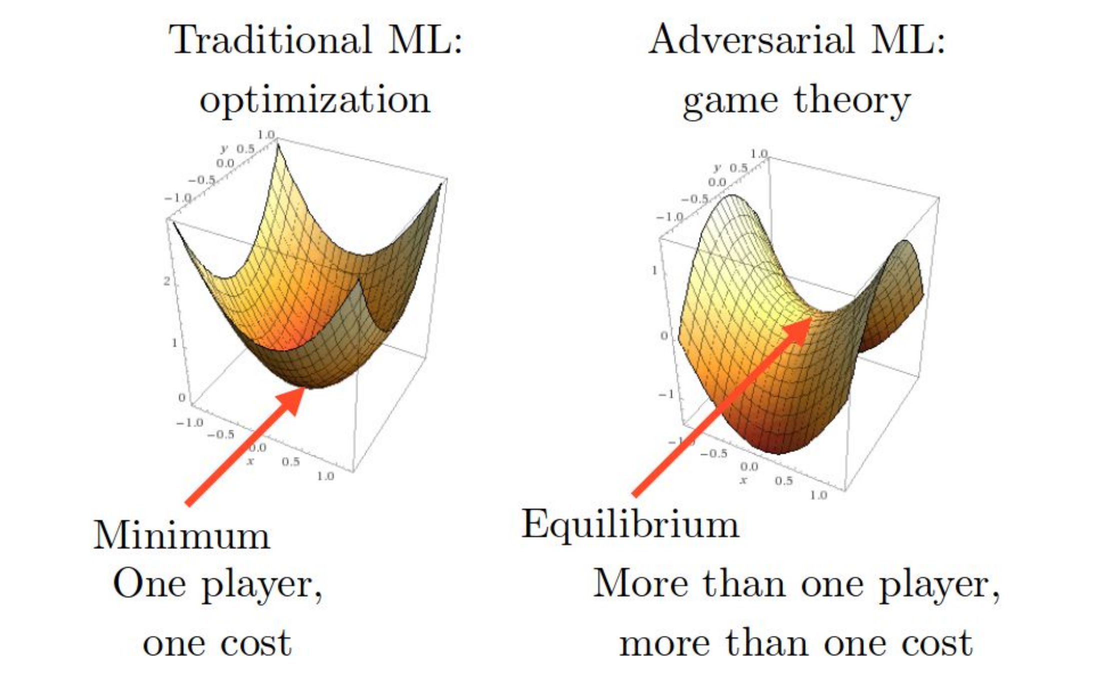
Adversarial Machine Learning

with a focus on GANs

Maria Rigaki
maria.rigaki@aic.fel.cvut.cz

Definition

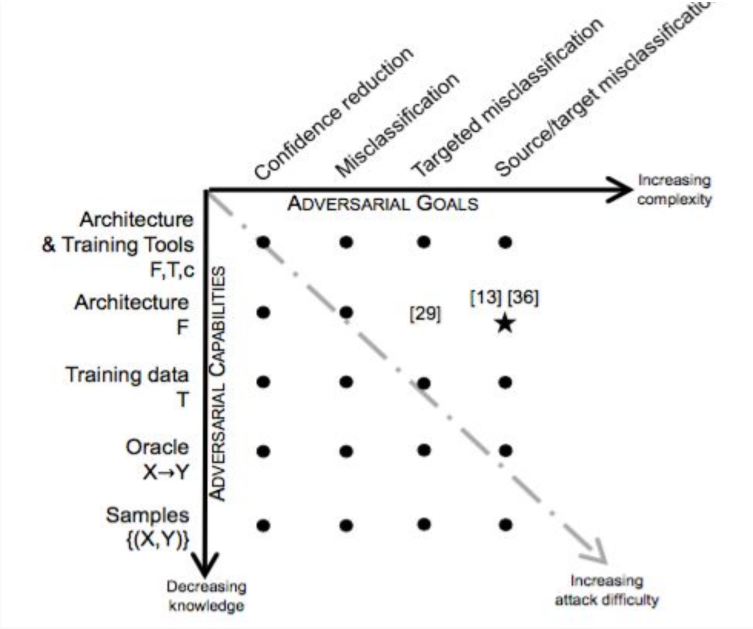
- Learning in the face of adversaries
- Two entities with their own cost functions

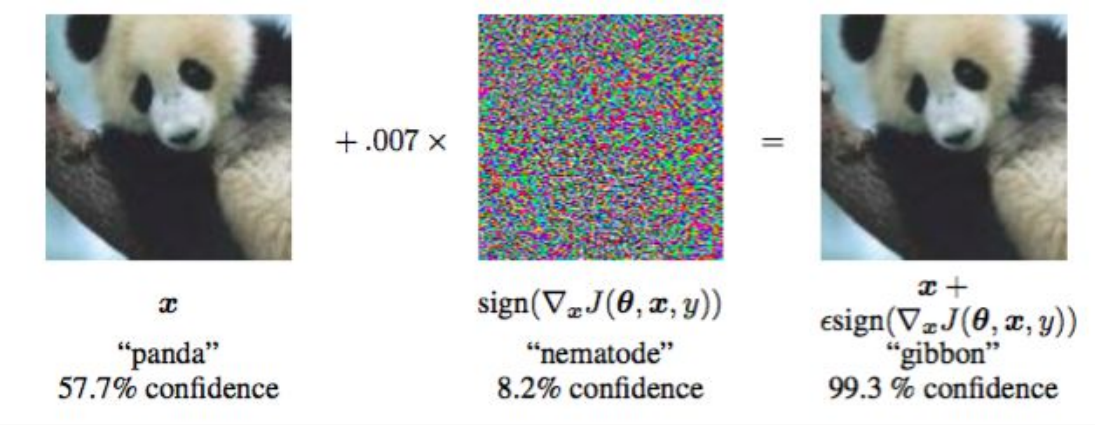


Security of Machine Learning

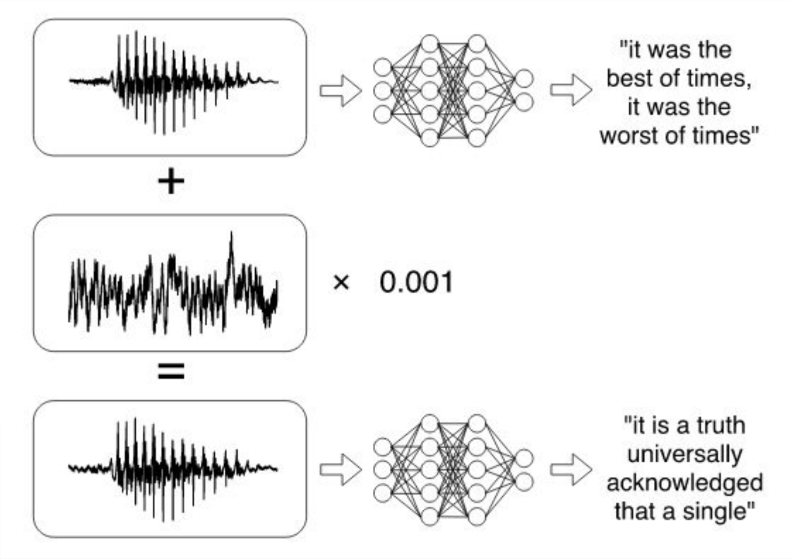
- Not a new thing but revived the past few years
- Attacks against ML models
- During training or testing
- Black box vs white box attacks

Attacker's Capability / **Attacker's Goal**	Misclassifications that do not compromise normal system operation	Misclassifications that compromise normal system operation	Querying strategies that reveal confidential information on the learning model or its users
	Integrity	**Availability**	**Privacy / Confidentiality**
Test data	Evasion (a.k.a. adversarial examples)	-	Model extraction / stealing and model inversion (a.k.a. hill-climbing attacks)
Training data	Poisoning (to allow subsequent intrusions) – e.g., backdoors or neural network trojans	Poisoning (to maximize classification error)	-



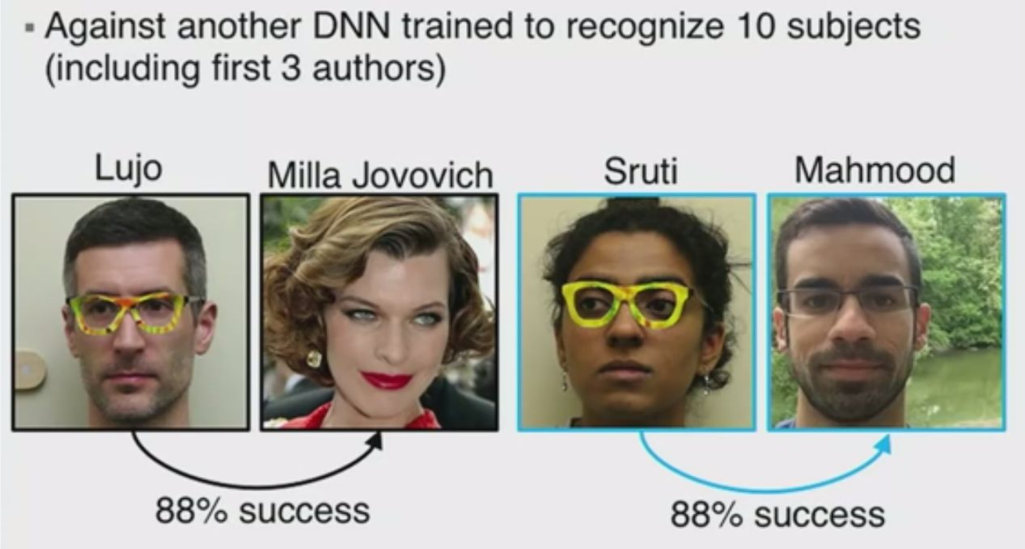


(Kurakin et al., 2014)

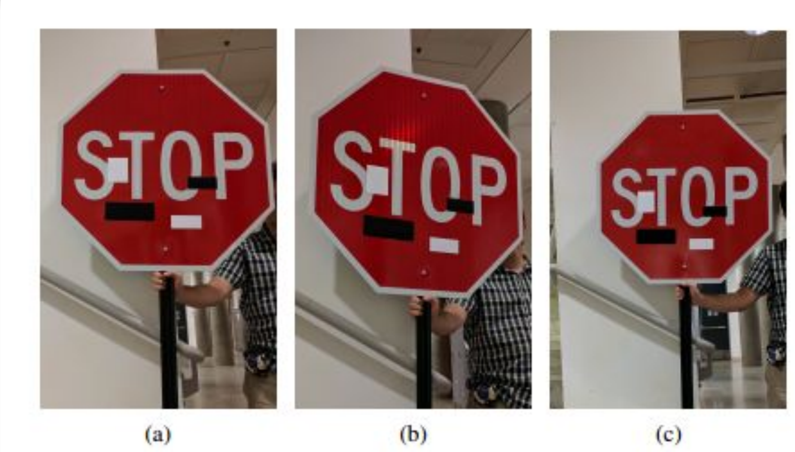


(Carlini & Wagner, 2018)

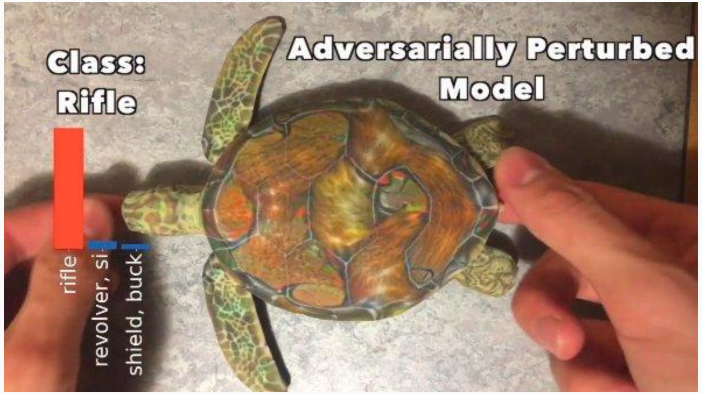
Physical Security



(Sharif et al. 2016)



(Evtimov et al. 2017)



(Athalye, A. and Sutskever, I., 2017)

Other topics

- Privacy (model leaks info about training data)
- Reinforcement Learning (Security, Self-play)
- Safety (self-driving cars, etc.)
- ...

Generative Adversarial Networks (GANs)

Motivation

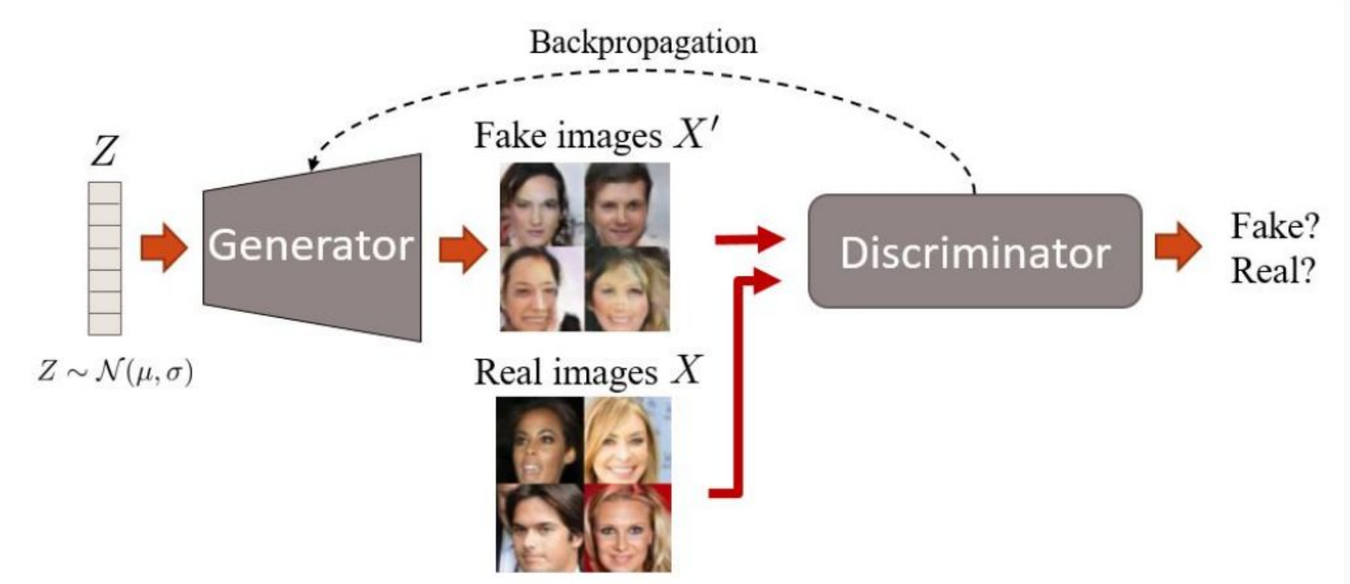
Sample Generation

CelebA dataset

(Karras et al, 2017)

Definition

A game between two neural networks



http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them

Mathematical Formulation

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

- Training the two networks until “equilibrium”
- Wanted equilibrium is “Generator wins”, i.e. the discriminator cannot tell apart the samples from $\boldsymbol{P}_{data}$ and $\boldsymbol{P}_{fake}$
- Not necessarily ***log()***

Loss Functions

Vanilla GAN:

Discriminator loss function:

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

Generator loss function:

$$J^{(G)} = -J^{(D)}$$

In practice:

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

Training algorithm

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

for number of training iterations **do**
  for k steps **do**
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

Code example (keras)

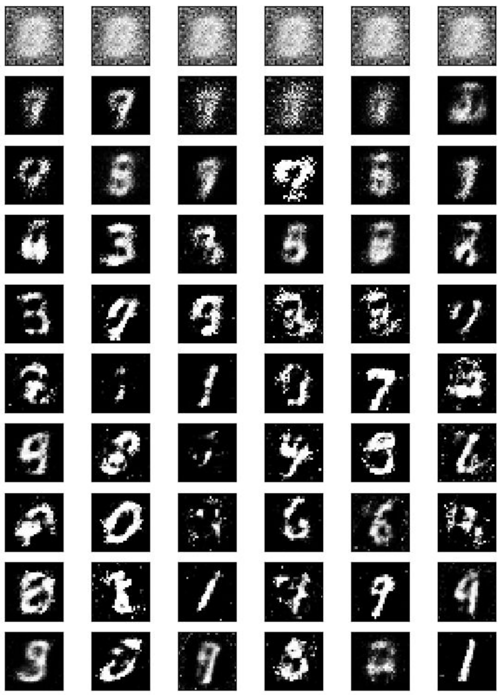
```
# Build and compile the discriminator
self.discriminator = self.build_discriminator()
self.discriminator.compile(loss='binary_crossentropy',
    optimizer=optimizer,
    metrics=['accuracy'])

# Build the generator
self.generator = self.build_generator()

# The generator takes noise as input and generates imgs
z = Input(shape=(self.latent_dim,))
img = self.generator(z)

# For the combined model we will only train the generator
self.discriminator.trainable = False

# The discriminator takes generated images as input and determines validity
valid = self.discriminator(img)

# The combined model  (stacked generator and discriminator)
# Trains the generator to fool the discriminator
self.combined = Model(z, valid)
self.combined.compile(loss='binary_crossentropy', optimizer=optimizer)
```

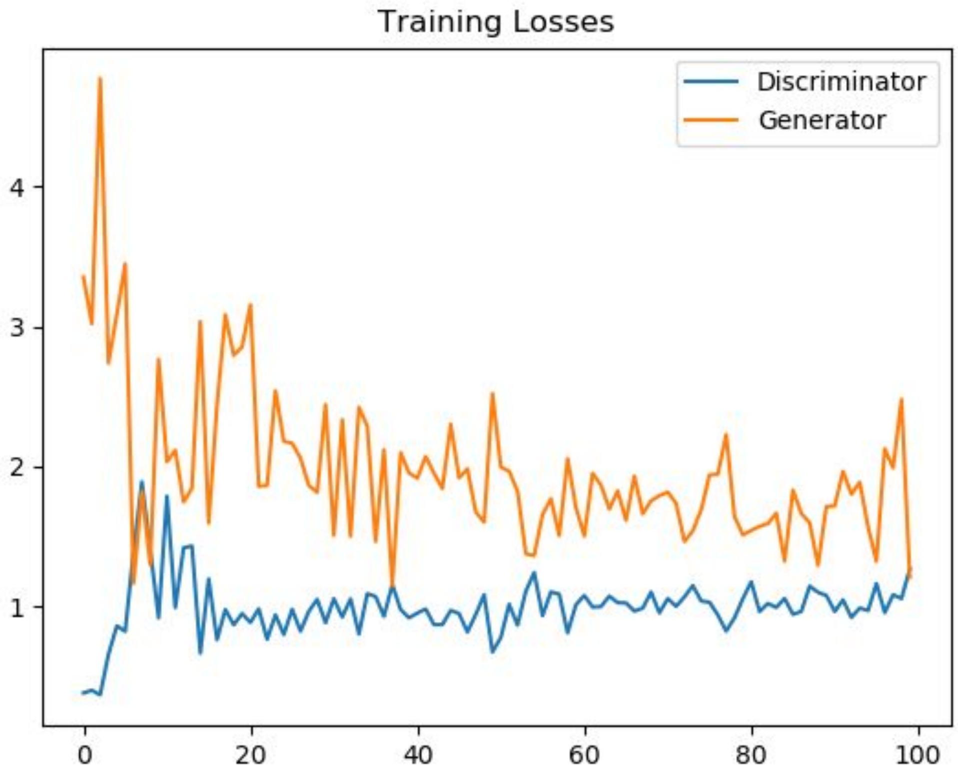
Define discriminator

Define generator

Define stacked combined model but freeze the discriminator parameters

Training results





LSGAN

Least squares GAN (LSGAN):

$$J^{(D)} = \frac{1}{2m}\sum_{i=1}^{m}\left[\left(D(x^{(i)}) - 1\right)^2\right] + \frac{1}{2m}\sum_{i=1}^{m}\left[\left(D(G(z^{(i)}))\right)^2\right]$$

$$J^{(G)} = \frac{1}{m}\sum_{i=1}^{m}\left[\left(D(G(z^{(i)})) - 1\right)^2\right]$$

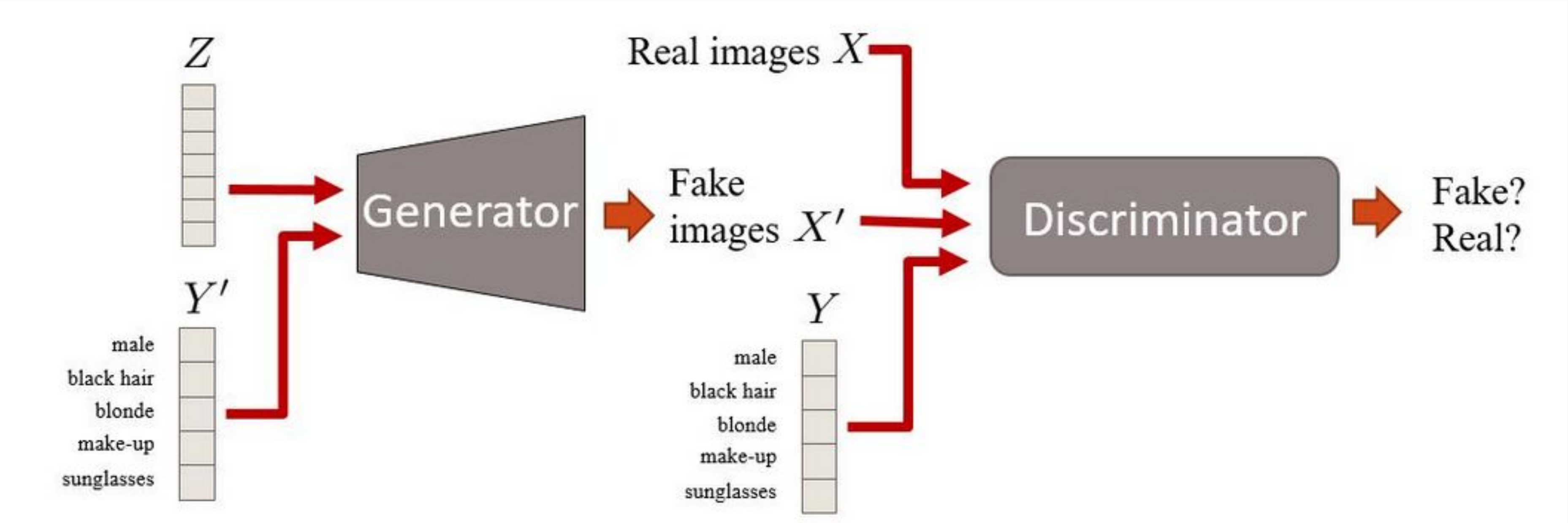
Hinge Loss

$$V_D(\hat{G}, D) = \underset{\boldsymbol{x} \sim q_{\text{data}}(\boldsymbol{x})}{\mathrm{E}} \left[\min\left(0, -1 + D(\boldsymbol{x})\right)\right] + \underset{\boldsymbol{z} \sim p(\boldsymbol{z})}{\mathrm{E}} \left[\min\left(0, -1 - D\left(\hat{G}(\boldsymbol{z})\right)\right)\right]$$

$$V_G(G, \hat{D}) = - \underset{\boldsymbol{z} \sim p(\boldsymbol{z})}{\mathrm{E}} \left[\hat{D}\left(G(\boldsymbol{z})\right)\right],$$

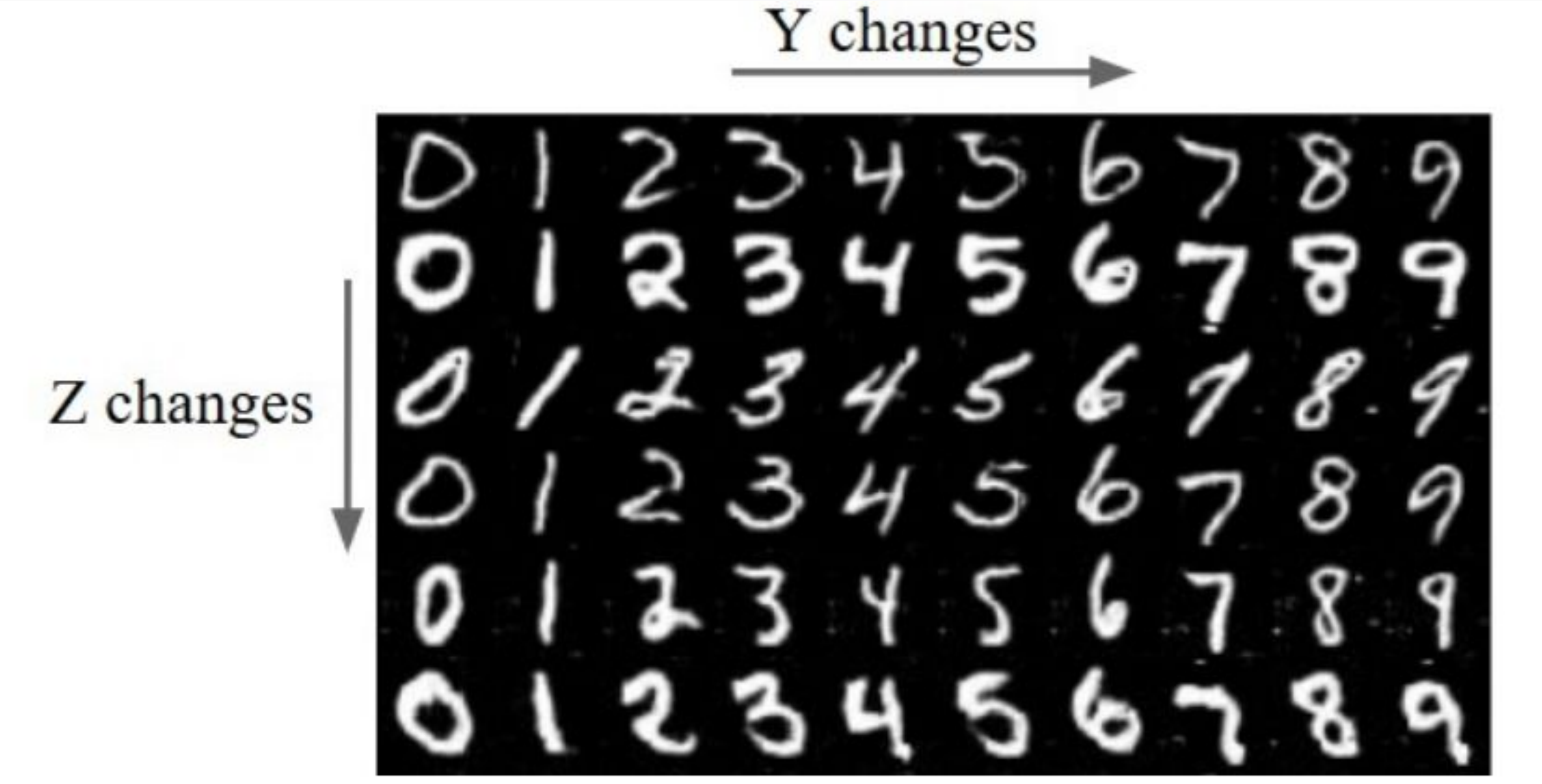
(Miyato et al 2017, Lim and Ye 2017, Tran et al 2017)

Conditional GAN

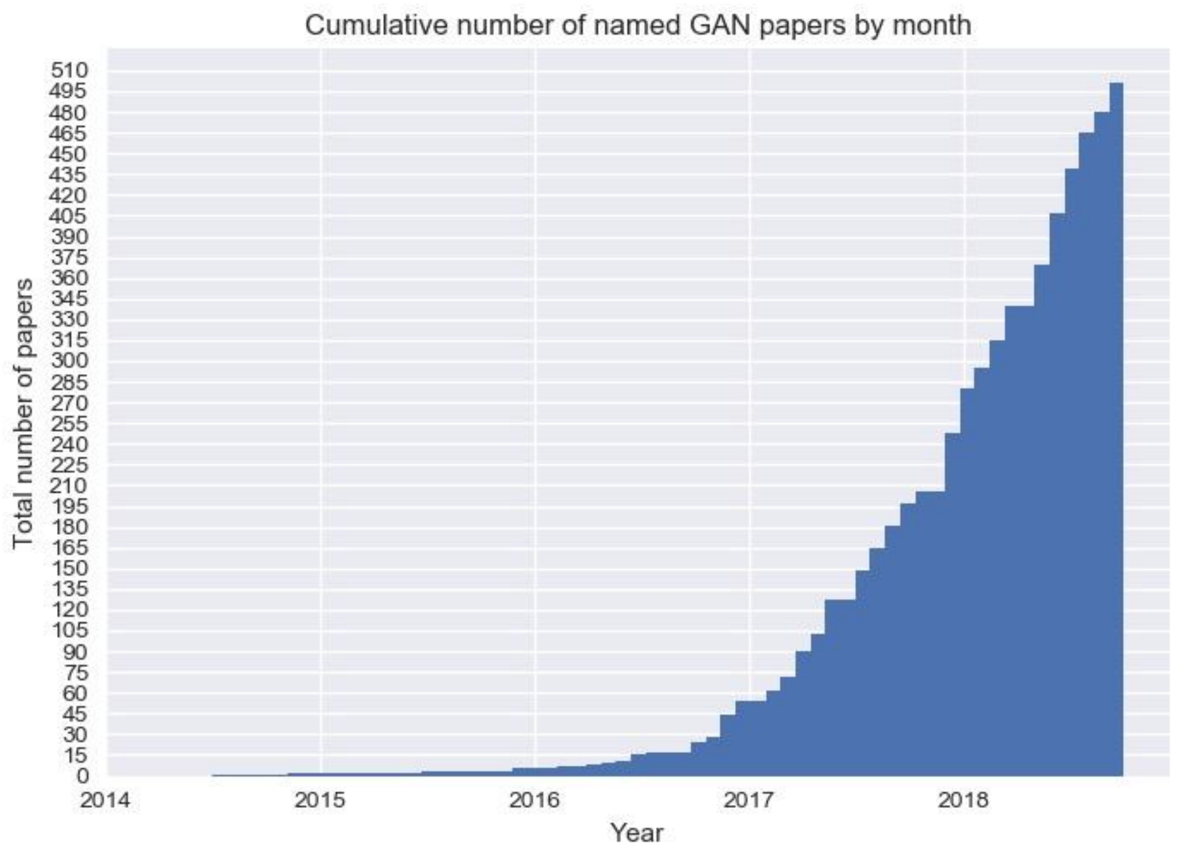


(Mizra et al. 2014)

cGAN (2)

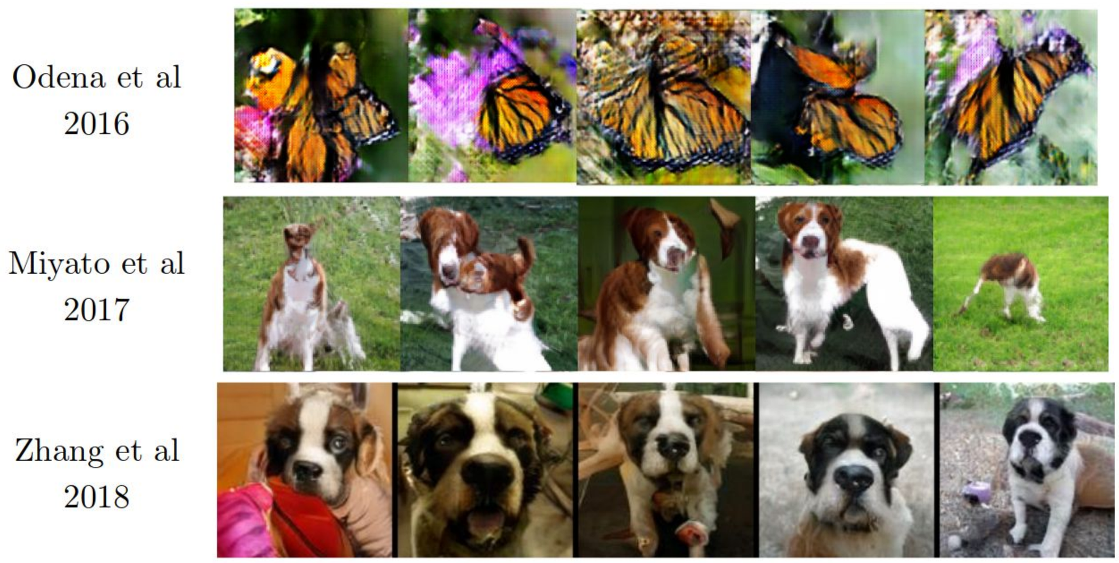

Y changes
Z changes
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

A GAN explosion



Progress over time

(Brundage et al. 2017)



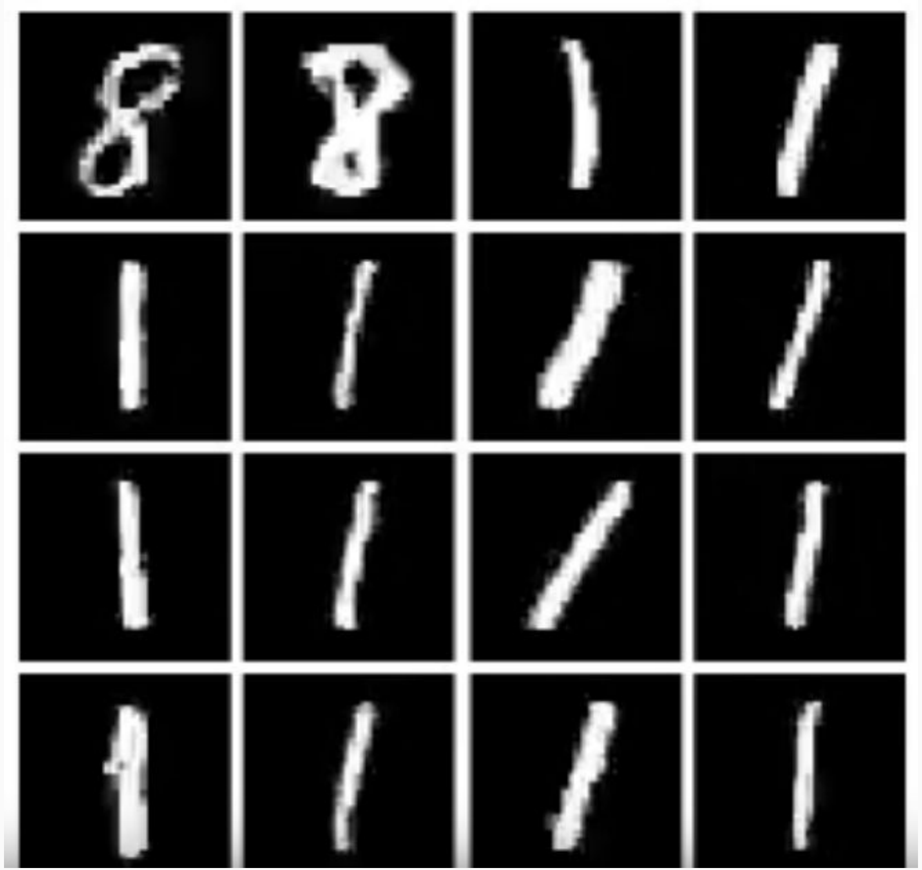
How well do GANs work?

Convergence vs quality

- No correlation between quality and convergence (in most GANs)
- Frequently we observe oscillations between the two loss functions
- How do we measure the quality of the generated data?

Mode Collapse

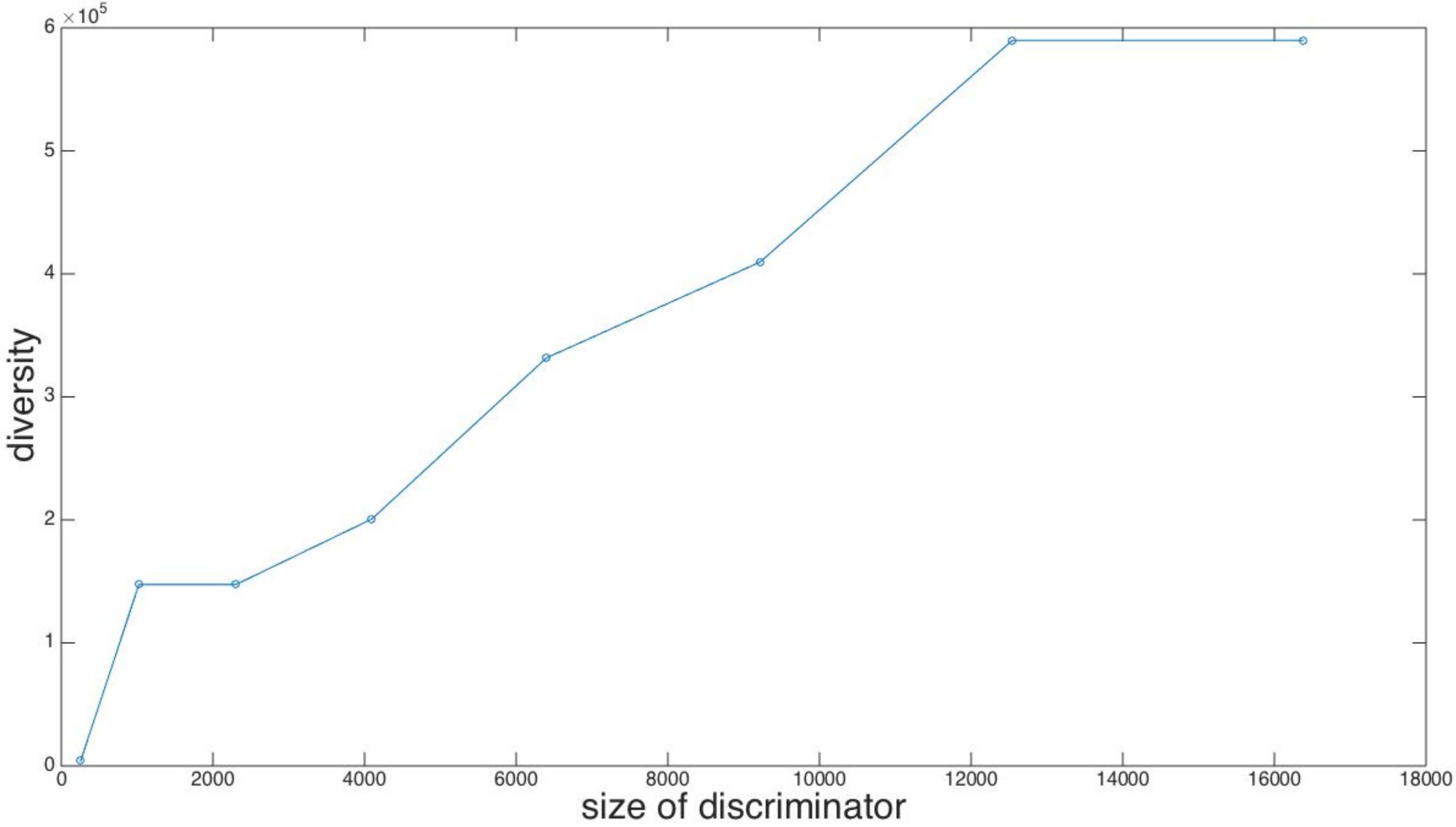
Sometimes a Generator generates data from a limited subset of the distribution



Do GANs actually learn the distribution?

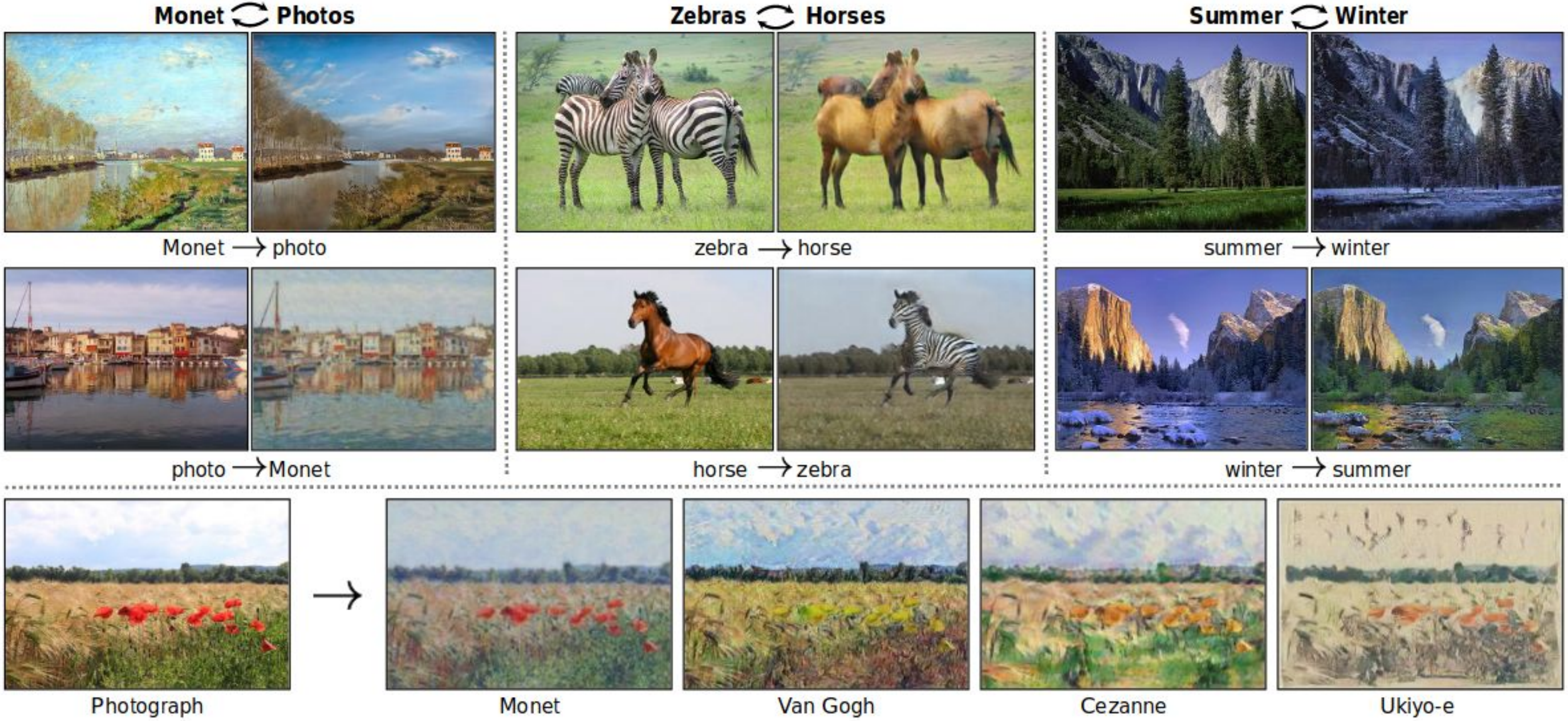
(Arora et al. 2017)

- Suppose the generator wins. What does that say about whether or not $\boldsymbol{P_{data}}$ is close to $\boldsymbol{P_{fake}}$?
- Original belief: "*All is well if the nets, the training data and the training time are large*" - Ian Goodfellow
- Unfortunately: if ***D*** has size ***N***, then $\exists$ ***G*** that generates a distribution supported by *O(NlogN)* images and still wins against all possible discriminators
- In other words: GANs training objective not guaranteed to avoid mode-collapse (generator can "win" using distributions of low support)

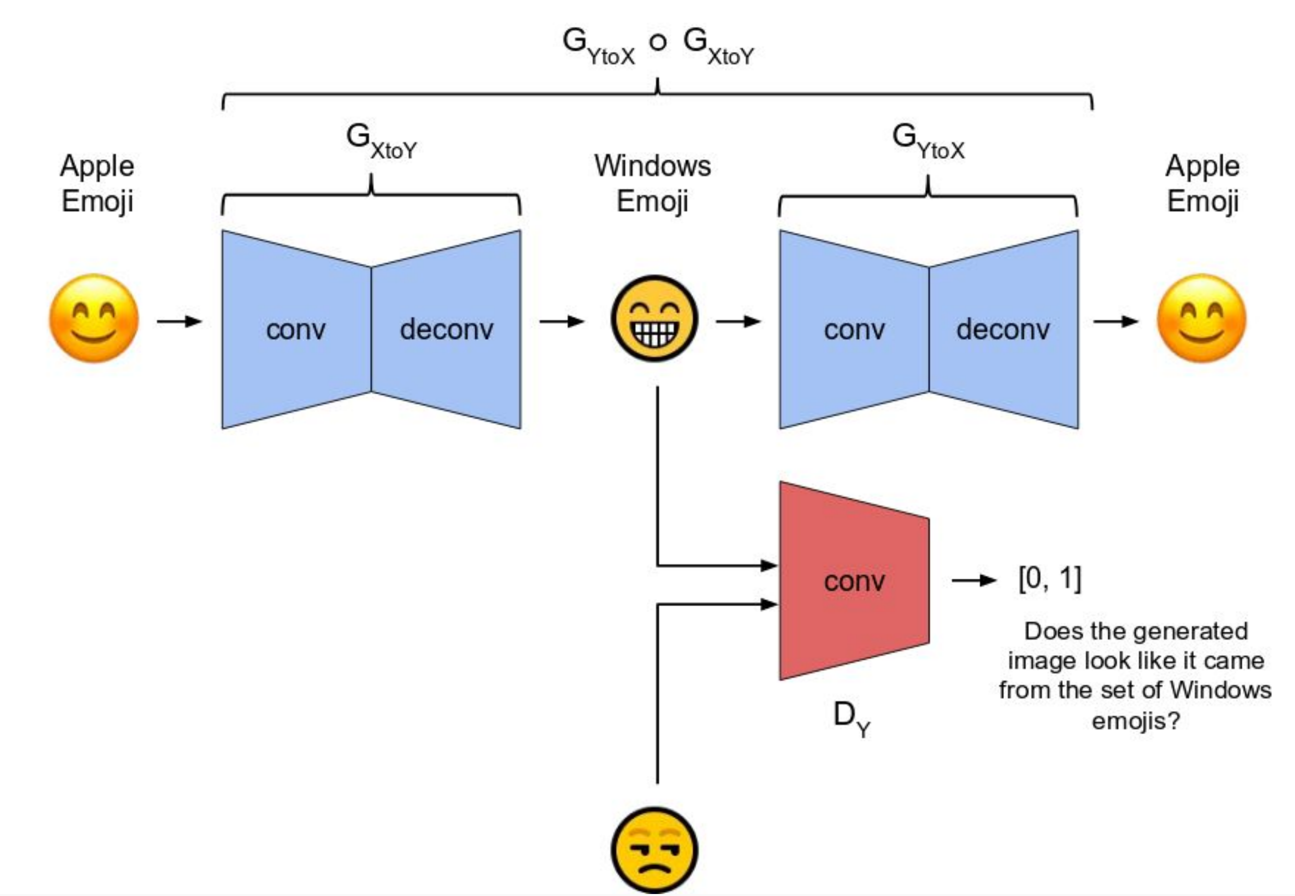

×10^5
6
5
4
3
2
1
0
diversity
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
size of discriminator

Interesting applications of GANs

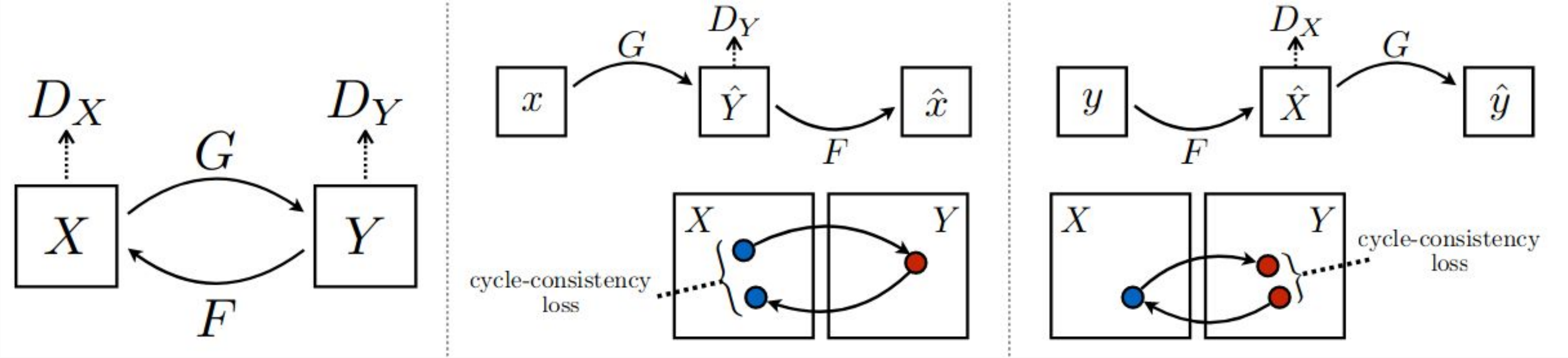
CycleGAN (unpaired image-to-image translation)



CycleGAN architecture

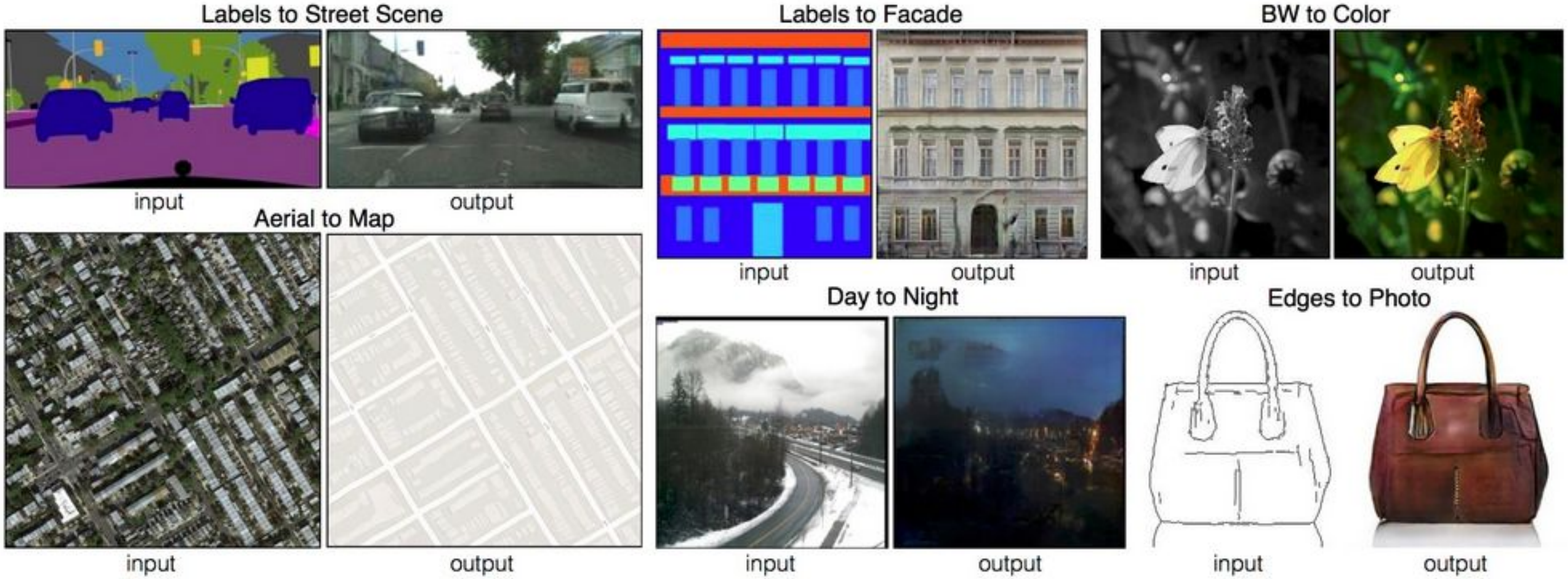


Cycle GAN architecture (2)



Failure case

(Paired) Image to image translation



(Isola et al. 2017)

And more...

- Music generation
- Text to image
- Super resolution images

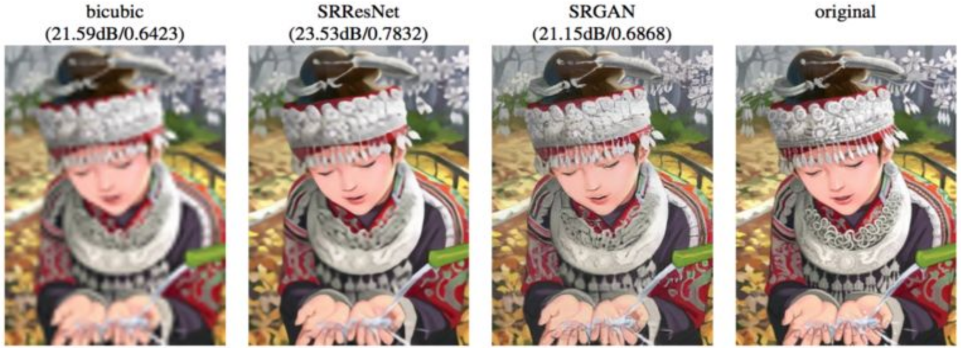


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

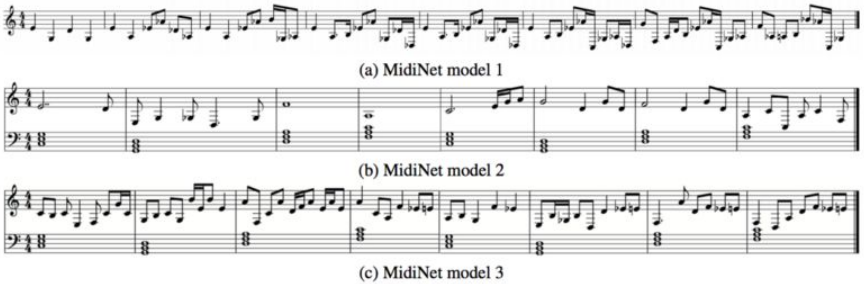


Figure 3. Example result of the melodies (of 8 bars) generated by different implementations of MidiNet.

Interesting links

GAN zoo - https://github.com/hindupuravinash/the-gan-zoo

GAN implementations in keras - https://github.com/eriklindernoren/Keras-GAN

Off the convex path blog - http://www.offconvex.org (Arora et al.)

GAN playground - https://reiinakano.github.io/gan-playground/

References

Arora, S., Ge, R., Liang, Y., Ma, T. and Zhang, Y., 2017. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573.*

Athalye, A. and Sutskever, I., 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*.

Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., Dafoe, A., Scharre, P., Zeitzoff, T., Filar, B. and Anderson, H., 2018. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228.*

Carlini, N.and Wagner, D., 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, (pp. 1-7).

Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A. and Song, D., 2017. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

References (2)

Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks. *arXiv preprint*.

Karras, T., Aila, T., Laine, S. and Lehtinen, J., 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Kurakin, A., Goodfellow, I. and Bengio, S., 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Mirza, M. and Osindero, S., 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Sharif, M., Bhagavatula, S., Bauer, L. and Reiter, M.K., 2016, October. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1528-1540). ACM.

Zhu, J.Y., Park, T., Isola, P. and Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*.

Q & A