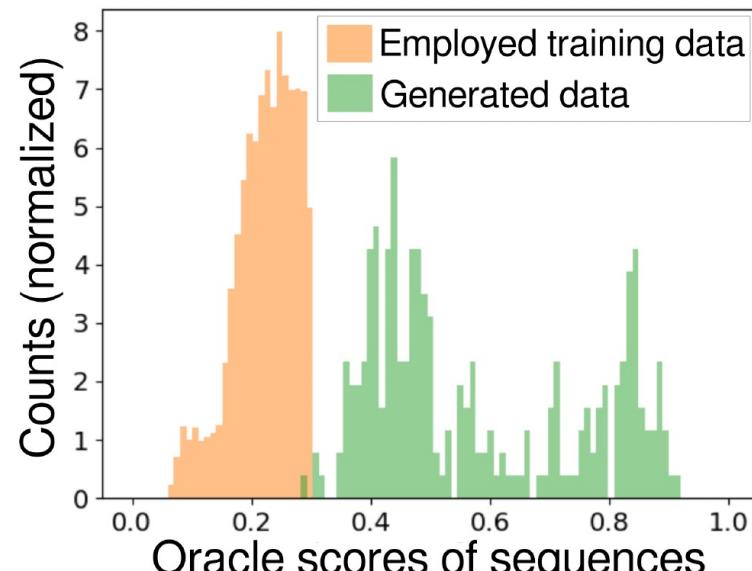
**Table 4. Few-shot generation.** Following the setting in Zhao et al. (2020a), we calculate the FID with 5K samples and the training dataset is adopted as the reference distribution. All transfer learning methods have their pre-trainings from the FFHQ face dataset (Karras et al., 2019), and the StyleGAN2 tickets at 48.80% sparsity level are found and used with all the few-shot datasets.

Methods	Pre-training?	100-shot by (Zhao et al., 2020a)		AnimalFace	
		a	Grumpy Cat	Panda	Cat
;	;	34.20	21.38	54.83	83.04
0	;	287.96	331.86	279.48	254.08
;	;	34.06	23.20	52.61	82.38
;	;	31.22	17.95	47.70	70.46
;	;	61.97	90.96	95.75	164.54
;	;	35.46	15.94	54.02	72.47
;	;	56.81	82.45	85.92	153.90
;	;	31.02	14.75	47.40	68.28

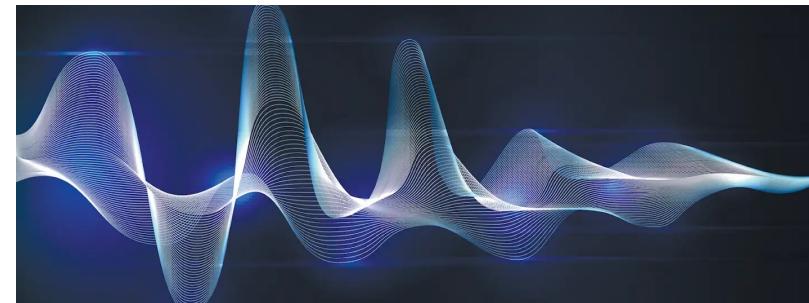
**Figure A10.** Few-shot generalization results of StyleGAN2 tickets (48.80%) with AdvAug only on 100-shot Obama, Grumpy Cat, Panda, and AnimalFace datasets, respectively.

# Emerging non-image applications

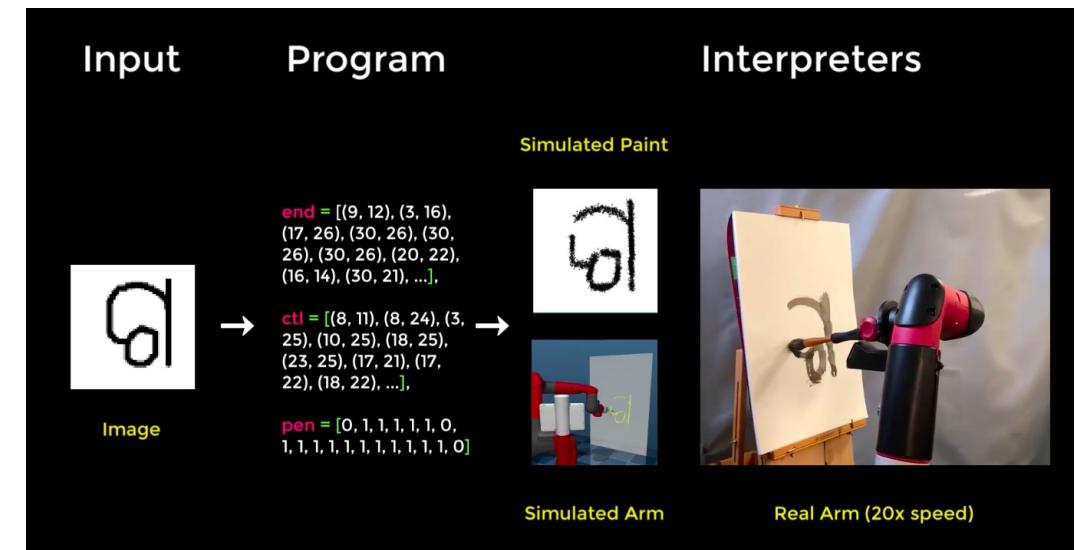
Designing DNA to optimize protein binding



(Killoran et al, 2017)



Text-to-Speech (TTS)



Program Synthesis and RL

# Summary & Take-Home Messages

- **Good:** GANs can produce awesome, crisp results for many problems
- **Bad:** GANs have stability issues and many open theoretical questions
- **Ugly:** Many ad-hoc tricks and modifications to get GANs to work correctly

*The Rising Field Welcomes YOU!*





The University of Texas at Austin  
**Electrical and Computer  
Engineering**  
*Cockrell School of Engineering*