Introduction to Deep Learning (IT3320E)

10 - Language Models

Hung Son Nguyen

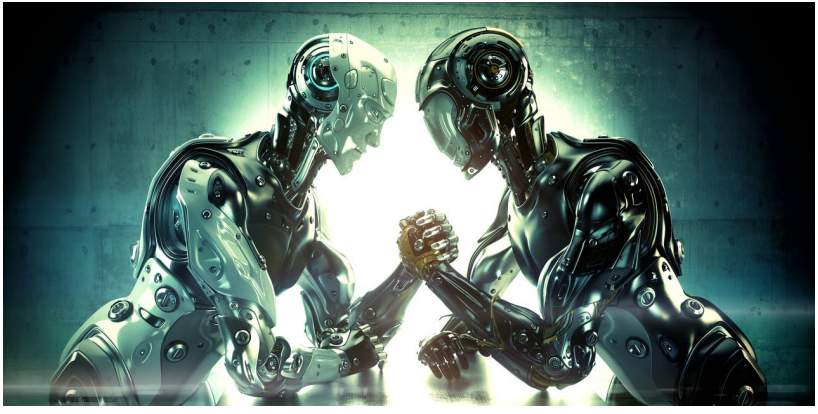
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Nov. 22, 2022

Agenda

1. Generative adversarial networks (GANs)
   - Vanilla GAN
   - WGAN
2. Training GANs
3. Boundary equilibrium GAN (BEGAN)
4. Adversarial examples

What Is Adversarial?

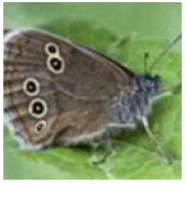


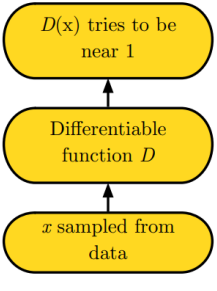
- Generative adversarial networks (GANs)

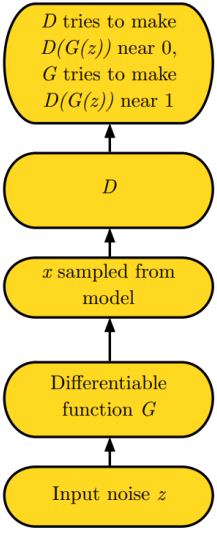
Section 1

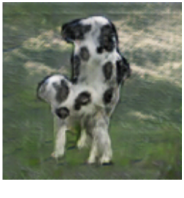
Generative adversarial networks (GANs)

Adversarial Nets Framework









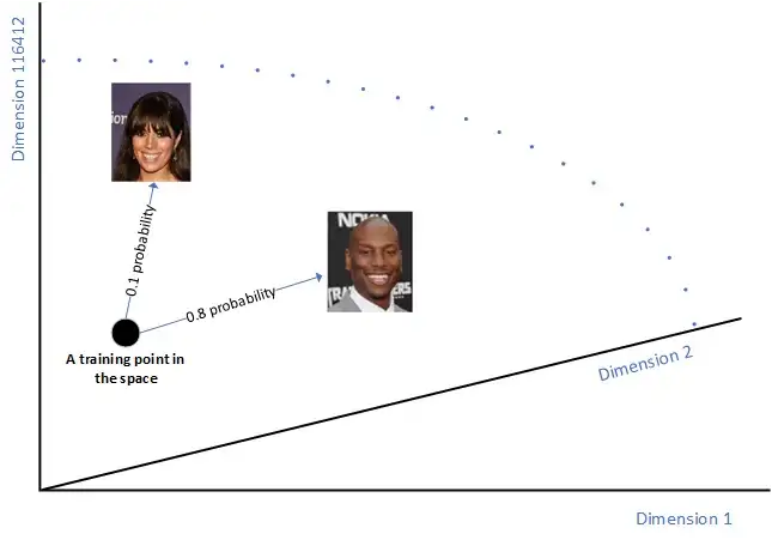
(Goodfellow et al., 2014)

- Generating new data is a game of probabilities. When we are observing the world around us and collecting data, we are performing an experiment. A simple example is taking a photo of a celebrity's face.
- This can be considered as a probabilistic experiment, with an unknown outcome X, also called a random variable.
- If the experiment is repeated many times, we usually define the probability that the random variable X will get the value little x, as the fraction of times that little x occurs
- Therefore we can consider probability as a function that takes an outcome, i.e. an element from the sample space (a photo) and maps the outcome to a non-negative real number so that the sum of all these numbers equals 1.
- We also call this a probability distribution function $P(X)$. When we know the sample space (all possible celebrity faces) and the probability distribution (the probability of occurrence of each face), we have the full description of the experiment and we can reason about uncertainty.

A celebrity-face probability distribution

- Generating new faces can be expressed by a random variable generation problem. The face is described by random variables, represented through its RGB values, flatten into a vector of N numbers.
- The celebrity faces are $218px$ height, $178px$ width with 3 color channels. Therefore each vector is 116412-dimensional.
- If we build a space with $116412(N)$ axes, each face will be a point in that space. A celebrity-face probability distribution function $P(X)$ would map each face to a non-negative real number so that the sum of all these numbers for all faces equals 1.
- Some points of that space are very likely to represent celebrity faces whereas it is highly unlikely for some others.
- A GAN generates a new celebrity face by generating a new vector following the celebrity face probability distribution over the N-dimensional vector space.

A celebrity-face probability distribution

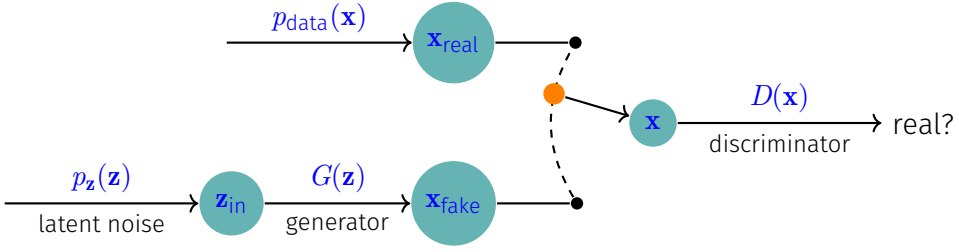


In simple words, a GAN would generate a random variable with respect to a specific probability distribution.

How to generate random variables from complex distributions?

- The celebrity-face probability distribution over the N-dimensional vector space is a very complex one and we don't know how to directly generate complex random variables.
- Luckily, we can represent our complex random variable by a function applied to a uniform random variable. This is the idea of the transform method.
- It first generates N uncorrelated uniform random variables, which is easy. It then applies a very complex function to that simple random variable! Very complex functions are naturally approximated by a neural network.
- After training the network will be able to take as input a simple N-dimensional uniform random variable and return another N-dimensional random variable that would follow our celebrity-face probability distribution. This is the core motivation behind generative adversarial networks.

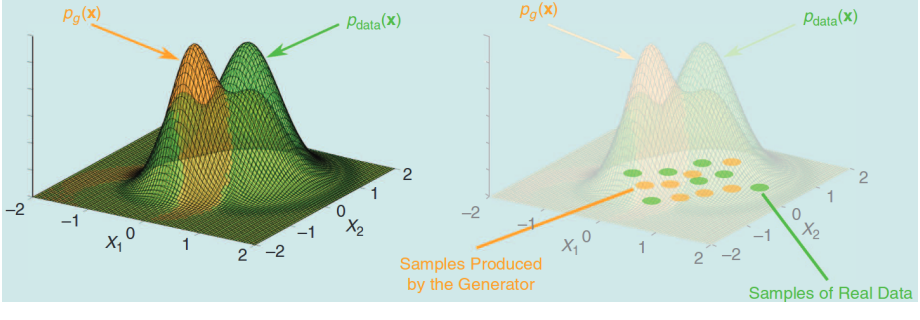
- Generator G: capture the data distribution (make realistic images)
- Discriminator D: estimate the probability that a sample came from the training data rather than G (tell real and fake images apart)



- $p_{\mathbf{z}}(\mathbf{z})$: input noise
- $p_{\text{data}}(\mathbf{x})$: real data's distribution
- $p_g(\mathbf{x})$: generator's distribution of $G(\mathbf{z})$

Two-player minimax game

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



Now comes the hard and slow part: training a generative adversarial network. Because a GAN consists of two separately trained networks, convergence is hard to identify.

The following steps are executed back and forth allowing GANs to tackle otherwise intractable generative problems.

Step 1 — Select a number of real images from the training set.

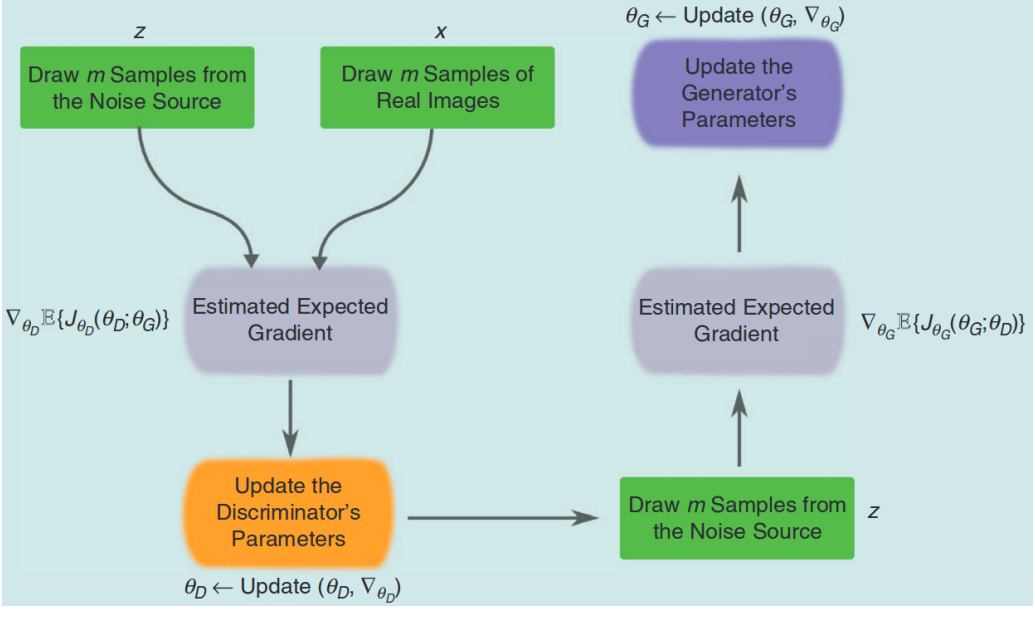
Step 2 — Generate a number of fake images. This is done by sampling random noise vectors and creating images from them using the generator.

Step 3 — Train the discriminator for one or more epochs using both fake and real images. This will update only the discriminator's weights by labeling all the real images as 1 and the fake images as 0.

Step 4 — Generate another number of fake images.

Step 5 — Train the full GAN model for one or more epochs using only fake images. This will update only the generator's weights by labeling all fake images as 1.

Main loop of GAN training



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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

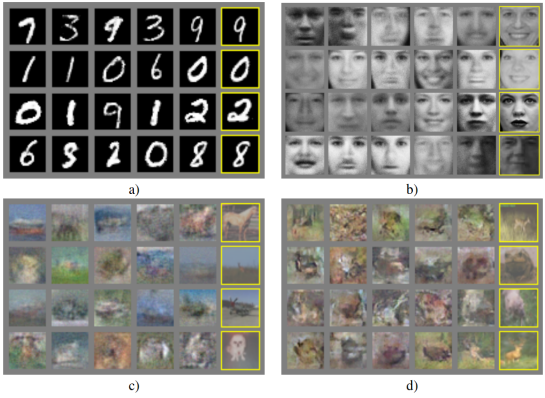
end for

- 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.



Difficult to train **Salimans2016**; **Arjovsky2017a**

- Vanishing gradients
- Instability
- Model collapse
- ...

17 tips to make GANs work (*https://github.com/soumith/ganhacks*)

- Use a spherical $\mathbf{z}$
- Batch normalization **Ioffe2015**
- ...

GAN variants (> 100)

- Deep convolutional GAN (DCGAN) **Radford2015**
- Conditional GAN **Mirza2014**
- Adversarially learned inference (ALI) **Dumoulin2016**
- Adversarial autoencoder (AAE) **Makhzani2015**
- Energy-based GAN (EBGAN) **Zhao2016**
- Wasserstein GAN (WGAN) **Arjovsky2017b**
- Boundary equilibrium GAN (BEGAN) **Berthelot2017**
- Bayesian GAN **Saatchi2017**
- ...

Recall GAN

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

Earth-Mover (EM) distance or Wasserstein-1 **Monge1781**

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

By Kantorovich-Rubinstein duality **Villani2008**,

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L\leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

Discriminator f_w, generator g_θ

$$\min_\theta \max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$$

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

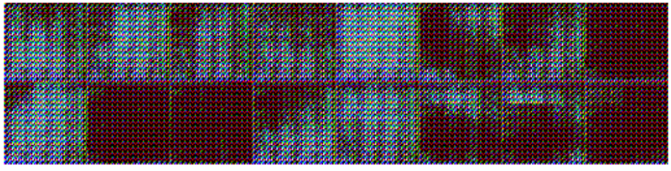
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

- Improved stability of learning
- Get rid of mode collapse
- Meaningful learning curves

GAN without tricks during training

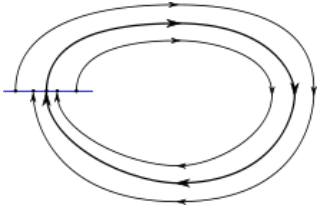


WGAN samples

Section 2

Training GANs

Limit cycling behavior in training (W)GAN



[*https://en.wikipedia.org/wiki/Limit_cycle*]

GD

$$w_{t+1} = w_t + \eta \cdot \nabla_{w,t}$$

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_{\theta,t}$$

OMD **Rakhlin2013**: $M_{\cdot,t+1}$ is a predictor of $\nabla_{\cdot,t}$

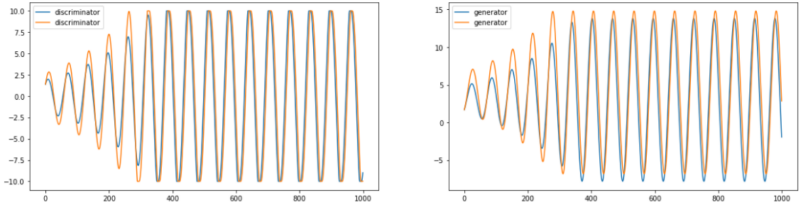
$$w_{t+1} = w_t + \eta \cdot (\nabla_{w,t} + M_{w,t+1} - M_{w,t})$$

$$\theta_{t+1} = \theta_t - \eta \cdot (\nabla_{\theta,t} + M_{\theta,t+1} - M_{\theta,t})$$

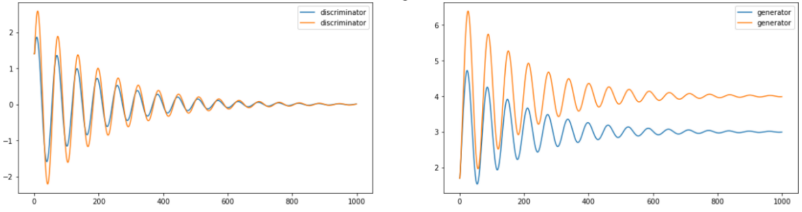
In this paper, choose $M_{\cdot,t+1} = \nabla_{\cdot,t}$

$$w_{t+1} = w_t + \eta \cdot (2\nabla_{w,t} - \nabla_{w,t-1}) = w_t + 2\eta \cdot \nabla_{w,t} - \eta \cdot \nabla_{w,t-1}$$

$$\theta_{t+1} = \theta_t - \eta \cdot (2\nabla_{\theta,t} - \nabla_{\theta,t-1}) = \theta_t - 2\eta \cdot \nabla_{\theta,t} + \eta \cdot \nabla_{\theta,t-1})$$



(a) GD dynamics.



(b) OMD dynamics.

OMD dynamics converge in terms of the last iterate.

ADAM (adaptive moment estimation) **Kingma2014** (126165 citations)

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

ADAM:

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where $\hat{m}_t$ is first moment, $\hat{v}_t$ is second moment

Algorithm 1 *Optimistic ADAM*, proposed algorithm for training WGANs on images.

Parameters: stepsize η, exponential decay rates for moment estimates $\beta_1, \beta_2 \in [0, 1)$, stochastic loss as a function of weights $\ell_t(\theta)$, initial parameters θ_0

for each iteration $t \in \{1, \ldots, T\}$ **do**

Compute stochastic gradient: $\nabla_{\theta,t} = \nabla_\theta \ell_t(\theta)$

Update biased estimate of first moment: $m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot \nabla_{\theta,t}$

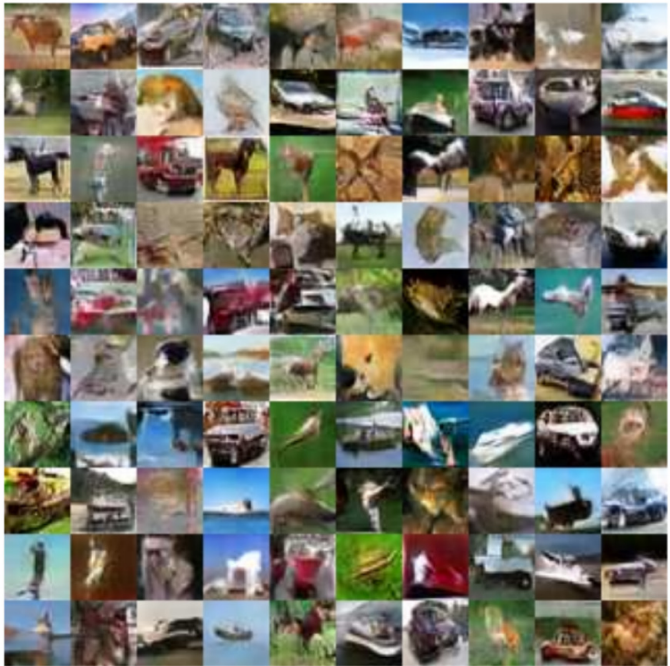
Update biased estimate of second moment: $v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot \nabla^2_{\theta,t}$

Compute bias corrected first moment: $\hat{m}_t = m_t / (1 - \beta_1^t)$

Compute bias corrected second moment: $\hat{v}_t = v_t / (1 - \beta_2^t)$

Perform *optimistic gradient step*: $\theta_t = \theta_{t-1} - 2\eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \eta \frac{\hat{m}_{t-1}}{\sqrt{\hat{v}_{t-1}} + \epsilon}$

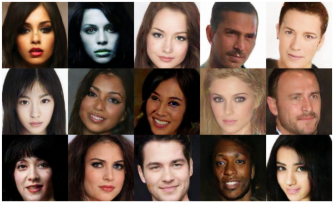
Return θ_T



Section 3

Boundary equilibrium GAN (BEGAN)

Samples

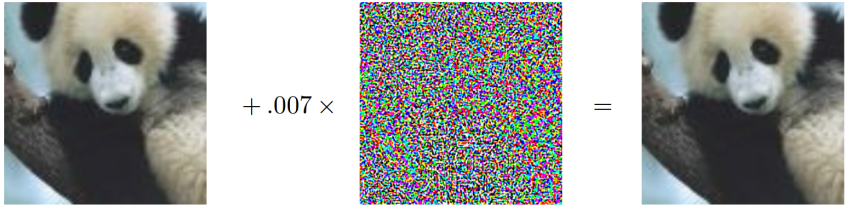


Interpolations of real images

Section 4

Adversarial examples

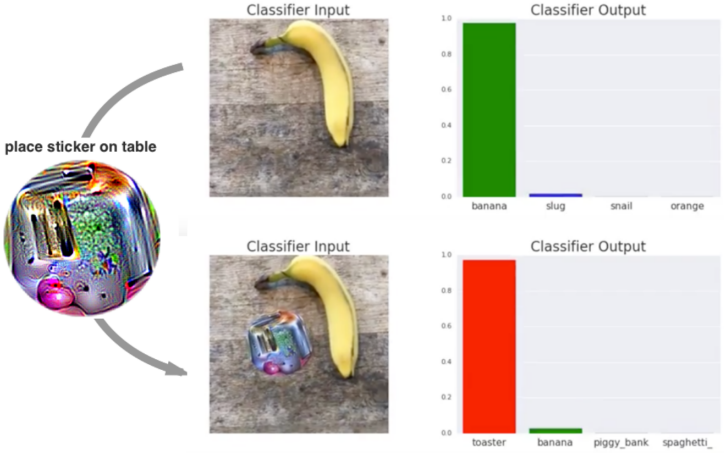
- Examples that are similar to examples in the true distribution, but that fool a classifier **Szegedy2013**
- A demonstration of adversarial example **Goodfellow2014a**



Paper review: Ilyas, The Robust Manifold Defense: Adversarial Training using Generative Models, 2017.

Art?

- Why small changes?
- Universal, robust, targeted adversarial image patches in the real world Brown2017
- *https://youtu.be/i1sp4X57TL4*



[Szegedy, 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013).

Intriguing properties of neural networks.

arXiv preprint arXiv:1312.6199.

[Goodfellow, 2014a] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014).

Explaining and harnessing adversarial examples.

arXiv preprint arXiv:1412.6572.

[Brown, 2017] Brown, T. B., Mané, D., Roy, A., Abadi, M., & Gilmer, J. (2017).

Adversarial patch.

arXiv preprint arXiv:1712.09665.

[Goodfellow, 2014b] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014).

Generative adversarial nets.

Advances in neural information processing systems, 2672–2680.

[Salimans, 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016).

Improved techniques for training gans.

Advances in Neural Information Processing Systems, 2234–2242.

[Arjovsky, 2017a] Arjovsky, M., & Bottou, L. (2017).

Towards principled methods for training generative adversarial networks.

arXiv preprint arXiv:1701.04862.

[Ioffe, 2015] Ioffe, S., & Szegedy, C. (2015).

Batch normalization: Accelerating deep network training by reducing internal covariate shift.

International conference on machine learning, 448–456.

[Denton, 2015] Denton, E. L., Chintala, S., & Fergus, R. (2015).

Deep generative image models using a laplacian pyramid of adversarial networks.

Advances in neural information processing systems, 1486–1494.

[Radford, 2015] Radford, A., Metz, L., & Chintala, S. (2015).

Unsupervised representation learning with deep convolutional generative adversarial networks.

arXiv preprint arXiv:1511.06434.

[Mirza, 2014] Mirza, M., & Osindero, S. (2014).

Conditional generative adversarial nets.

arXiv preprint arXiv:1411.1784.

[Dumoulin, 2016] Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., & Courville, A. (2016).

Adversarially learned inference.

arXiv preprint arXiv:1606.00704.

[Makhzani, 2015] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015).

Adversarial autoencoders.

arXiv preprint arXiv:1511.05644.

[Mescheder, 2017] Mescheder, L., Nowozin, S., & Geiger, A. (2017).

Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks.

arXiv preprint arXiv:1701.04722.

[Zhao, 2016] Zhao, J., Mathieu, M., & LeCun, Y. (2016).

Energy-based generative adversarial network.

arXiv preprint arXiv:1609.03126.

[Arjovsky, 2017b] Arjovsky, M., Chintala, S., & Bottou, L. (2017).

Wasserstein gan.

arXiv preprint arXiv:1701.07875.

[Berthelot, 2017] Berthelot, D., Schumm, T., & Metz, L. (2017).

Began: Boundary equilibrium generative adversarial networks.

arXiv preprint arXiv:1703.10717.

[Saatchi, 2017] Saatchi, Y., & Wilson, A. G. (2017).

Bayesian GAN.

Advances in Neural Information Processing Systems, 3625–3634.

[Monge, 1781] avec les Memoires de Mathematique et de Physique. 1781.

Memoire sur la theorie des deblais et des remblais.

Histoire de l'Academie Royale des Science, Annee 1781.

[Villani, 2008] Villani, C. (2008).

Optimal transport: old and new.

Springer Science & Business Media.

[Rakhlin, 2013] Rakhlin, S., & Sridharan, K. (2013).

Optimization, learning, and games with predictable sequences.

Advances in Neural Information Processing Systems, 3066–3074.

[Kingma, 2014] Kingma, D. P., & Ba, J. (2014).

Adam: A method for stochastic optimization.

arXiv preprint arXiv:1412.6980.

Thank you!