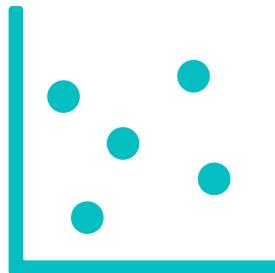


Lecture Notes for
Neural Networks
and Machine Learning



Holy GAN-Zooks



Logistics and Agenda

- Logistics
 - None!
- Agenda
 - Student Paper Presentation
 - GAN Town Hall
 - Lightning Round for GANs:
 - ◆ CycleGAN
 - ◆ Using GANs for Boosting Classifiers
 - ◆ Text-to-image Synthesis

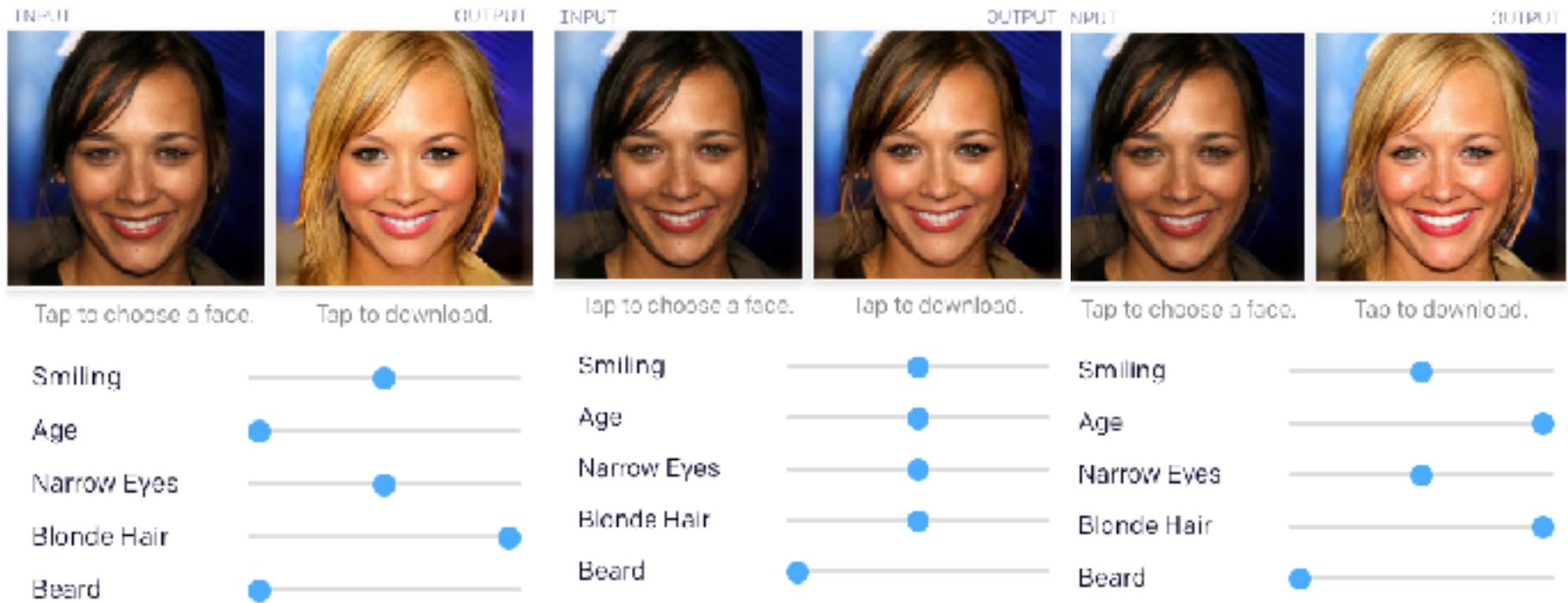


GAN Town Hall

Asking a friend about adversarial attacks



Student Presentation: GLOW



Last Time

$$\frac{1}{m} \sum_{i=1}^m f(g(z^{(i)}))$$

Maximize g ,
freeze weights of critic
No change

- Theorem: a function is 1-Lipschitz if and only if its gradient has a norm less than or equal to one.

Maximize f
freeze generator weights,
incentivize critic gradient norm to be one

Choose large lambda
like lambda = 10

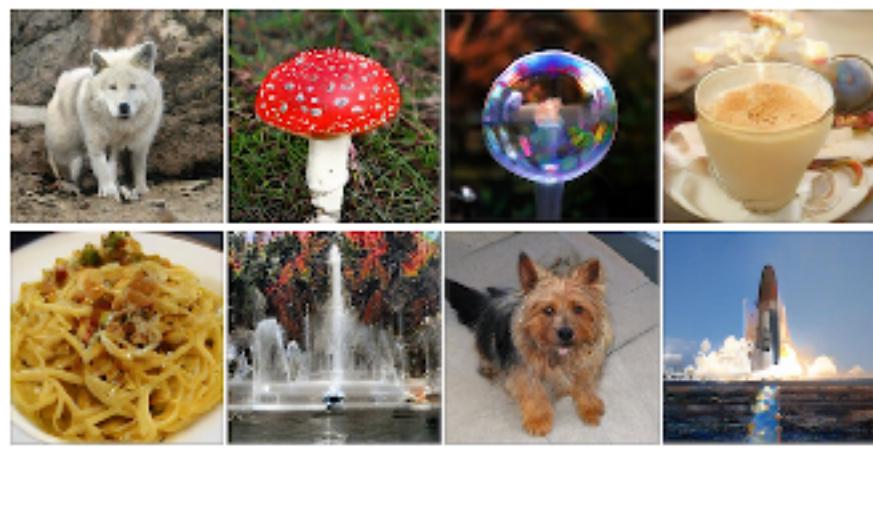
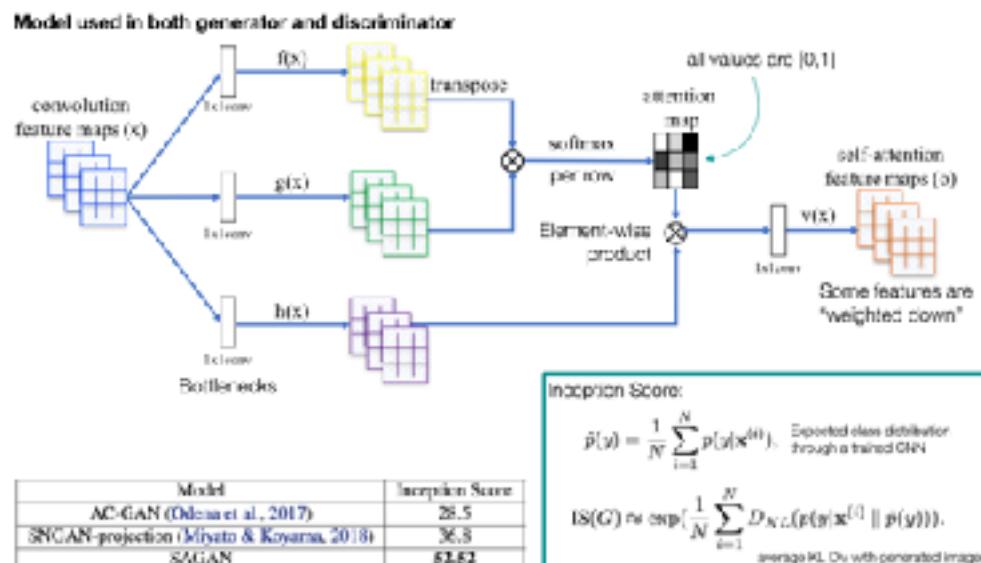
$$\frac{1}{m} \sum_{i=1}^m f(x_{real}^{(i)}) - f(g(z^{(i)})) + \lambda \frac{1}{W} \sum_{i=1}^W (\|\nabla f(\hat{x}^{(i)})\|_2 - 1)^2$$

where for ϵ in $U[0,1]$ $\hat{x}^{(i)} = \epsilon \cdot x_{real}^{(i)} + (1 - \epsilon) \cdot g(z^{(i)})$
randomly mix together real and fake images

- Update discriminator twice as often as generator
- Large batch size: 1024 or 2048
- Use skip connections in model architecture, starting from z
- Use LOTS of filters: 150% more filters than related work
- Truncation trick:** during training, use wider sampling than during evaluation

- Use hinge loss: $\frac{1}{m} \sum_{i=1}^m f(x_{real}^{(i)}) \cdot f(g(z^{(i)}))$

- Use moving average of weights: $W_k = \sum_i \gamma^i \frac{W_{k-i}}{\text{past}}$



Example GANs, Abridged

Cumulative Number of GAN Papers, by Year



ABC-GAN — [ABC-GAN: Adap](#)
ABC-GAN — [GANs for LIFE:](#)
AC-GAN — [Conditional Image](#)
acGAN — [Face Aging With C](#)
ACGAN — [Coverless Informa](#)
acGAN — [On-line Adaptative](#)
ACtuAL — [ACTual: Actor-Critic](#)
AdaGAN — [AdaGAN: Boostin](#)
Adaptive GAN — [Customizing](#)
AdvEntuRe — [AdvEntuRe: Ad](#)
AdvGAN — [Generating adver](#)
AE-GAN — [AE-GAN: adversa](#)
AE-OT — [Latent Space Optim](#)
AEGAN — [Learning Inverse M](#)
AF-DCGAN — [AF-DCGAN: AF](#)
AffGAN — [Amortised MAP Inf](#)
AIM — [Generating Informati](#)
AL-CGAN — [Learning to Gen](#)
ALI — [Adversarially Learned I](#)
AlignGAN — [AlignGAN: Lear](#)
AlphaGAN — [AlphaGAN: Ger](#)
AM-GAN — [Activation Maxim](#)
AmbientGAN — [AmbientGAN](#)
AMC-GAN — [Video Predicti](#)
AnoGAN — [Unsupervised And](#)
APD — [Adversarial Distillatio](#)
APE-GAN — [APE-GAN: Adv](#)
ARAE — [Adversarially Regulari](#)
ARDA — [Adversarial Representati](#)
ARIGAN — [ARIGAN: Synthetic](#)
ArtGAN — [ArtGAN: Artwork S](#)
ASDL-GAN — [Automatic Steg](#)
ATA-GAN — [Attention-Aware](#)
Attention-GAN — [Attention-G/](#)
AttGAN — [Arbitrary Facial Attr](#)
AttnGAN — [AttnGAN: Fine-Grained](#)
AVID — [AVID: Adversarial Vis](#)



CycleGAN

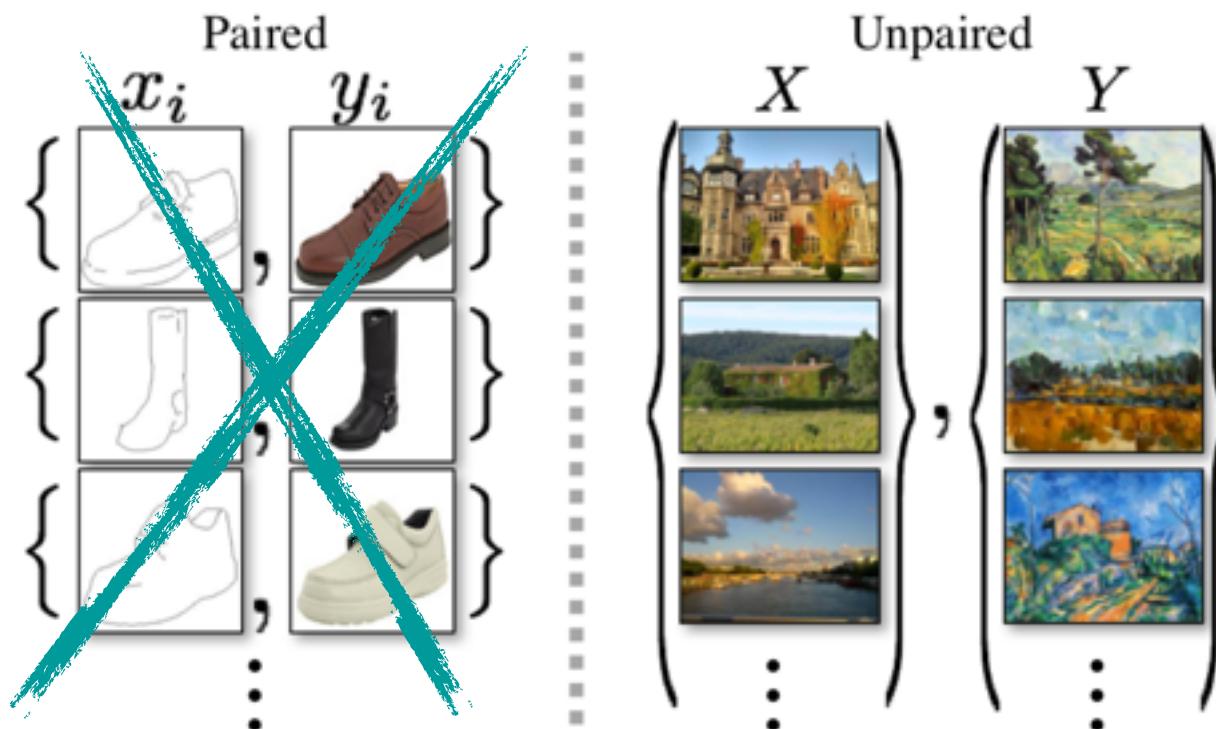
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

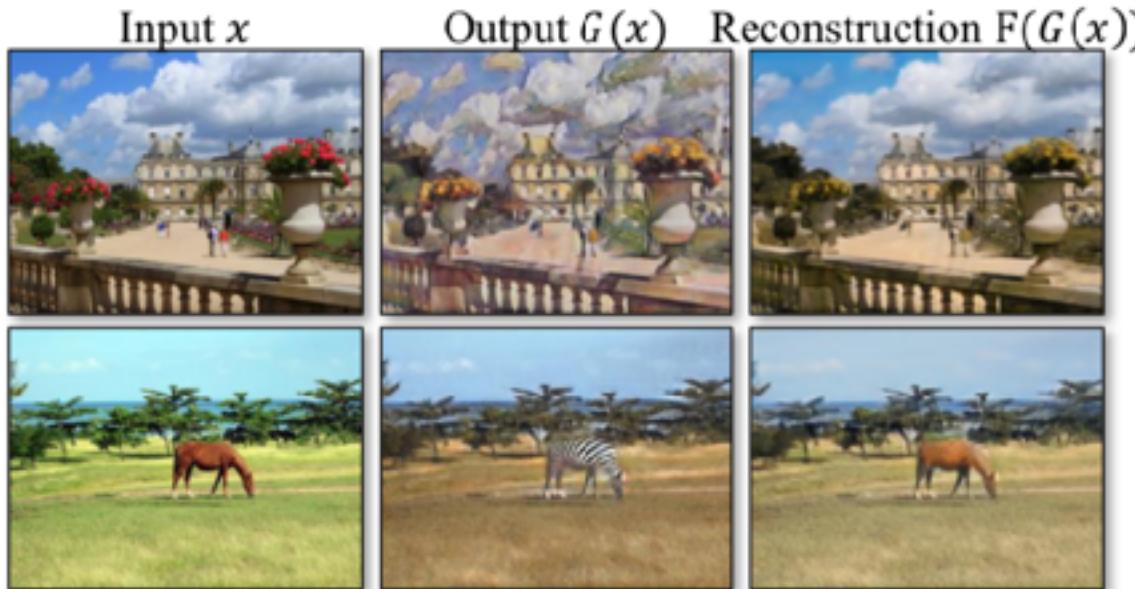
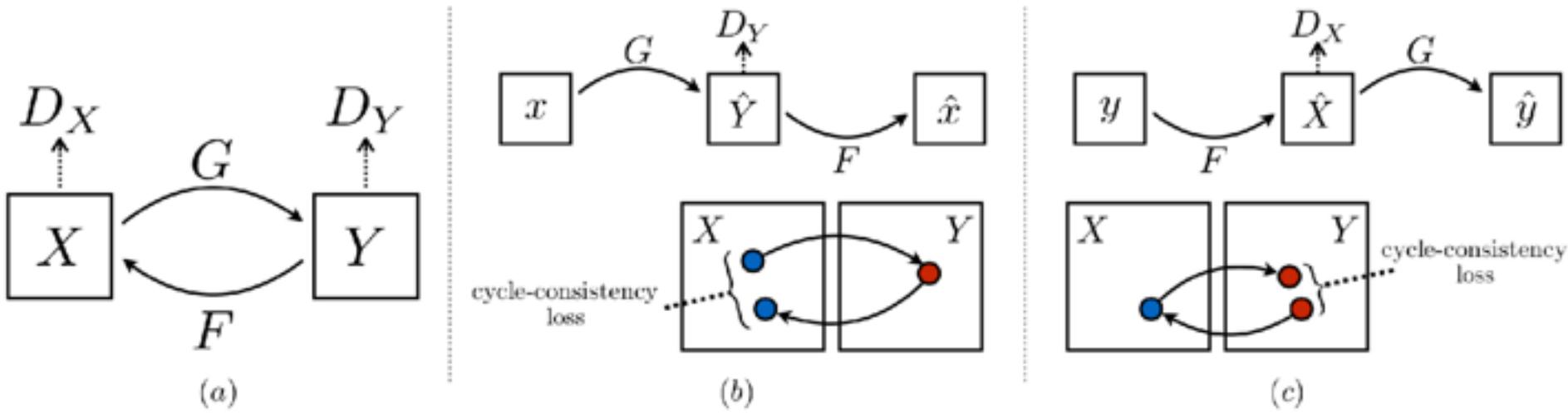


Unpaired Cycles

- Idea: Try to find specialized unpaired translations that are consistent between domains



Defining Cycle Consistency



Adding to the loss function

- Have two discriminator losses for two auto encoders:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = & \mathbb{E}_{y \sim p_{\text{data}}(y)} [\log D_Y(y)] \\ & + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))]\end{aligned}$$

- Enforce cycle consistency and self consistency:

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = & \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] \\ & + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1].\end{aligned}$$

$$\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(x) - x\|_1]$$

- Put it all together:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$



Training Notes

Training details We apply two techniques from recent works to stabilize our model training procedure. First, for \mathcal{L}_{GAN} (Equation 1), we replace the negative log likelihood objective by a least-squares loss [35]. This loss is more stable during training and generates higher quality results. In particular, for a GAN loss $\mathcal{L}_{\text{GAN}}(G, D, X, Y)$, we train the G to minimize $\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$ and train the D to minimize $\mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$.

Second, to reduce model oscillation [15], we follow generators using a history of generated images rather than the ones produced by the latest generators. We keep an image buffer that stores the 50 previously created images.

For all the experiments, we set $\lambda = 10$ in Equation 3. We use the Adam solver [26] with a batch size of 1. All networks were trained from scratch with a learning rate of 0.0002. We keep the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. Please see the appendix (Section 7) for more details about the datasets, architectures, and training procedures.

So, not using Entropy?

Yes! Experience Replay!

Linear learning rate decay.



Show me the Code!

Discriminators

```
# Build and compile the discriminators
self.d_A = self.build_discriminator()
self.d_B = self.build_discriminator()
self.d_A.compile(loss='mse',
    optimizer=optimizer,
    metrics=['accuracy'])
self.d_B.compile(loss='mse',
    optimizer=optimizer,
    metrics=['accuracy'])
```

Cycle Producers

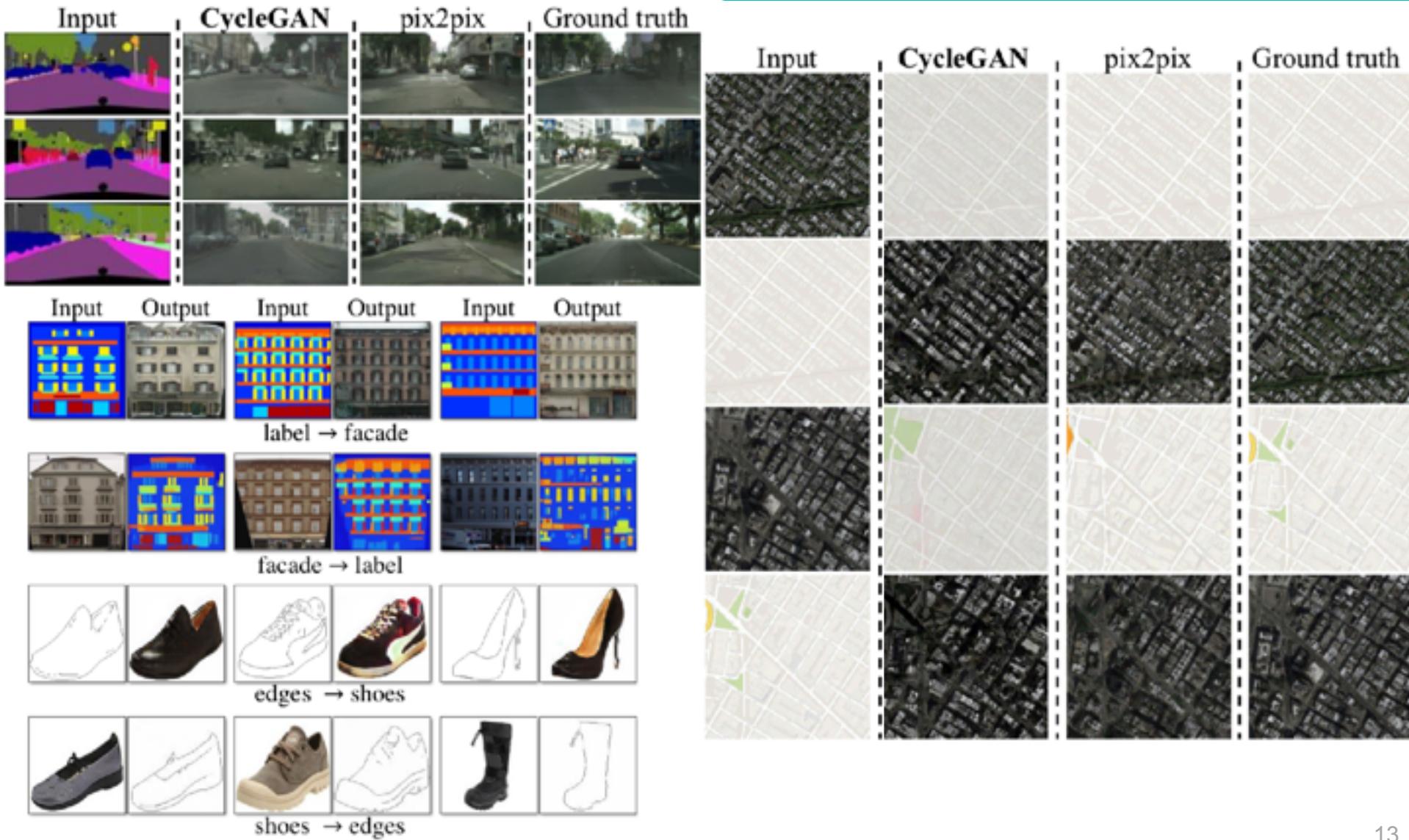
```
# Translate images to the other domain
fake_B = self.g_AB(img_A)
fake_A = self.g_BA(img_B)
# Translate images back to original domain
reconstr_A = self.g_BA(fake_B)
reconstr_B = self.g_AB(fake_A)
# Identity mapping of images
img_A_id = self.g_BA(img_A)
img_B_id = self.g_AB(img_B)
```

```
# Combined model trains generators to fool discriminators
self.combined = Model(inputs=[img_A, img_B],
                      outputs=[ valid_A, valid_B,
                                reconstr_A, reconstr_B,
                                img_A_id, img_B_id ])
self.combined.compile(loss=['mse', 'mse', discriminator MSE, 0,1
                           'mae', 'mae', cycle consistency
                           'mae', 'mae'], self consistency
                      loss_weights=[ 1, 1,
                                     self.lambda_cycle, self.lambda_cycle,
                                     self.lambda_id, self.lambda_id ],
                      optimizer=optimizer)
```

<https://github.com/eriklindernoren/Keras-GAN/blob/master/cyclegan/cyclegan.py>



Results



Fun Results



winter Yosemite → summer Yosemite



summer Yosemite → winter Yosemite



apple → orange

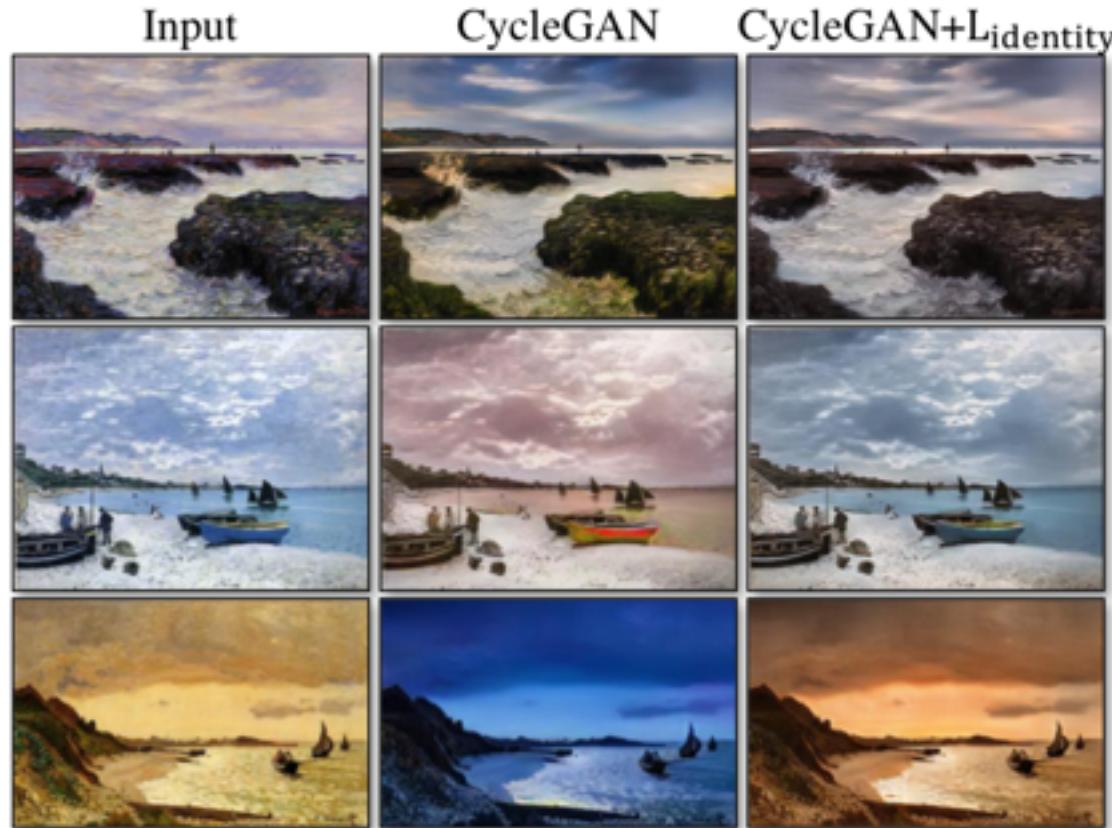


orange → apple



Results: Fixing Hue Changes

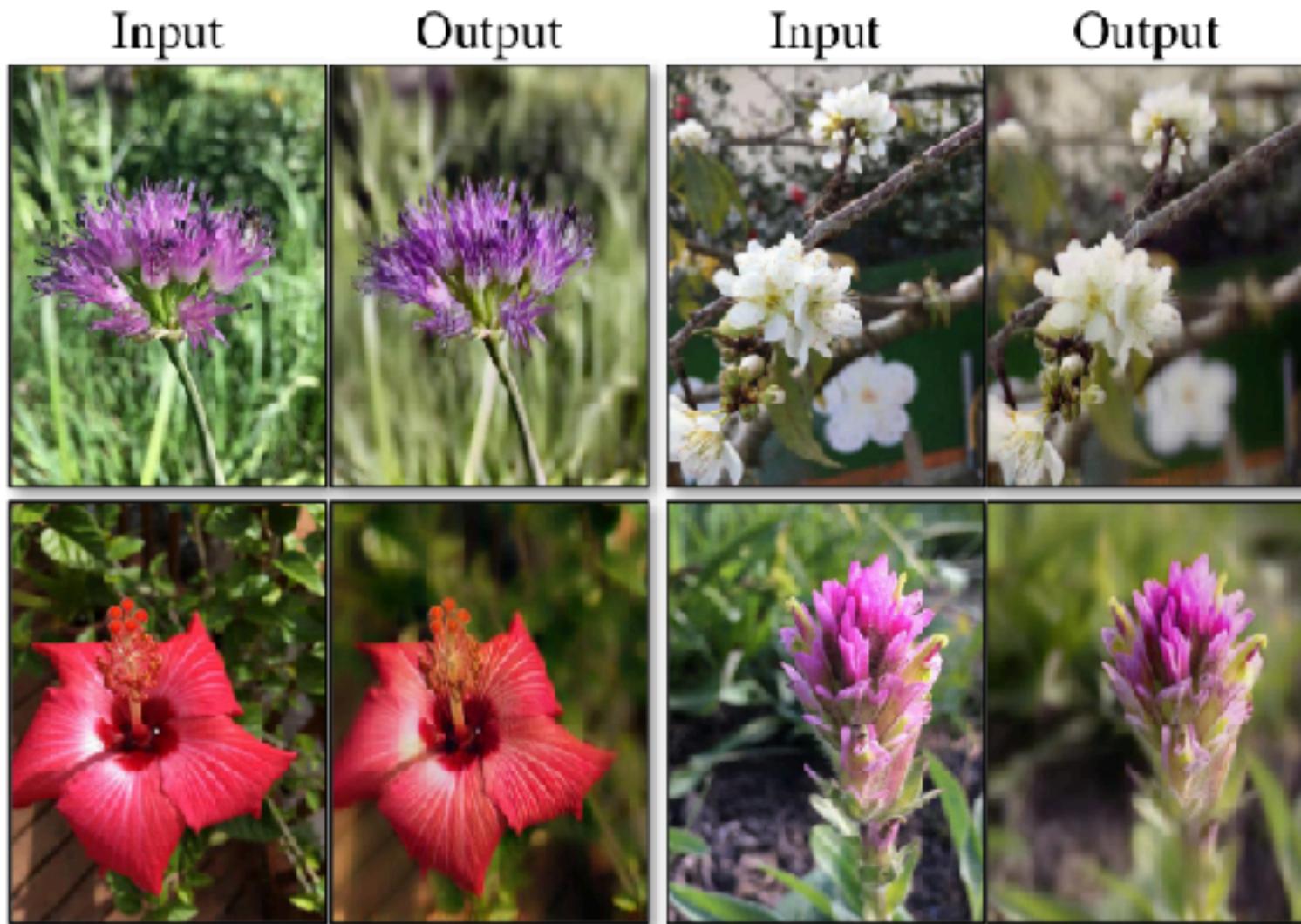
- Self consistency is mostly about preserving color



$$\begin{aligned}\mathcal{L}_{\text{identity}}(G, F) = \\ \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(y) - y\|_1] + \\ \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(x) - x\|_1]\end{aligned}$$



Image Enhancement



Map from iPhone Images to DSLR Photos



Full Circle: Back to Style Transfer

Input



Monet



Van Gogh



Cezanne



Ukiyo-e



They can't all be gold

Input



Output

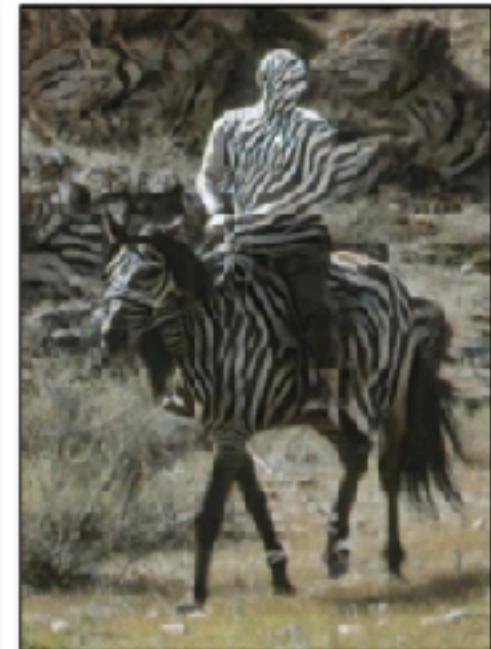


winter → summer

Input



Output



horse → zebra

Monet → photo



iPhone photo → DSLR photo



ImageNet “wild horse” training images



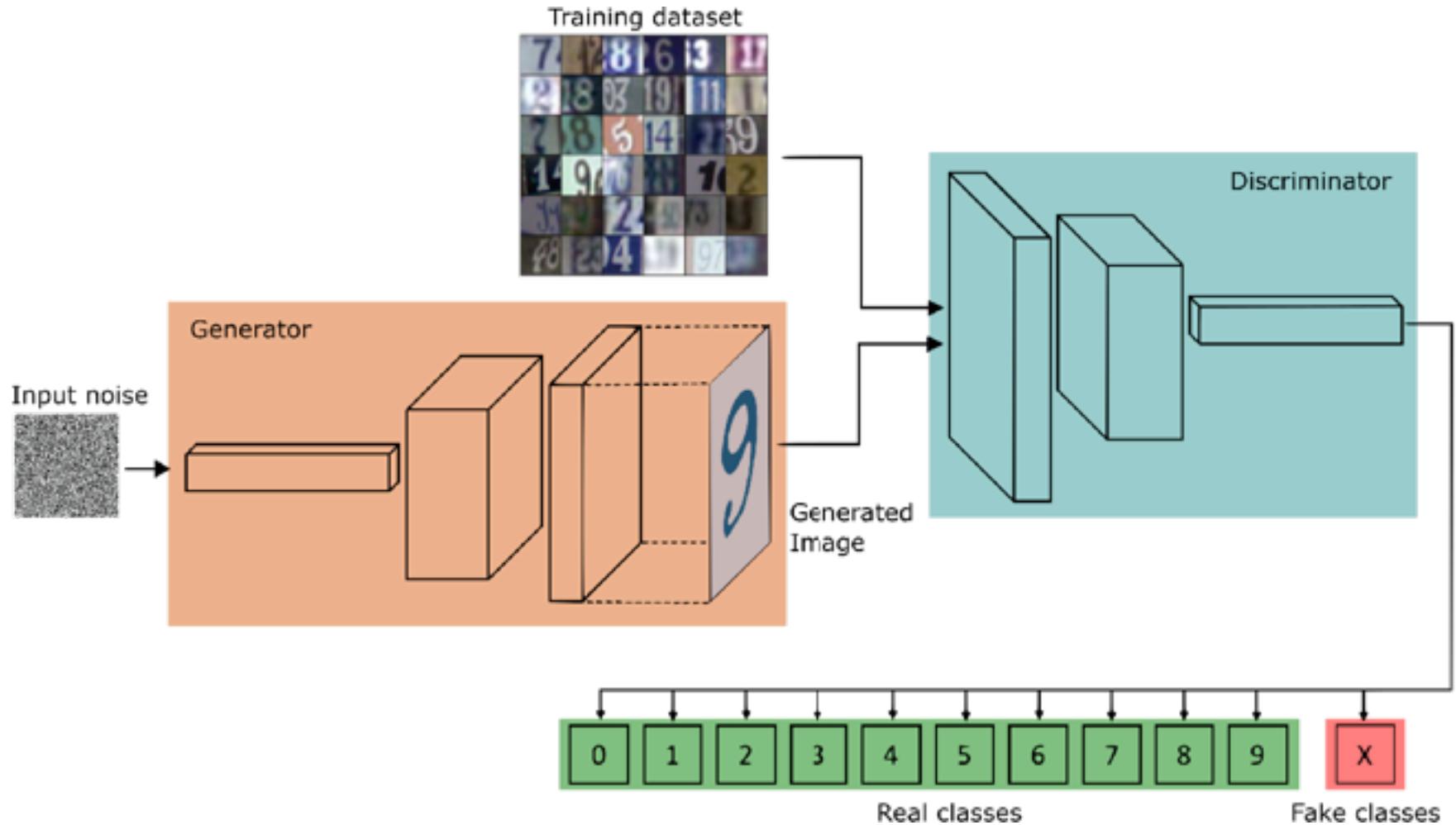
GANs for Improving Classifiers



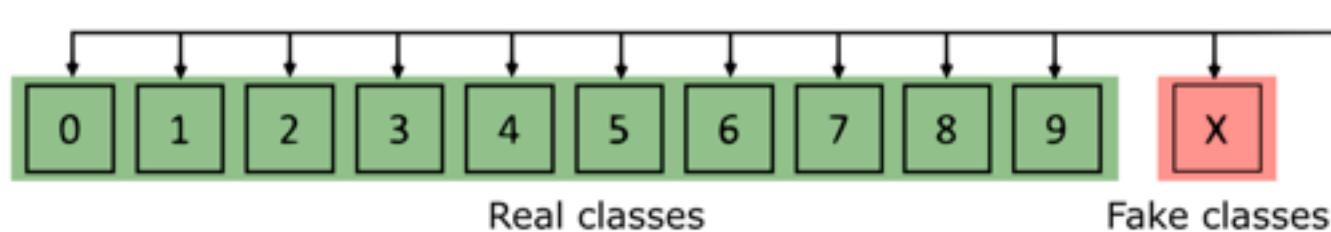
hadou-GAN



Fewer Labels, Still Lots of Data



Minimizing Labels



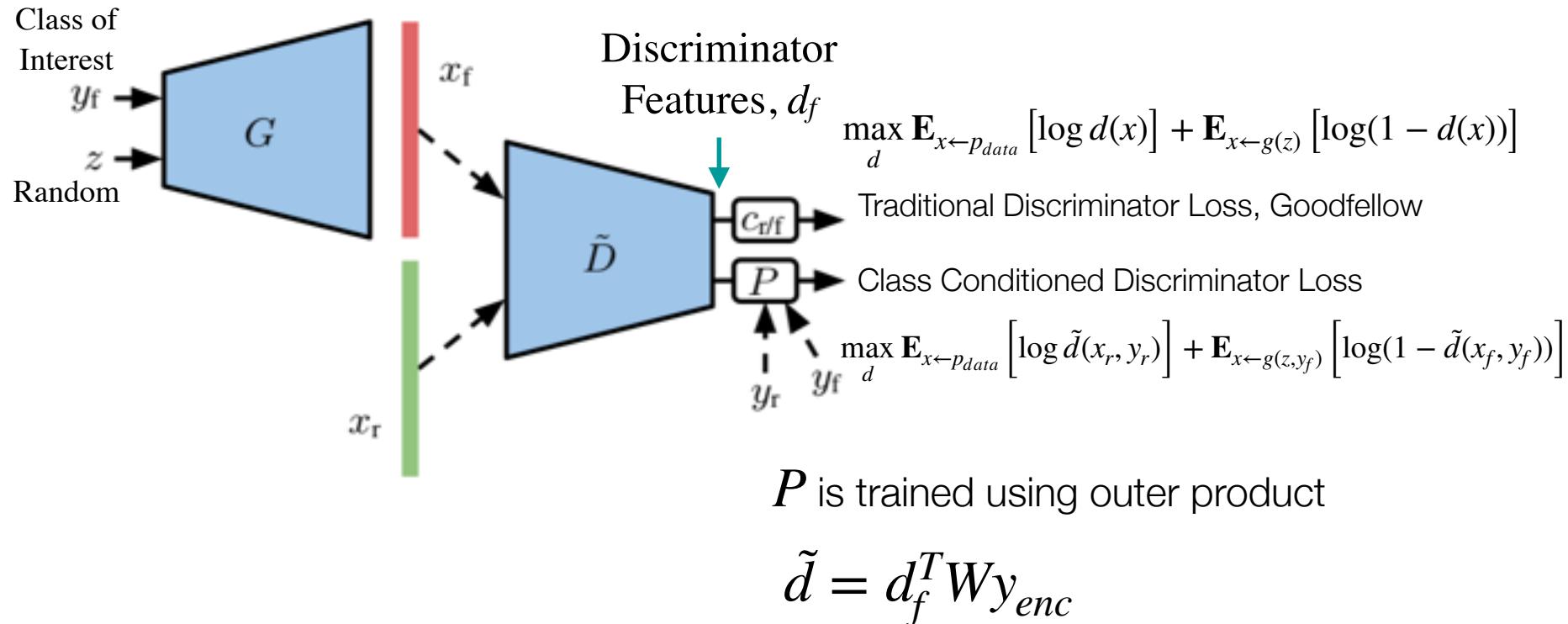
- MNIST Performance
 - 60,000 labels → 1,000 labels
 - 0.6% Error (~SOA)
- SVHN Performance
 - 73,000 labels → 1,000 labels
 - 1.3% Error (~SOA)
- ImageNet
 - 20% Labels (of 1.3M)
 - 10.3% Top-5 Error (slightly worse than SOA)

Need a slightly more complicated loss function



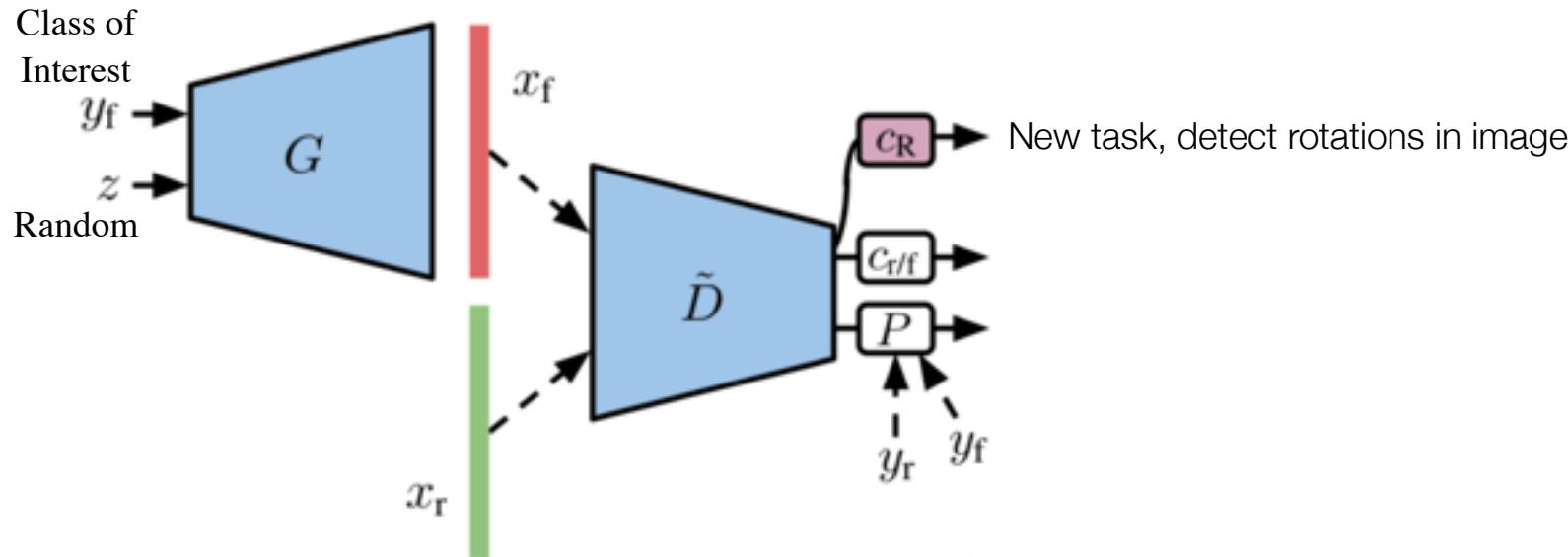
Conditional GANs and Self-Supervision

Lucic, Mario, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. "High-Fidelity Image Generation With Fewer Labels." *arXiv preprint arXiv:1903.02271* (2019).



Conditional GANs and Self-Supervision

Lucic, Mario, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. "High-Fidelity Image Generation With Fewer Labels." *arXiv preprint arXiv:1903.02271* (2019).



All these equations are saying is that they classify rotations from real and fake images.

and

And... nothing in these equations is meaningful except α, β

$$-\frac{\beta}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log p(c_R(\tilde{D}(x^r) = r)]$$

$$-\frac{\alpha}{|\mathcal{R}|} \mathbb{E}_{(z,y) \sim p(z,y)} [\log p(c_R(\tilde{D}(G(z,y)^r) = r)],$$



Conditional GANs and Self-Supervision

Table 3. Top-1 and top-5 error rate (%) on the IMAGENET validation set of $c_{S^2L}(F(x))$ using both self- and semi-supervised losses as described in Section 3.1. While the models are clearly not state-of-the-art compared to the fully supervised IMAGENET classification task, the quality of labels is sufficient to match and in some cases improve the state-of-the-art GAN natural image synthesis.

METRIC	LABELS		
	5%	10%	20%
TOP-1 ERROR	50.08	36.74	29.21
TOP-5 ERROR	26.94	16.04	10.33

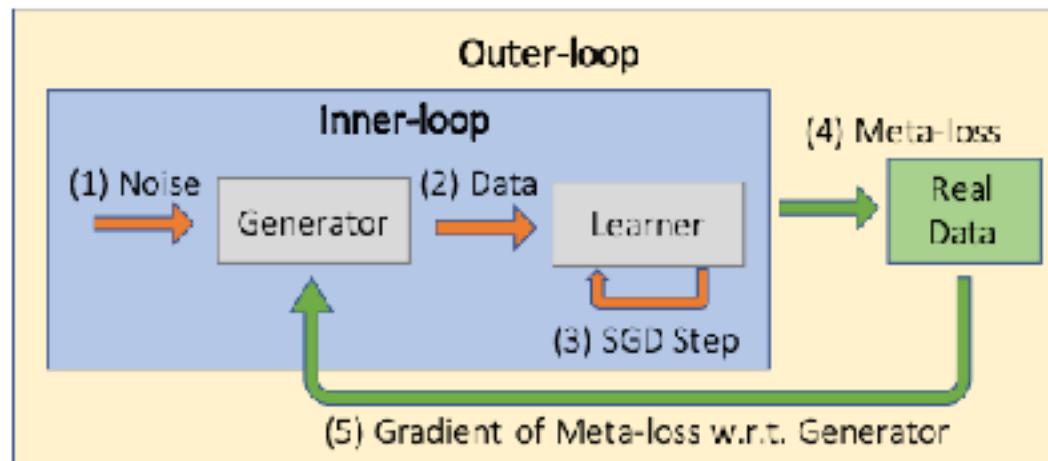
And... nothing in these equations
is meaningful except α, β

$$|\mathcal{R}|^{w(z,y) \sim p(z,y) \text{ loss}} P(\mathcal{R} \sim \{z, y\}) = \prod_j P_j$$



Generative Teaching Networks

- Train a network on generated examples
- Update generator, re-init learners, and repeat



Generator: given a class, provide an image.

Also provide **hyper-params** for SGD like learning rate, momentum scaling, etc.

Goal: Provide examples to train learner as quickly and as accurately as possible

Hyper-parameters are therefore selected through back-propagation techniques...

meta-gradients: <https://arxiv.org/pdf/1502.03492.pdf>

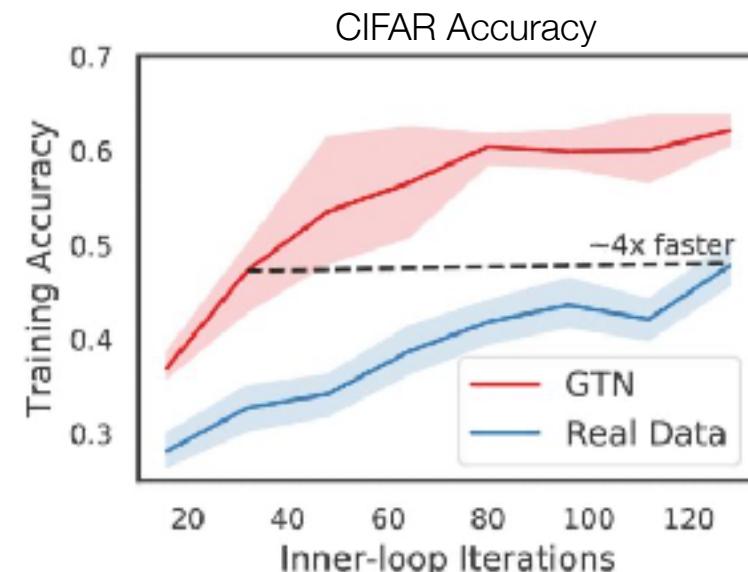
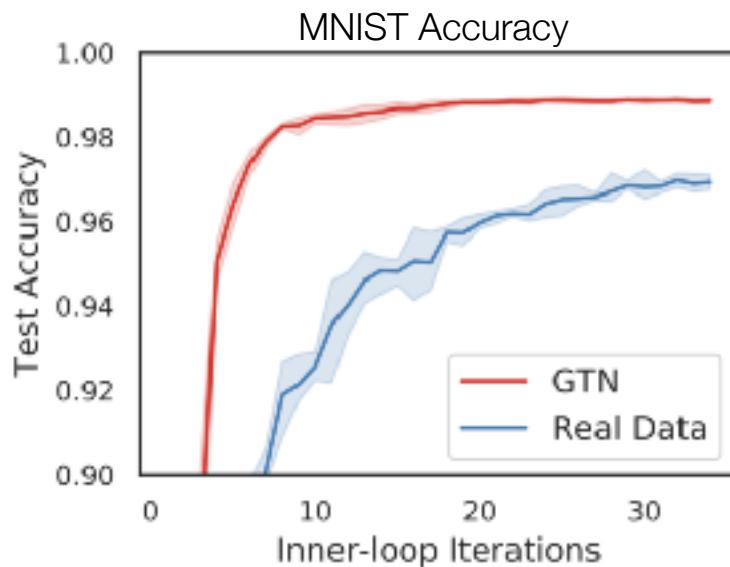


Generative Teaching Networks



Curriculum beginning

Curriculum end



Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Generative Teaching Networks, 2019

<https://arxiv.org/pdf/1912.07768.pdf>



Summary

- When we care about the discriminator for classification:
 - More unsupervised tasks helps feature extraction
 - Conditioning on classes with outer product helps
 - Many other attempts have been tried...
 - ◆ Open research topic!



Text to Image Synthesis with GANs

Generative Adversarial Text to Image Synthesis

**Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran
Bernt Schiele, Honglak Lee**

REEDSCOT¹, AKATA², XCYAN¹, LLAJAN¹
SCHIELE², HONGLAK¹

¹ University of Michigan, Ann Arbor, MI, USA (UMICH.EDU)

² Max Planck Institute for Informatics, Saarbrücken, Germany (MPI-INF.MPG.DE)



Correspondence Images/Text Features

- Need a correspondence function for text and image representations:

sampled from text
for class y

$$f_v(v) = \arg \max_{y \in \mathcal{Y}} \mathbb{E}_{t \sim \mathcal{T}(y)} [\phi(v)^T \varphi(t)]$$

sampled from images
for class y

$$f_t(t) = \arg \max_{y \in \mathcal{Y}} \mathbb{E}_{v \sim \mathcal{V}(y)} [\phi(v)^T \varphi(t)]$$

text description * image description

When text and image match, these products should be large, else smaller. Therefore f_v and f_t will learn to make similar text and image features.

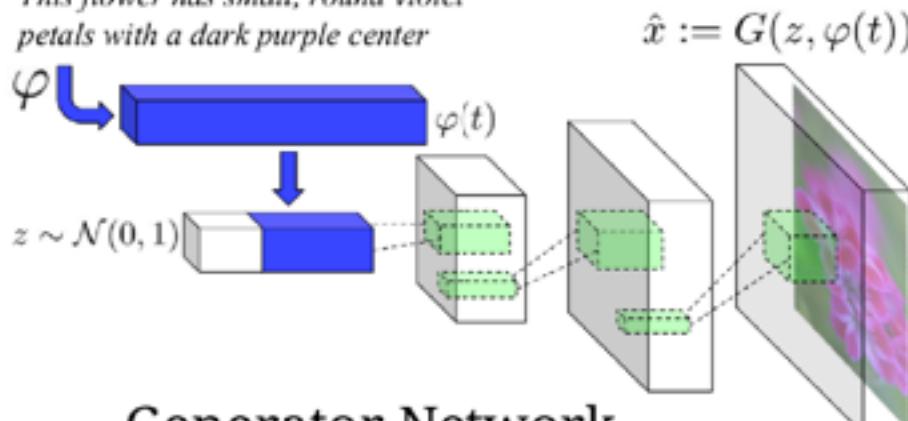
Can also add to loss function by interpolating
text descriptors of the same class

$$\mathbb{E}_{t_1, t_2 \sim p_{data}} [\log(1 - D(G(z, \beta t_1 + (1 - \beta)t_2)))]$$



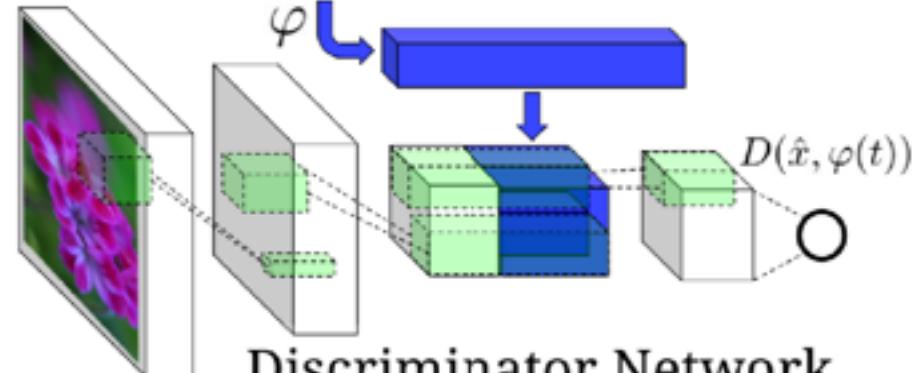
Architecture

This flower has small, round violet petals with a dark purple center



Generator Network

This flower has small, round violet petals with a dark purple center



Discriminator Network

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

Algorithm 1 GAN-CLS training algorithm with step size α , using minibatch SGD for simplicity.

```
1: Input: minibatch images  $x$ , matching text  $t$ , mis-  
   matching  $\hat{t}$ , number of training batch steps  $S$   
2: for  $n = 1$  to  $S$  do  
3:    $h \leftarrow \varphi(t)$  {Encode matching text description}  
4:    $\hat{h} \leftarrow \varphi(\hat{t})$  {Encode mis-matching text description}  
5:    $z \sim \mathcal{N}(0, 1)^Z$  {Draw sample of random noise}  
6:    $\hat{x} \leftarrow G(z, h)$  {Forward through generator}
```

```
7:    $s_r \leftarrow D(x, h)$  {real image, right text}  
8:    $s_w \leftarrow D(x, \hat{h})$  {real image, wrong text}  
9:    $s_f \leftarrow D(\hat{x}, h)$  {fake image, right text}  
10:   $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$   
11:   $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$  {Update discriminator}  
12:   $\mathcal{L}_G \leftarrow \log(s_f)$   
13:   $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$  {Update generator}  
14: end for
```



Results Not Satisfying

- So we need more processing!
- Add attention!

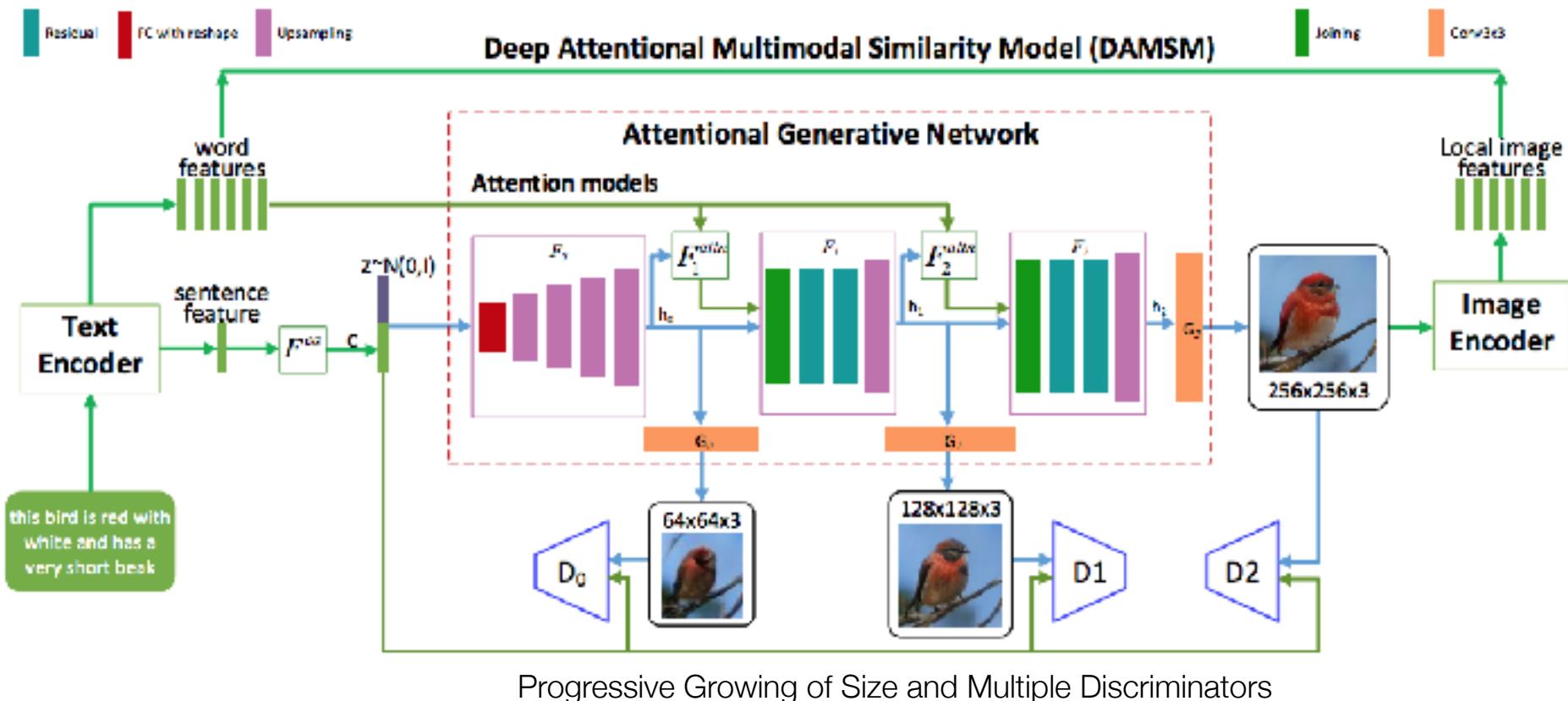
AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks

Tao Xu^{*1}, Pengchuan Zhang², Qiuyuan Huang²,
Han Zhang³, Zhe Gan⁴, Xiaolei Huang¹, Xiaodong He²

¹Lehigh University ²Microsoft Research ³Rutgers University ⁴Duke University
`{tax313, xih206}@lehigh.edu, {penzhan, qihua, xiaohe}@microsoft.com`
`han.zhang@cs.rutgers.edu, zhe.gan@duke.edu`

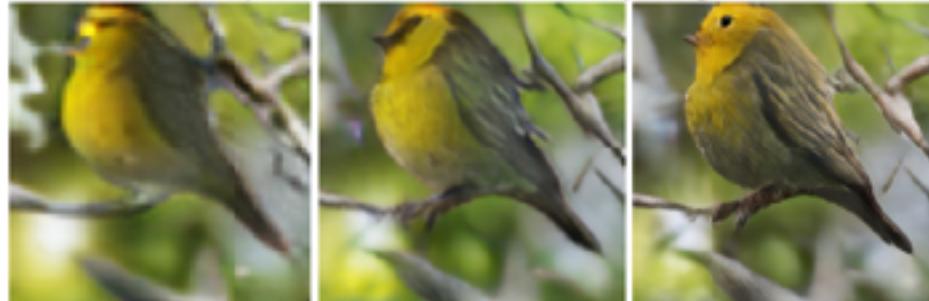


New Process is Similar



Results are more satisfying!

the bird has a yellow crown and a black eyering that is round



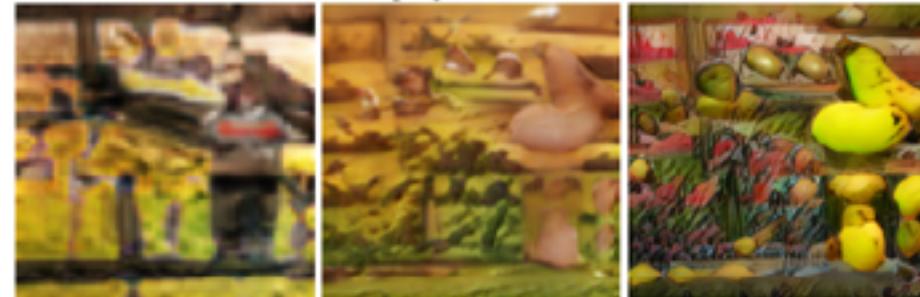
this bird has a green crown black primaries and a white belly



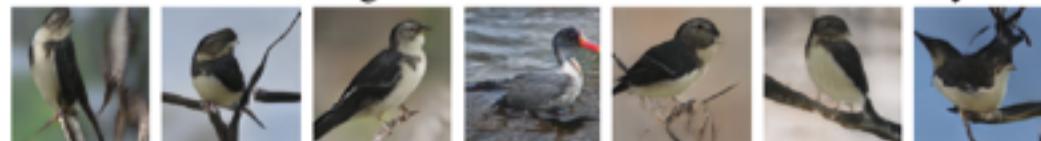
a photo of a homemade swirly pasta with broccoli carrots and onions



a fruit stand display with bananas and kiwi



this bird has wings that are black and has a white belly



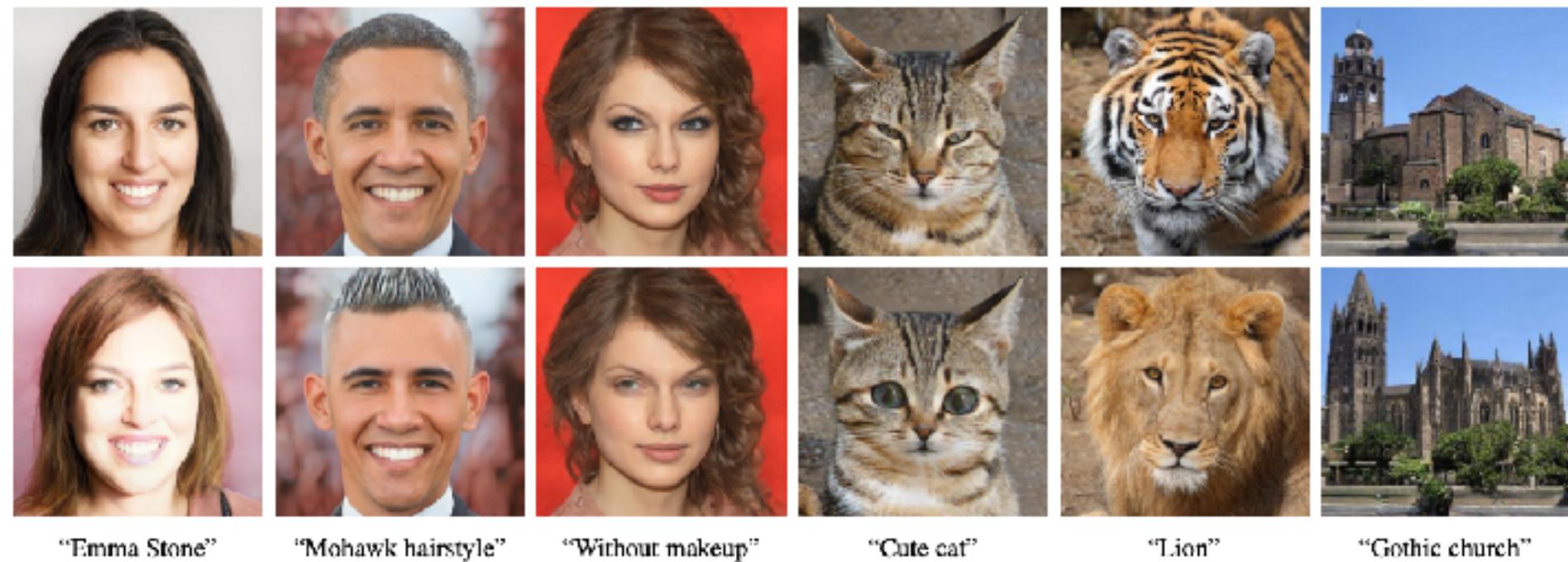
this bird has wings that are red and has a yellow belly



this bird has wings that are blue and has a red belly



StyleGAN and StyleCLIP



<https://arxiv.org/abs/2103.17249>



StyleGAN

- Progressive Growing
- Bilinear Upsampling
- Noise added everywhere!
- Adaptive Input Normalization

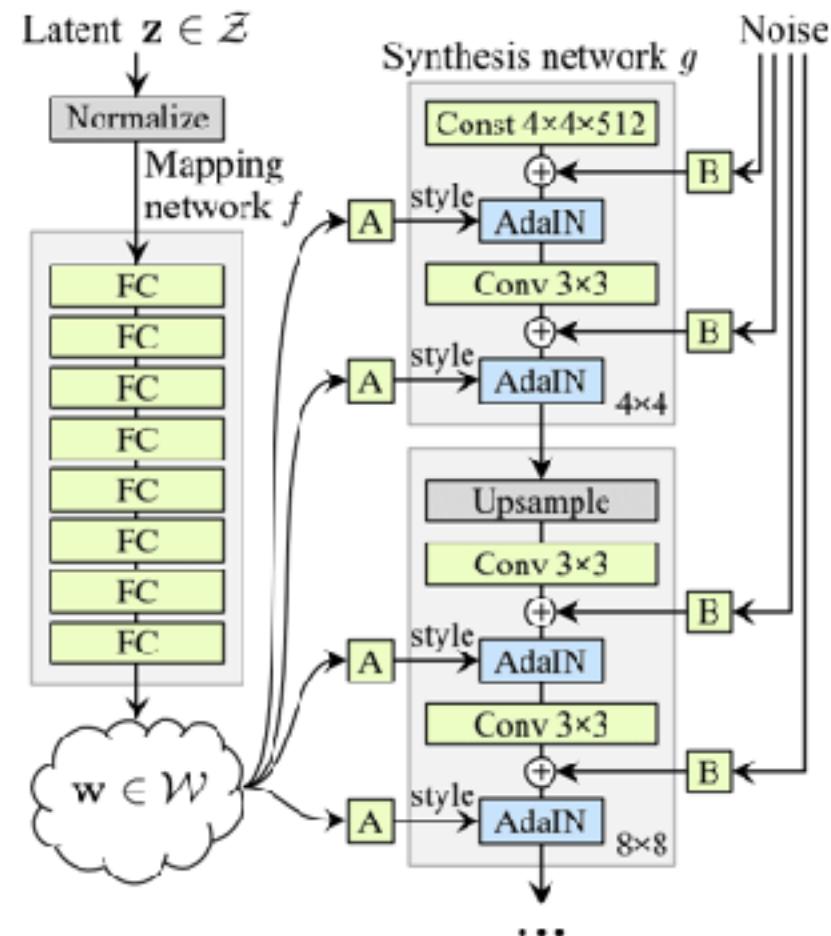


A Style-Based Generator Architecture for Generative Adversarial Networks

Tero Karras
NVIDIA
tkarras@nvidia.com

Samuli Laine
NVIDIA
slaine@nvidia.com

Timo Aila
NVIDIA
taila@nvidia.com



<https://arxiv.org/pdf/1812.04948.pdf>



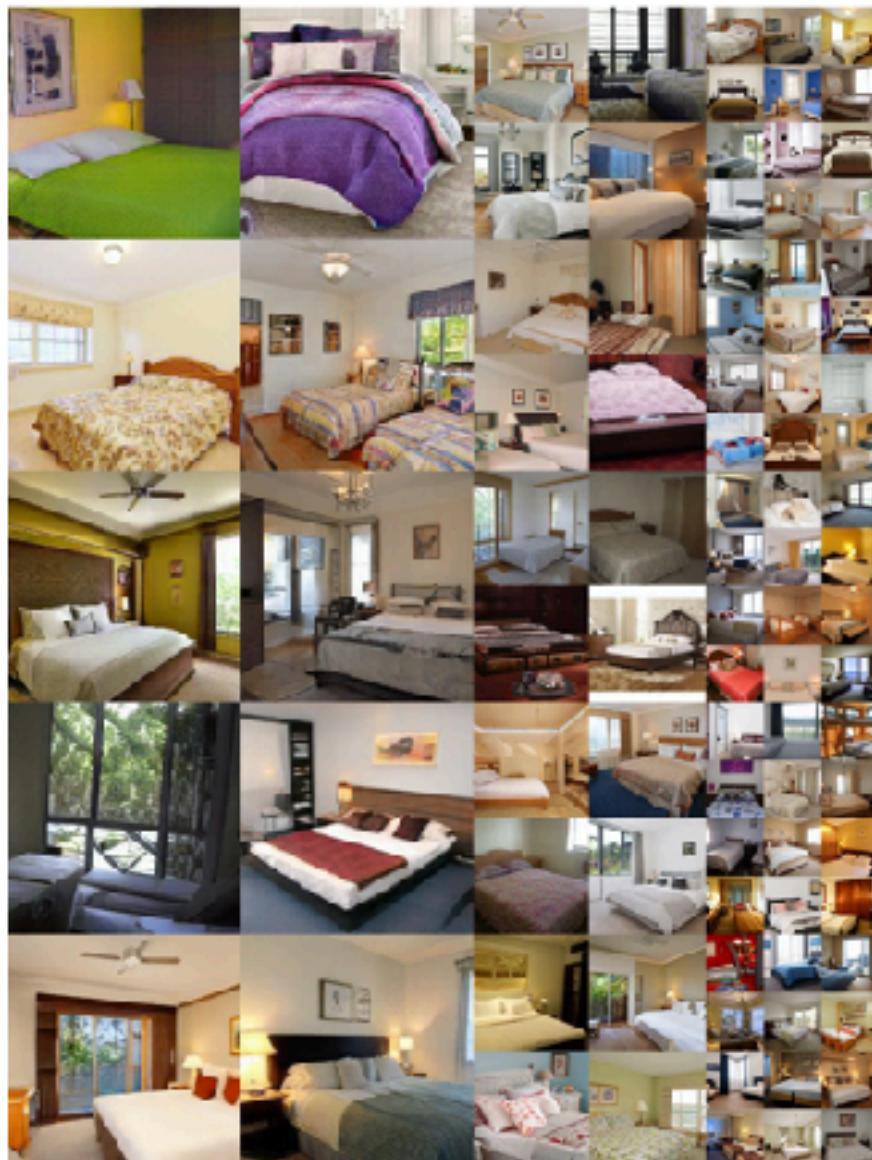


Figure 10. Uncurated set of images produced by our style-based generator (config F) with the LSUN BEDROOM dataset at 256^2 . FID computed for 50K images was 2.65.

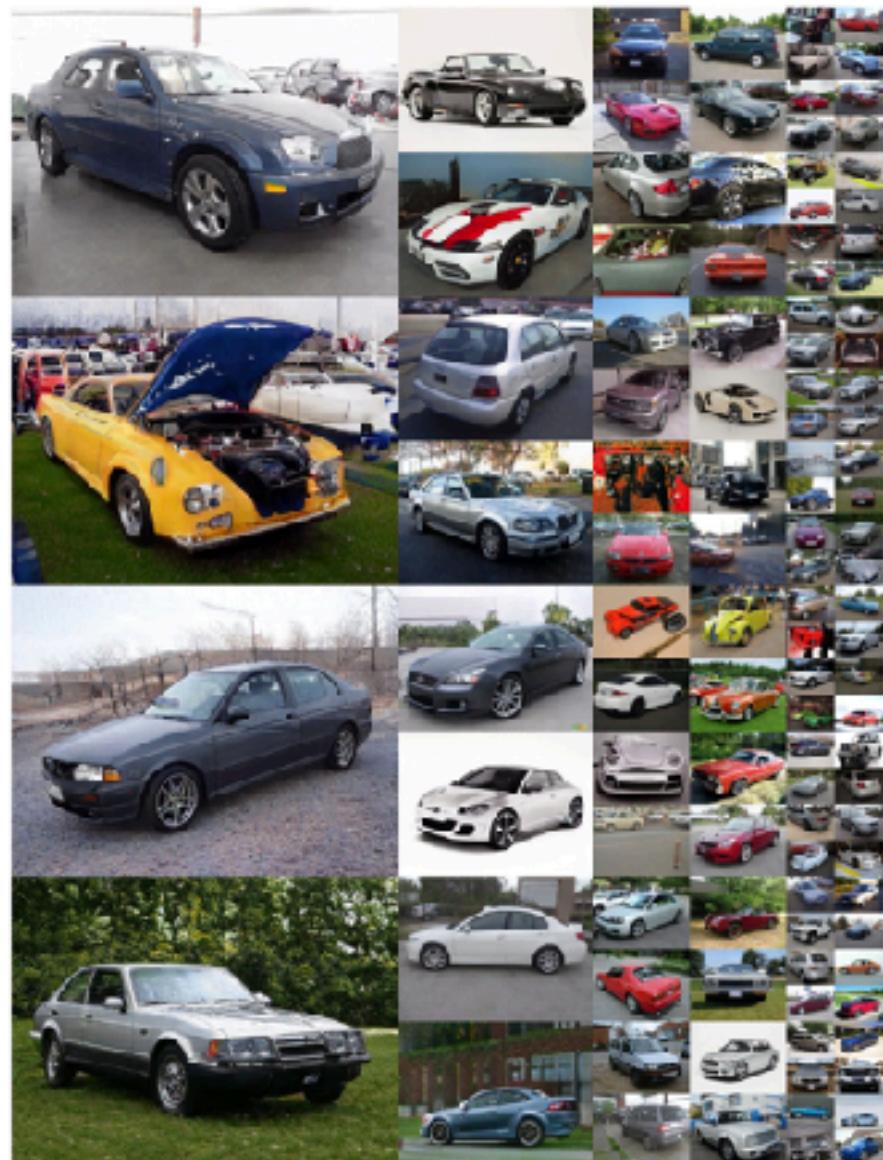


Figure 11. Uncurated set of images produced by our style-based generator (config F) with the LSUN CAR dataset at 512×384 . FID computed for 50K images was 3.27.



StyleCLIP

- Also add in noise representations from text, guides the latent space...

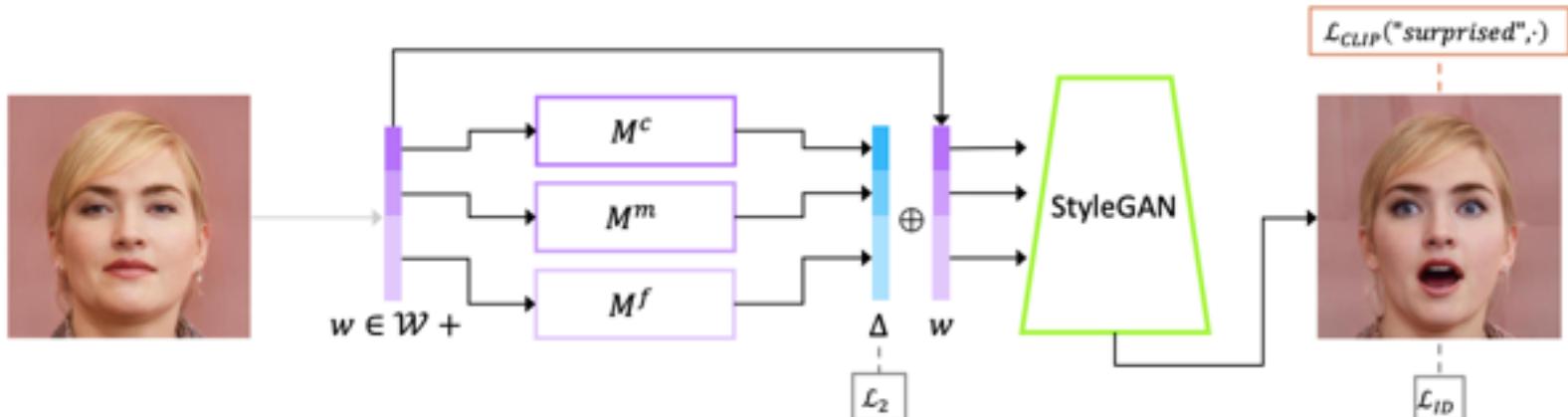


Figure 2. The architecture of our text-guided mapper (using the text prompt “surprised”, in this example). The source image (left) is inverted into a latent code w . Three separate mapping functions are trained to generate residuals (in blue) that are added to w to yield the target code, from which a pretrained StyleGAN (in green) generates an image (right), assessed by the CLIP and identity losses.

StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery

Or Patashnik^{†*} Zongze Wu^{†*} Eli Shechtman[§] Daniel Cohen-Or[†] Dani Lischinski[†]
†Hebrew University of Jerusalem [†]Tel-Aviv University [§]Adobe Research

<https://arxiv.org/abs/2103.17249>

37



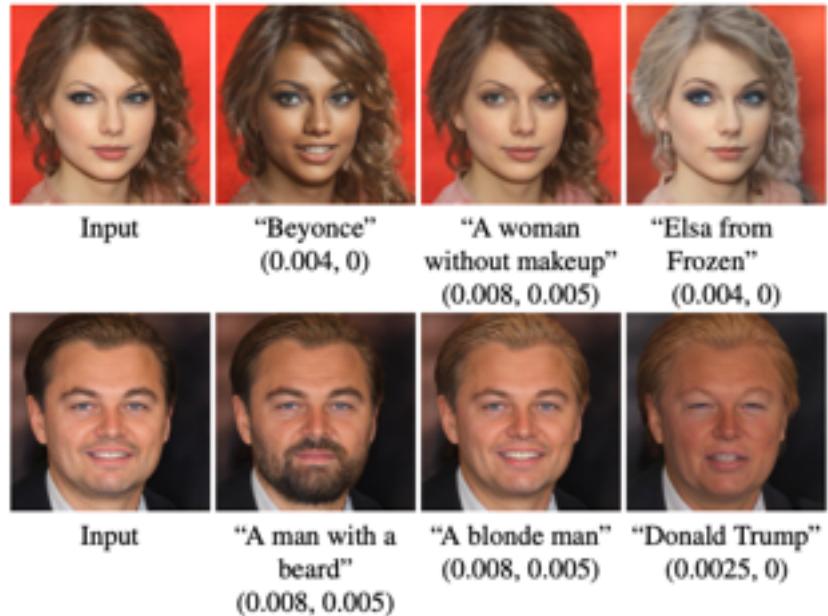
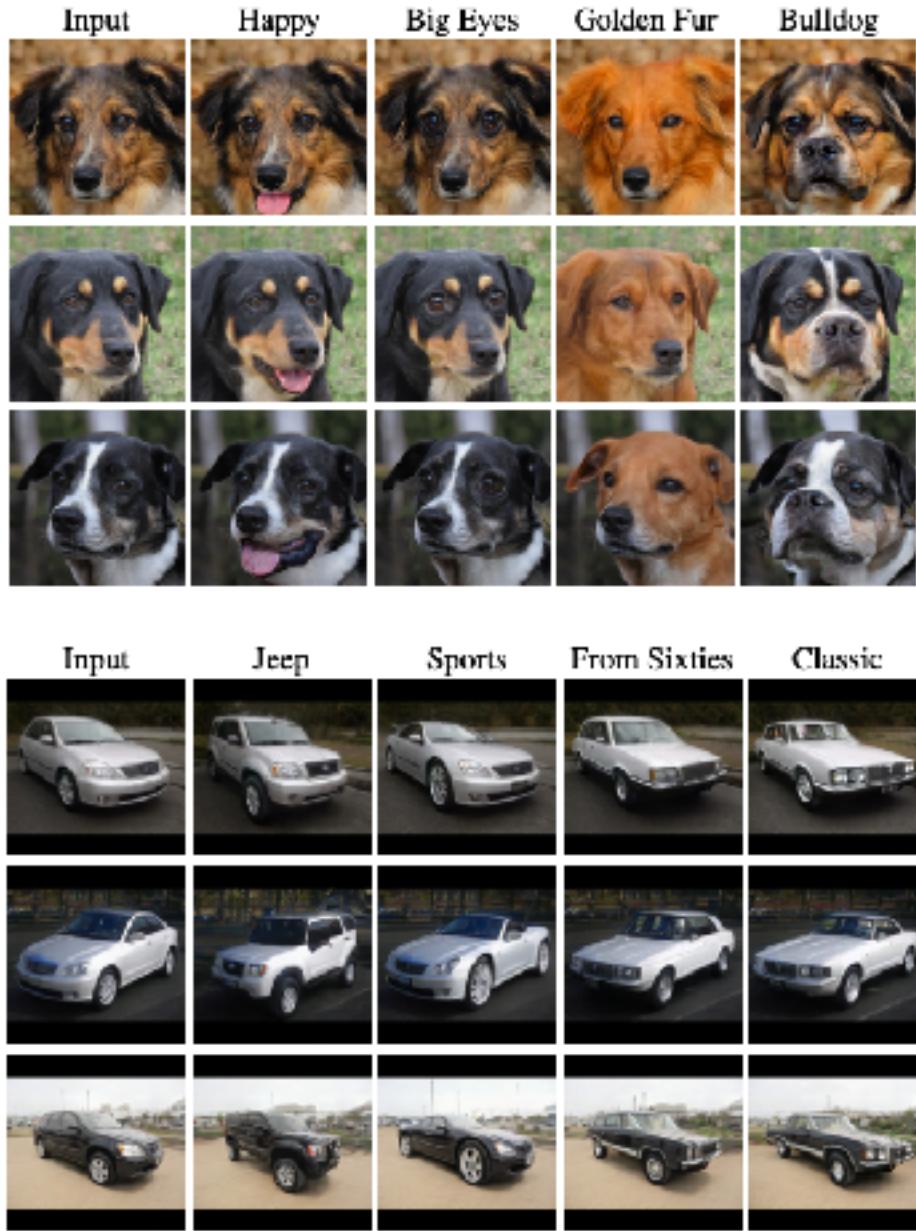
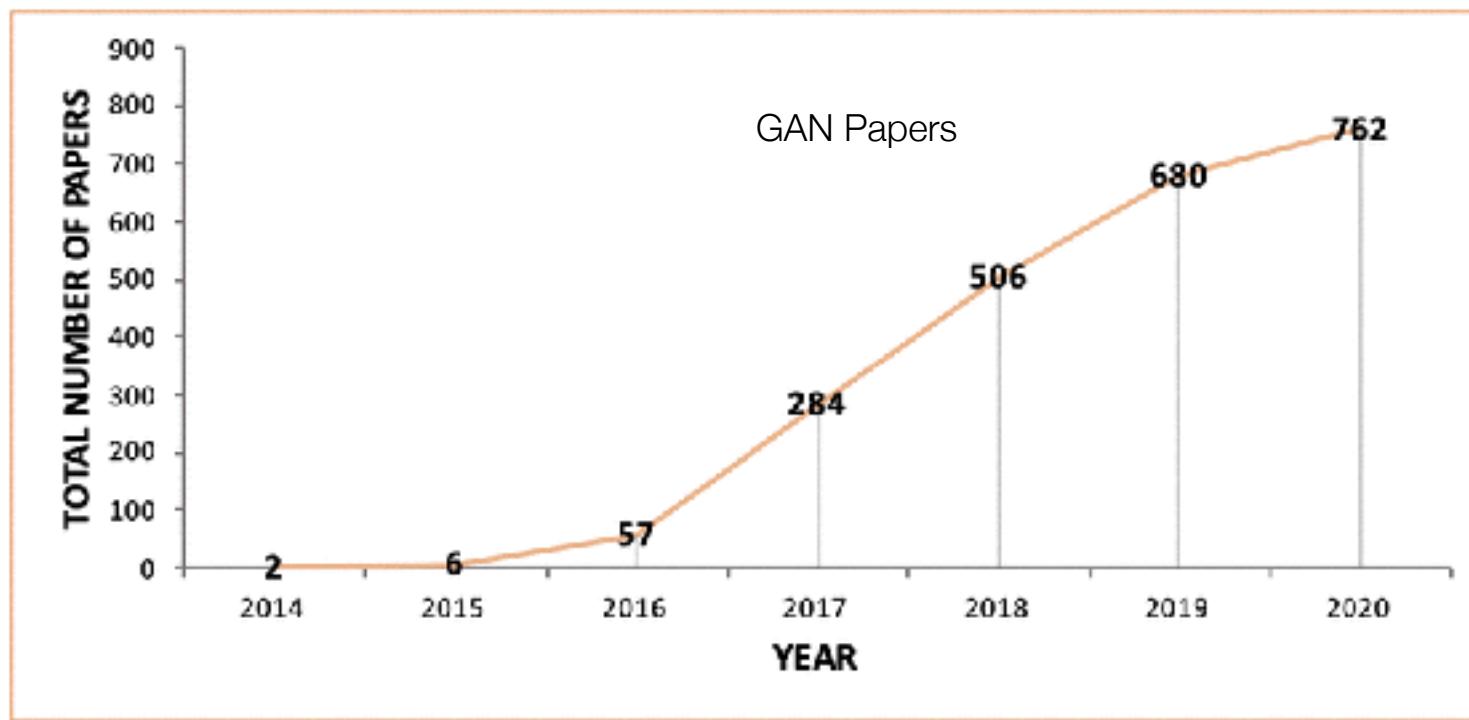


Figure 3. Edits of real celebrity portraits obtained by latent optimization. The driving text prompt and the $(\lambda_{L2}, \lambda_{ID})$ parameters for each edit are indicated under the corresponding result.



The END! Lots of open research problems!

- Look at some of the recent research
- There are so many individuals in the field
- ...and so many great applications.



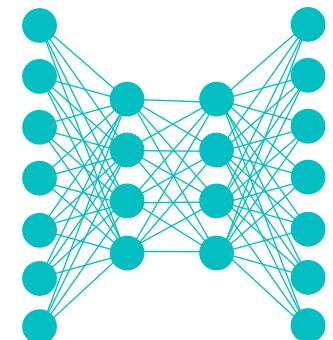


Lecture Notes for **Neural Networks** **and Machine Learning**

Holy GAN-Zooks



Next Time:
Lapan Ch. 1
Reading: None





Backup slides



A Quick Aside



Deep Fakes: A Looming Crisis for National Security, Democracy and Privacy?



Figure 1: (a) The input image. (b) The result of face swapping with Nicolas Cage using our method. (c) The result of a manual face swap (source: <http://niccageaseeveryone.blogspot.com>).

By Robert Chesney, Danielle Citron · Wednesday, February 21, 2018, 10:00 AM

Privacy Paradox: Rethinking Solitude

Bobby Chesney is the Charles E. Francis Professor in Law and Associate Dean for Academic Affairs at the University of Texas School of Law. He also serves as the Director of UT-Austin's interdisciplinary research center the Robert S. Strauss Center for International Security and Law. His scholarship encompasses a wide range of issues relating to national security and the law, including detention, targeting, prosecution, covert action, and the state secrets privilege; most of it is posted here. Along with Ben Wittes and Jack Goldsmith, he is one of the co-founders of the blog.

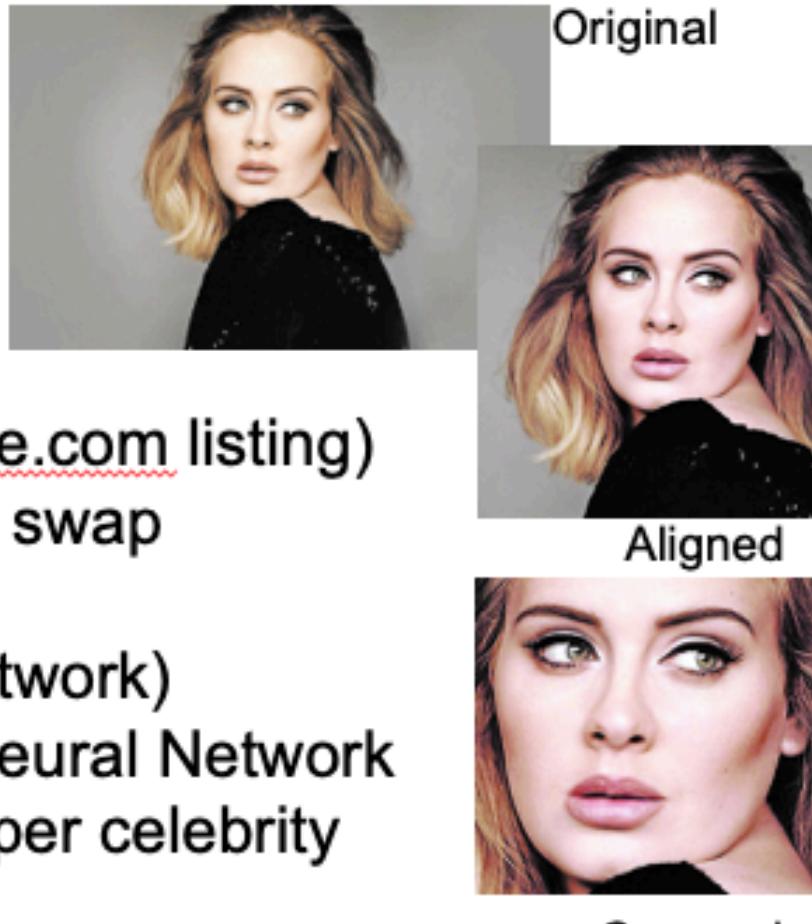
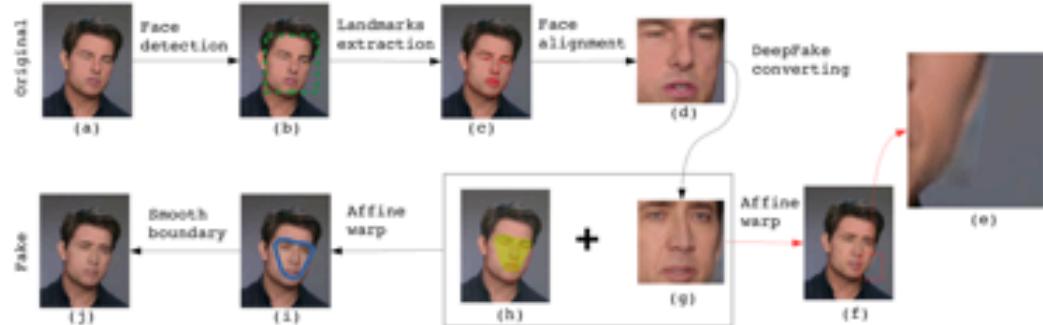
#BobbyChesney

Danielle Citron is the Morton B. Sophia Macht Professor of Law at the University of Maryland-Carey School of Law. She is the author of *Hate Crimes in Cyberspace* (Harvard University Press 2014).

"As deepfake technology becomes more advanced and more accessible, it could pose a threat to United States public discourse and national security, with broad and concerning implications for offensive active measured campaigns targeting the United States."

"US lawmakers say AI Deepfakes have the potential to disrupt every facet of our society." US Congressional Letter





- 50 different celebrities (from [People.com](#) listing)
- Selected celebrity pairs of faces to swap
- Swapped Faces via:
 - Auto-Encoding (Adversarial Network)
 - Pipelining with Convolutional Neural Network
- 1000 – 3000 fake images created per celebrity pair
- 400,000 images for training and evaluation



Training Phase

Person A

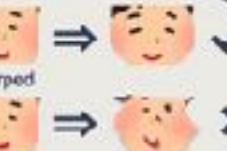


Real
face A



Warped
face A

Encoder



Decoder
A

Segmentation
mask



Reconstructed
face A



Reconstructed
face A

Masked
face A



Warped
face A

Test Phase

Person B



Real
face B

Encoder

Decoder
A



Result: face B
(face A look-alike)

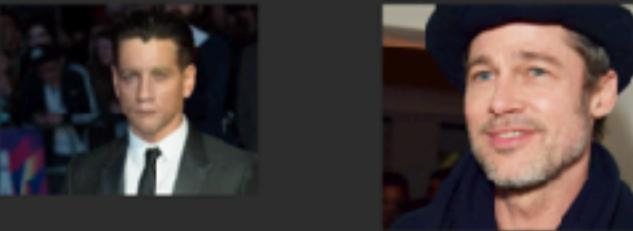




Rating Swapped Images

 **SMU. FaceFace** VOTE

Which of the following two faces looks
MORE FAKE to you?



faceface.lyle.smu.edu

- Selected subset of 400 images to rate
- Used paired selection algorithm on website to choose most “needed” image comparison
- Collected **40,000 ratings** from ~200 unique individuals
- **Result:** Sorting of images from most fake to most real

- <https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.5-introduction-to-gans.ipynb>
- <https://sthalles.github.io/semi-supervised-learning-with-gans/>
- <https://openai.com/blog/generative-models/#gan>
- <https://openai.com/blog/glow/>
- <https://arxiv.org/pdf/1807.03039.pdf>
- <https://arxiv.org/pdf/1606.03498.pdf>
- https://github.com/sthalles/blog-resources/blob/master/semi-supervised/semi-supervised_learning.ipynb
- <https://arxiv.org/pdf/1701.07875.pdf>
- https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
- <https://medium.com/nurture-ai/keeping-up-with-the-gans-66e89343b46>
- Reversible Flow Algorithms (<https://arxiv.org/pdf/1807.03039.pdf>)



InfoGAN

InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Xi Chen^{†‡}, Yan Duan^{†‡}, Rein Houthooft^{†‡}, John Schulman^{†‡}, Ilya Sutskever[‡], Pieter Abbeel^{†‡}

† UC Berkeley, Department of Electrical Engineering and Computer Sciences

‡ OpenAI



InfoGAN Problem Statement

- We need to disentangle information in the latent space for it to be truly useful
- Disentangling should be ascertained by the ability of the generator to have interpretable latent space axes
- Each axis should have little impact on the other axes, which is not imposed by any GAN structure yet
 - *For example, for a dataset of faces, a useful disentangled representation may allocate a separate set of dimensions for each of the following attributes: facial expression, eye color, hairstyle, presence or absence of eyeglasses, and the identity of the corresponding person.*



How to perform latent space arithmetic?



man
with glasses

Need to find a
bunch of these



man
without glasses

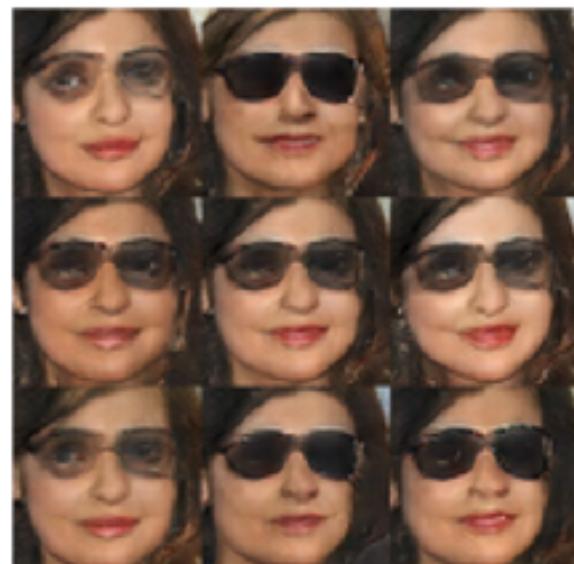
Need to find a
bunch of these



woman
without glasses

Need to find a
bunch of these

NOT Systematic!



woman with glasses

To generate all these



The InfoGAN Money Back Guarantee

However, many domains naturally decompose into a set of semantically meaningful factors of variation. For instance, when generating images from the MNIST dataset, it would be ideal if the model automatically chose to allocate a discrete random variable to represent the numerical identity of the digit's angle and by simply specifying continuous variables.

**Authors: by Maximizing Mutual Information
the network will learn to disentangle latent space**

- Decompose latent variable representation into
 - z : continuous incompressible noise
 - c : latent code, hopefully learns semantic features
- We want mutual information of c and $G(z,c)$ to be large
- Using standard entropy definition: $I(X;Y)=H(X)-H(X|Y)$
- New Loss Function: $\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$
Regular GAN Loss MI Loss



Maximizing Mutual Information

- You guessed it, Mutual Information with P is intractable to compute!
- So let's dust off our approximation pants and put some ELBO grease into this
- Variational Inference Maximization, Assume there is an approximation of P from Q :

$$\begin{aligned} I(c; G(z, c)) &= H(c) - H(c|G(z, c)) \\ &= \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log P(c'|x)]] + H(c) && \text{Expand Def. of Conditional Entropy} \\ &= \mathbb{E}_{x \sim G(z, c)} [\underbrace{D_{\text{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\text{make Q close to P}} + \mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) && \begin{array}{l} \text{Insert Q Approximation} \\ \text{sample from Q} \end{array} \\ &\geq \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) && \begin{array}{l} \text{Define a Lower Bound to Maximize because} \\ \text{we don't like other terms...} \end{array} \end{aligned}$$

Lemma 5.1 *For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:*
 $\mathbb{E}_{x \sim X, y \sim Y|x} [f(x, y)] = \mathbb{E}_{x \sim X, y \sim Y|x, x' \sim X|y} [f(x', y)].$



Maximizing Mutual Information (kinda)

$$I(c; G(z, c)) \geq \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c)$$

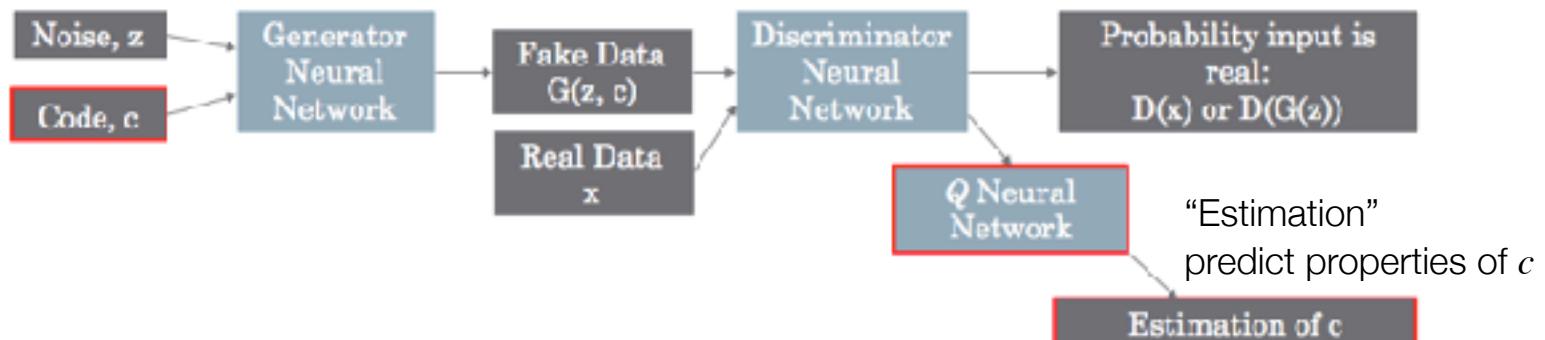
Lemma 5.1 For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:
 $\mathbb{E}_{x \sim X, y \sim Y|x} [f(x, y)] = \mathbb{E}_{x \sim X, y \sim Y|x, x' \sim X|y} [f(x', y)].$

$$\begin{aligned} L_I(G, Q) &= \mathbb{E}_{c \sim P(c), x \sim G(z, c)} [\log Q(c|x)] + H(c) \\ &= \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) \\ &\leq I(c; G(z, c)) \end{aligned}$$

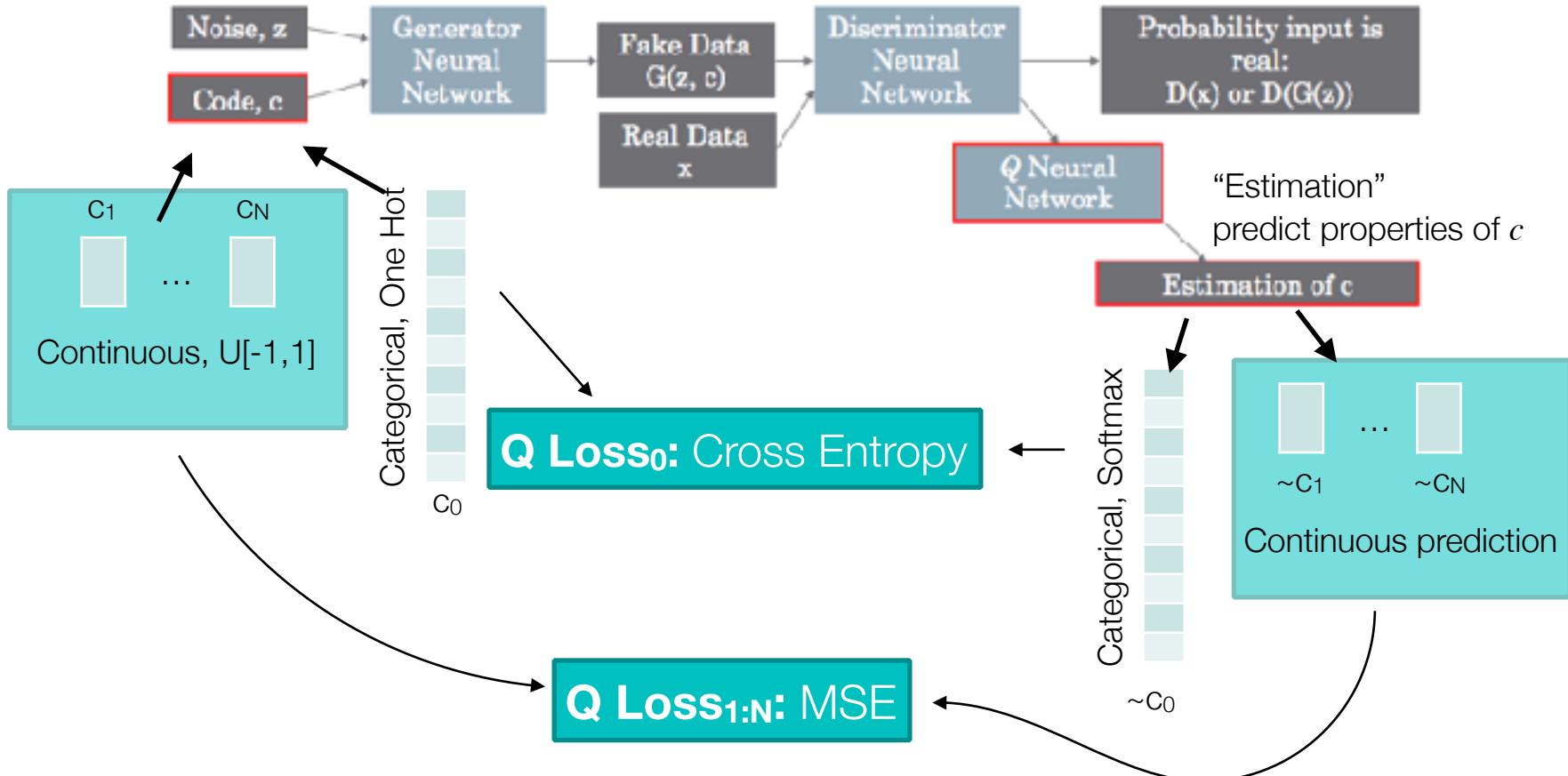
Re-parameterize:

Q will be good if we can measure
 c from the Generator network

New Objective Function Becomes: $\min_{G, Q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q)$



Measuring Latent Codes in Loss



Intuition: If the generator starts to use these codes and variables semantically, then the Q network (and therefore the discriminator) can start learning to predict these semantics. By learning these semantics, the generator is incentivized to produce disentangled, meaningful features from the codes.



Implementation: D and Q

```
# define the standalone discriminator model
def define_discriminator(n_cat, in_shape=(28,28,1)):
    # weight initialization
    init = RandomNormal(stddev=0.02)
    # image input
    in_image = Input(shape=in_shape)
    # downsample to 14x14
    d = Conv2D(64, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(in_image)
    d = LeakyReLU(alpha=0.1)(d)
    # downsample to 7x7
    d = Conv2D(128, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
    d = LeakyReLU(alpha=0.1)(d)
    d = BatchNormalization()(d)
    # normal
    d = Conv2D(256, (4,4), padding='same', kernel_initializer=init)(d)
    d = LeakyReLU(alpha=0.1)(d)
    d = BatchNormalization()(d)
    # flatten feature maps
    d = Flatten()(d)
    # real/fake output
    out_classifier = Dense(1, activation='sigmoid')(d)
    # define d model
    d_model = Model(in_image, out_classifier)
    # compile d model
    d_model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5))
    # create q model layers
    q = Dense(128)(d)
    q = BatchNormalization()(q)
    q = LeakyReLU(alpha=0.1)(q)
    # q model output
    out_codes = Dense(n_cat, activation='softmax')(q)
    # define q model
    q_model = Model(in_image, out_codes)
    return d_model, q_model
```

<https://machinelearningmastery.com/how-to-develop-an-information-maximizing-generative-adversarial-network-infogan-in-keras/>

57



Implementation: G and Q Trained

```
# define the combined discriminator, generator and q network model
def define_gan(g_model, d_model, q_model):
    # make weights in the discriminator (some shared with the q model) as not trainable
    d_model.trainable = False
    # connect g outputs to d inputs
    d_output = d_model(g_model.output)
    # connect g outputs to q inputs
    q_output = q_model(g_model.output)
    # define composite model
    model = Model(g_model.input, [d_output, q_output])
    # compile model
    opt = Adam(lr=0.0002, beta_1=0.5)
    model.compile(loss=['binary_crossentropy', 'categorical_crossentropy'], optimizer=opt)
    return model
```

```
# generate points in latent space as input for the generator
```

```
def generate_latent_points(latent_dim, n_cat, n_samples):
```

```
    # generate points in the latent space
```

```
    z_latent = randn(latent_dim * n_samples)
```

```
    # reshape into a batch of inputs for the network
```

```
    z_latent = z_latent.reshape(n_samples, latent_dim)
```

```
    # generate categorical codes
```

```
    cat_codes = randint(0, n_cat, n_samples)
```

Only uses codes, not continuous “c” vars

```
    # one hot encode
```

```
    cat_codes = to_categorical(cat_codes, num_classes=n_cat)
```

```
    # concatenate latent points and control codes
```

```
    z_input = hstack((z_latent, cat_codes))
```

```
    return [z_input, cat_codes]    Input to generator and labels for Q network returned (repeats “c”)
```

<https://machinelearningmastery.com/how-to-develop-an-information-maximizing-generative-adversarial-network-infogan-in-keras/>



Implementation: Training

```
# train the generator and discriminator
def train(g_model, d_model, gan_model, dataset, latent_dim, n_cat, n_epochs=100, n_batch=64):
    # calculate the number of batches per training epoch
    bat_per_epo = int(dataset.shape[0] / n_batch)
    # calculate the number of training iterations
    n_steps = bat_per_epo * n_epochs
    # calculate the size of half a batch of samples
    half_batch = int(n_batch / 2)
    # manually enumerate epochs
    for i in range(n_steps):
        # get randomly selected 'real' and 'fake' samples
        X_real, y_real = generate_real_samples(dataset, half_batch)
        X_fake, y_fake = generate_fake_samples(g_model, latent_dim, n_cat, half_batch)
        # update discriminator and q model weights
        d_loss1 = d_model.train_on_batch(X_real, y_real)
        d_loss2 = d_model.train_on_batch(X_fake, y_fake)
        # prepare points in latent space as input for the generator
        z_input, cat_codes = generate_latent_points(latent_dim, n_cat, n_batch)
        # create inverted labels for the fake samples
        y_mislead = ones((n_batch, 1))
        # update the g via the d and q error
        _,g_1,g_2 = gan_model.train_on_batch(z_input, [y_mislead, cat_codes])
        # summarize loss on this batch
        print('>%d, d[%.*f,%.*f], g[%.*f] q[%.*f]' % (i+1, d_loss1, d_loss2, g_1, g_2))
        # evaluate the model performance every 'epoch'
        if (i+1) % (bat_per_epo * 10) == 0:
            summarize_performance(i, g_model, gan_model, latent_dim, n_cat)
```



Did they really disentangle?

$c_1 = 10$ discrete codes with equal probability 0.1

c_2 and c_3 = uniformly sampled data from -1 to 1

0 1 2 3 4 5 6 7 8 9	7 7 7 7 7 7 7 7 7 7
0 1 2 3 4 5 6 7 8 7	0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9	7 7 7 7 7 7 7 7 7 7
0 1 2 3 4 5 6 7 8 9	9 9 9 9 9 9 9 9 9 9
0 1 2 3 4 5 6 7 8 9	8 8 8 8 8 8 8 8 8 8

(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1
8 8 8 8 8 8 8 8 8 8	8 8 8 8 8 8 8 8 8 8
3 3 3 3 3 3 3 3 3 3	3 3 3 3 3 3 3 3 3 3
9 9 9 9 9 9 9 9 9 9	9 9 9 9 9 9 9 9 9 9
5 5 5 5 5 5 5 5 5 5	5 5 5 5 5 5 5 5 5 5

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Columns: Hold other variables constant, Rows: Vary named variable



Did they really disentangle?



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow

Figure 3: **Manipulating latent codes on 3D Faces:** We show the effect of the learned continuous



Did they really disentangle?



(a) Azimuth (pose)

(b) Presence or absence of glasses



(c) Hair style

(d) Emotion



GAIA



Tim Sainburg

PhD Student @ UCSD studying
Psychology, Neuroscience,
Anthropogeny, Animal Communication,
and Machine Learning

Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions

Tim Sainburg
UC San Diego
tsainbur@ucsd.edu

Marvin Thielk
UC San Diego
mthielk@ucsd.edu

Brad Theilman
UC San Diego
btheilm@ucsd.edu

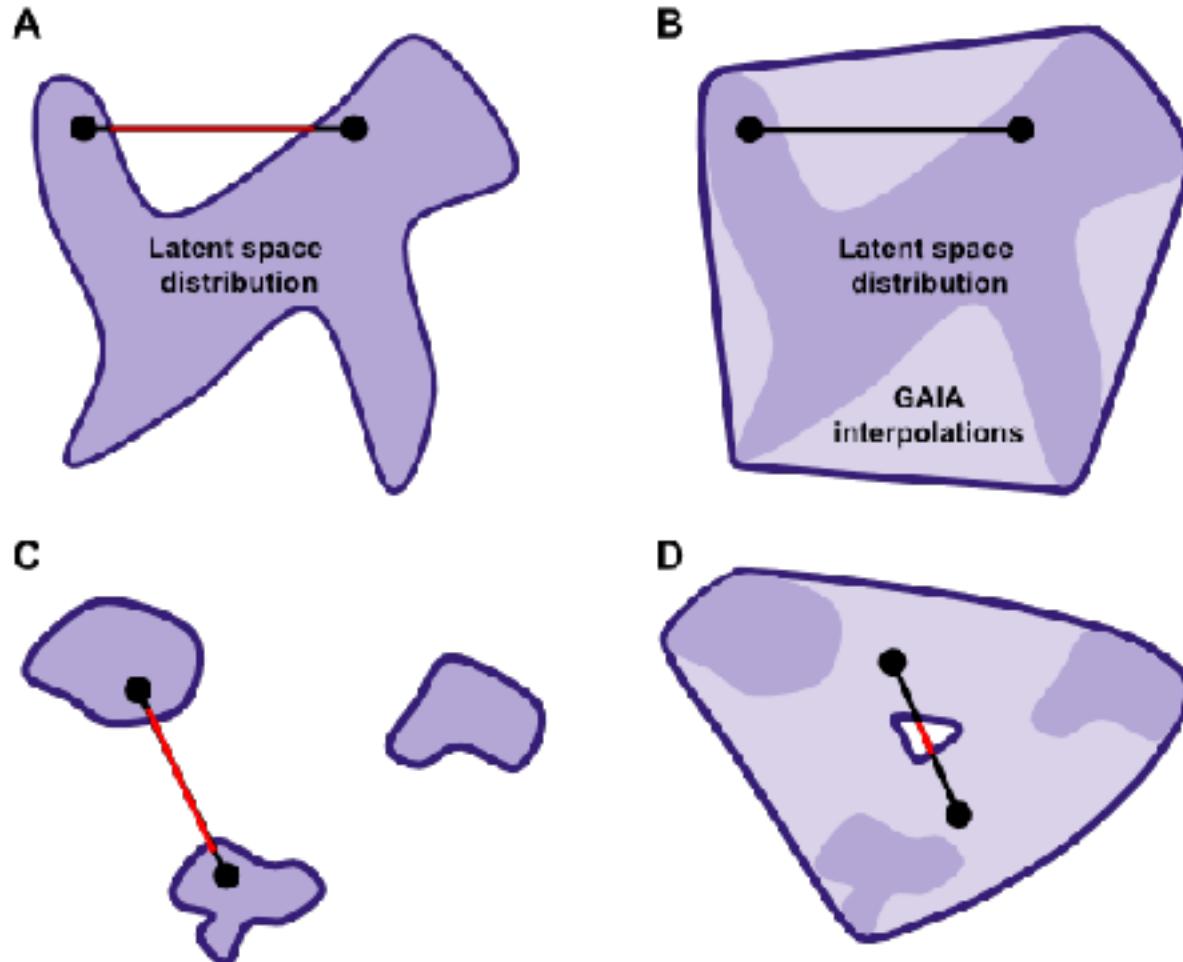
Benjamin Migliori
SPAWAR
migliori@spawar.navy.mil

Timothy Gentner
UC San Diego
tgentner@ucsd.edu



What does GAIA address?

- Explicitly Interpolate to make latent space as **convex** as possible



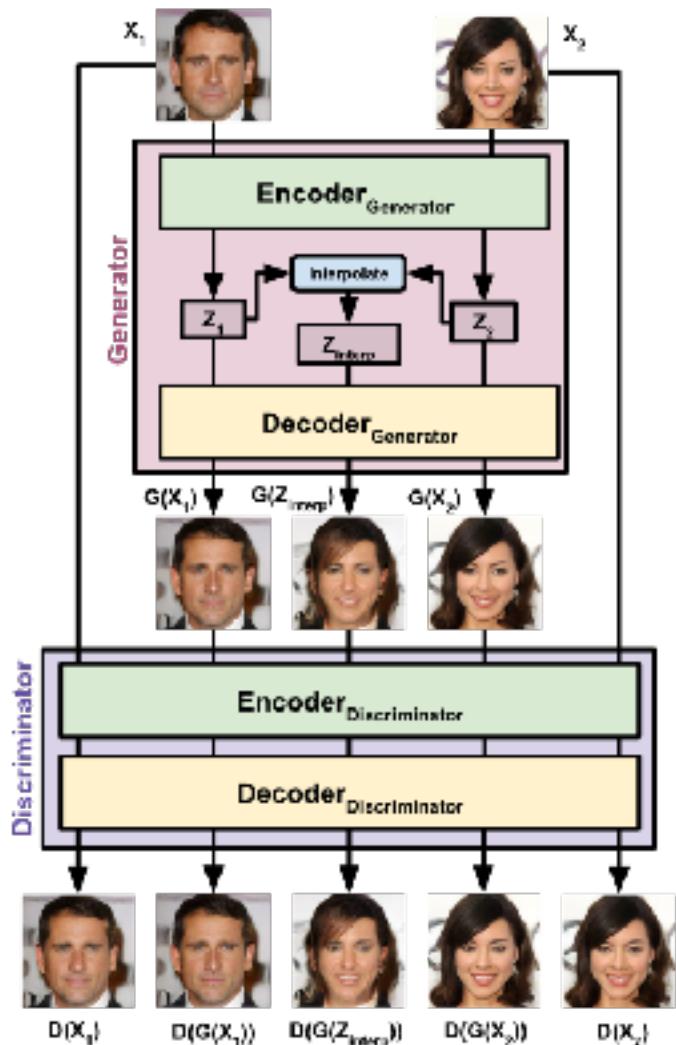
<https://timsainburg.com/gaia.html#gaia>

How to enforce convexity?

- Change discriminator function entirely
 - Discriminator also auto-encodes an image
 - Maximize **reconstruction error** in generated image for **fake**
 - Minimize **reconstruction error** in generated image for **real**
- Generator takes two images and outputs three
 - Two are simply auto encoded
 - Third Image is interpolated



How to enforce convexity?



Generator: Minimize

$$\mathcal{L}_{Gen} = \|X - D(G(X))\|_1 + \|G(Z_{\text{interp}}) - D(G(Z_{\text{interp}}))\|_1$$

Discriminator: Minimize

$$\begin{aligned}\mathcal{L}_{Disc} = & \|X - D(X)\|_1 + \\ & - \|G(X) - D(G(X))\|_1 + \\ & - \|G(Z_{\text{interp}}) - D(G(Z_{\text{interp}}))\|_1\end{aligned}$$

Regularize, for each mini-batch:

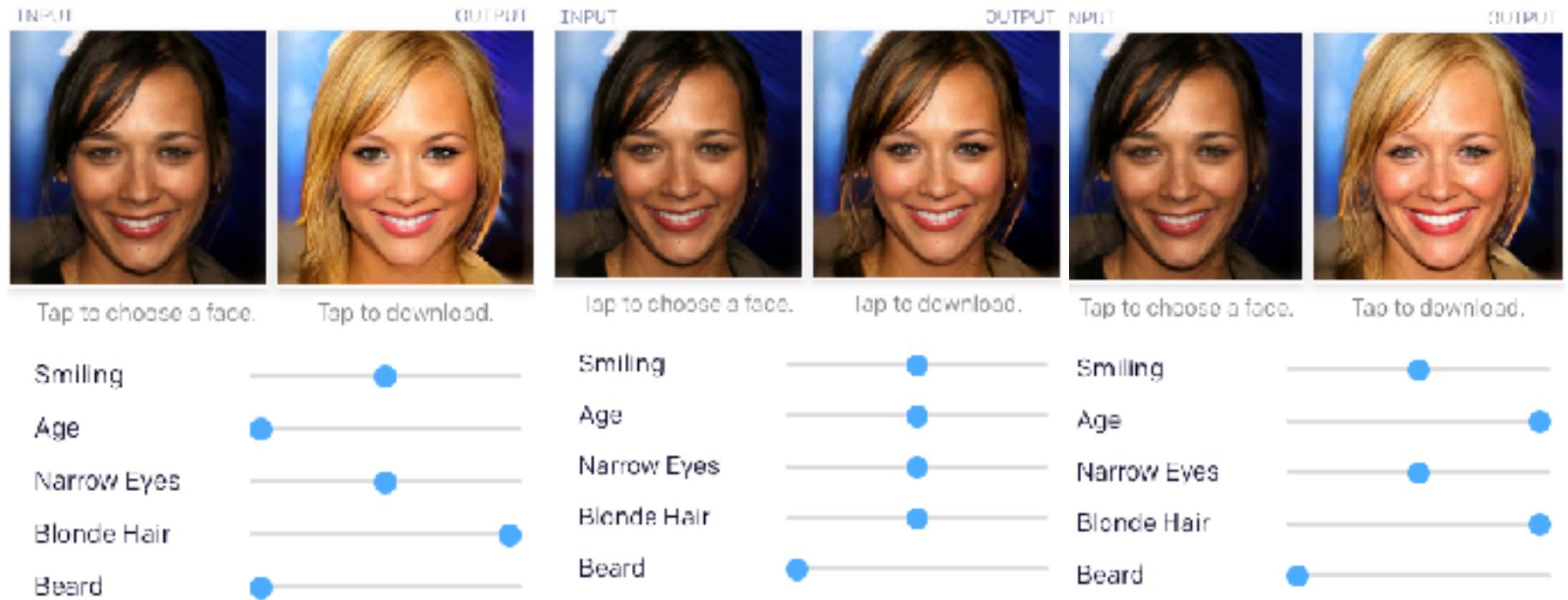
$$\begin{aligned}\mathcal{L}_{dist}(X, Z) = & \frac{1}{B} \sum_{i,j}^B \left[\log_2 \left(1 + \frac{(X_i - X_j)^2}{\frac{1}{N} \sum_{i,j} (X_i - X_j)^2} \right) \right. \\ & \left. - \log_2 \left(1 + \frac{(Z_i - Z_j)^2}{\frac{1}{N} \sum_{i,j} (Z_i - Z_j)^2} \right) \right]\end{aligned}$$



Results



GLOW



Results

