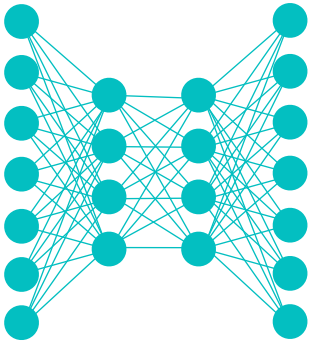
Lecture Notes for

Neural Networks and Machine Learning

Simple GAN and DCGAN



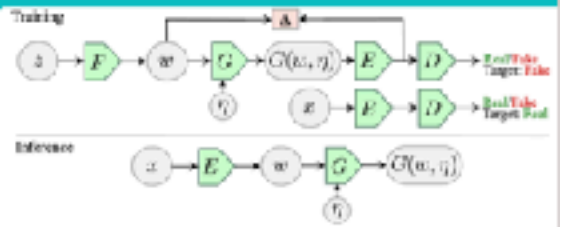
Logistics and Agenda

- Logistics
  - Student Presentations Today
  - Grading…
- Agenda
  - Review from Last Time
  - **Student Presentation**: Representational Learning with Deep Convolutional GANs, Radford
  - GAN Ticks and GAN Demo
  - Practical GANs
  - LS-GAN (Next Time)
  - Wasserstein GAN (Next time)
  - WGAN-GP (Next Time)
  - Big GAN (Next Next Time)

Last Time

Adversarial Latent Auto-Encoders, ALAE

$\sigma(x) = softplus = \log(1 + \exp(x))$ smoothed ReLU [0,∞)



E,D: Detect fake samples
minimize D for fake samples

E,D: Detect real samples
min -D (max D) for real samples

$$6: \mathcal{L}^{E,D}_{disc} = \sigma(D \circ E \circ G \circ F(z)) + \sigma(-D \circ E(x)) + \frac{\gamma}{2} \cdot \mathbf{E}[\|\nabla D \circ E(x)\|]$$

F,G: Try to fool discriminator,
min -D (max D) for fake samples

E,D: Gradient Penalty
Keep Gradient Magnitude Small
Keep in mind for later

$$10: \mathcal{L}^{F,G}_{gen} = \sigma(-D \circ E \circ G \circ F(z))$$

$$14: \mathcal{L}^{E,G}_{latent} = \|F(z) - E \circ G \circ F(z)\|^2$$

E,G: Keep latent spaces similar

Algorithm 1 ALAE Training

1: $\theta_F, \theta_G, \theta_E, \theta_D \leftarrow$ Initialize network parameters
2: **while** not converged **do**
3: Step I. Update E, and D
4: $x \leftarrow$ Random mini-batch from dataset
5: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
6: $L^{E,D}_{adv} \leftarrow \text{softplus}(D \circ E \circ G \circ F(z))) + \text{softplus}(-D \circ E(x)) + \frac{\gamma}{2} \mathbf{E}_{p_L(x)}[\|\nabla D \circ E(x)\|^2]$
7: $\theta_E, \theta_D \leftarrow \text{ADAM}(\nabla_{\theta_E, \theta_D} L^{E,D}_{adv}, \theta_D, \theta_E, \alpha, \beta_1, \beta_2)$
8: Step II. Update F, and G
9: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
10: $L^{F,G}_{adv} \leftarrow \text{softplus}(-D \circ E \circ G \circ F(z)))$
11: $\theta_F, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_F, \theta_G} L^{F,G}_{adv}, \theta_F, \theta_G, \alpha, \beta_1, \beta_2)$
12: Step III. Update E, and G
13: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
14: $L^{E,G}_{error} \leftarrow \|F(z) - E \circ G \circ F(z)\|^2_2$
15: $\theta_E, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_E, \theta_G} L^{E,G}_{error}, \theta_E, \theta_G, \alpha, \beta_1, \beta_2)$
16: **end while**

6: 10: Based On Wasserstein Distance, … which is really helpful…

Pidhorskyi, Stanislav, Donald A. Adjeroh, and Gianfranco Doretto. "Adversarial latent autoencoders." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14104-14113. 2020.

46

The Simple GAN

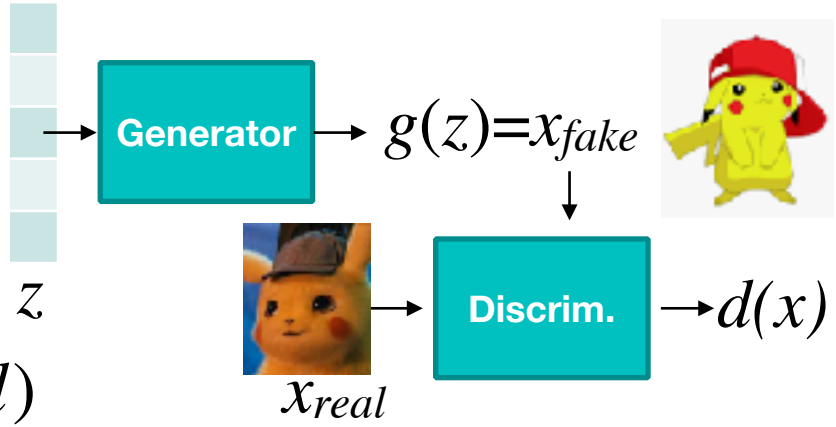
Every time Ian Goodfellow has a tutorial, It should be called:

"GAN-splaining with Ian"

—Geoffrey Hinton, Probably

Generative Adversarial Network

- Generator: $x = g(z)$
- Discriminator: $\{0,1\} = d(x)$
- Mini-max, turn-based game:



$$\hat{w} = \arg\min_{g} \max_{d} v(g,d)$$

- Nice differentiable choice for v (zero sum game):

$$v(g,d) = \mathbf{E}\left[\log d(x)\right] + \mathbf{E}\left[\log(1 - d \circ g(z))\right]$$

only portion dependent on g generator minimizes

everything dependent on d discriminator maximizes

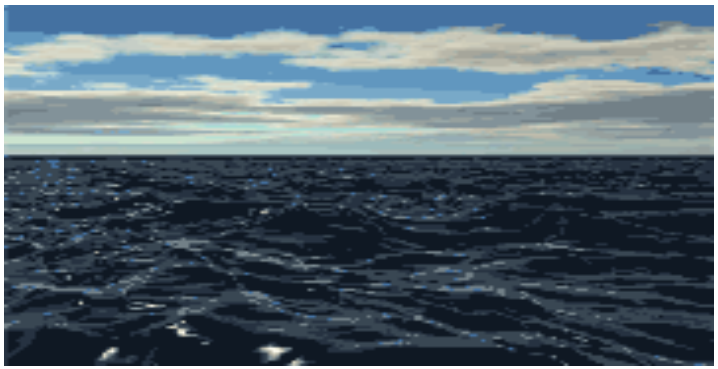
Nash Equilibrium:
Each player wins and loses repeatedly, thus its a zero sum game when they tradeoff performances

Taking turns: Nash Equilibrium

$$v(g,d) = \mathbf{E}\left[\log d(x)\right] + \mathbf{E}\left[\log(1 - d \circ g(z))\right]$$

$$\hat{w} = \arg\min_{g}\max_{d} v(g,d)$$

- If only the problem was convex! This would be easy to solve…
- But $v(g,d)$ is not convex
  - Saddle points everywhere
  - Like optimizing in an Ocean



- Taking turns on the gradient descent may never converge (Nash equilibrium is not convergence)
  - **So we have no convergence guarantee**
  - But practically we like when discriminator == chance

Practical Objectives

- Discriminator objective, gradient **ascent**:

$$\max_{d} \left(\mathscr{L}^{D}_{disc} = \mathbf{E}\left[\log d(x)\right] + \mathbf{E}\left[\log(1 - d \circ g(z))\right] \right)$$

- Generator objective, gradient **descent**:

$$\min_{g} \left(\mathscr{L}^{G}_{gen} = \mathbf{E}\left[\log(1 - d \circ g(z))\right] \right)$$

- But **practically** generator is really hard to train, amplified by a decreasing gradient
- Ian Goodfellow tried to solve this mathematically through a number of different formulations
  - Nothing seemed to work, and then…

Practical Objectives

- Original Generator objective, gradient descent:

$$\min_g \left(\mathscr{L}^G_{gen} = \mathbf{E}\left[\log(1 - d \circ g(z))\right] \right)$$

- Goodfellow *et al.* gave up on using rigorous mathematics (intractable) and relied on heuristic:
  - Instead of minimizing when discriminator is right
  - Why not maximize when it is wrong?
    - not trying to win, just make the other person lose

$$\max_g \left(\mathscr{L}^G_{gen,new} = \mathbf{E}\left[\log(d \circ g(z))\right] \right)$$

No longer a fair zero sum game?!

Now we are not guaranteed convergence or equilibrium when training the GAN…
But it trains…

Final Loss Functions

- Discriminator:

Do not Update Generator Weights

$$\max_{d} \left(\mathscr{L}^{D}_{disc} = \mathbf{E}\left[\log d(x)\right] + \mathbf{E}\left[\log(1 - d \circ g(z))\right] \right)$$

Same as minimizing the binary cross entropy of the model with: (1) real data=1 and (2) generated data=0!

- Generator:

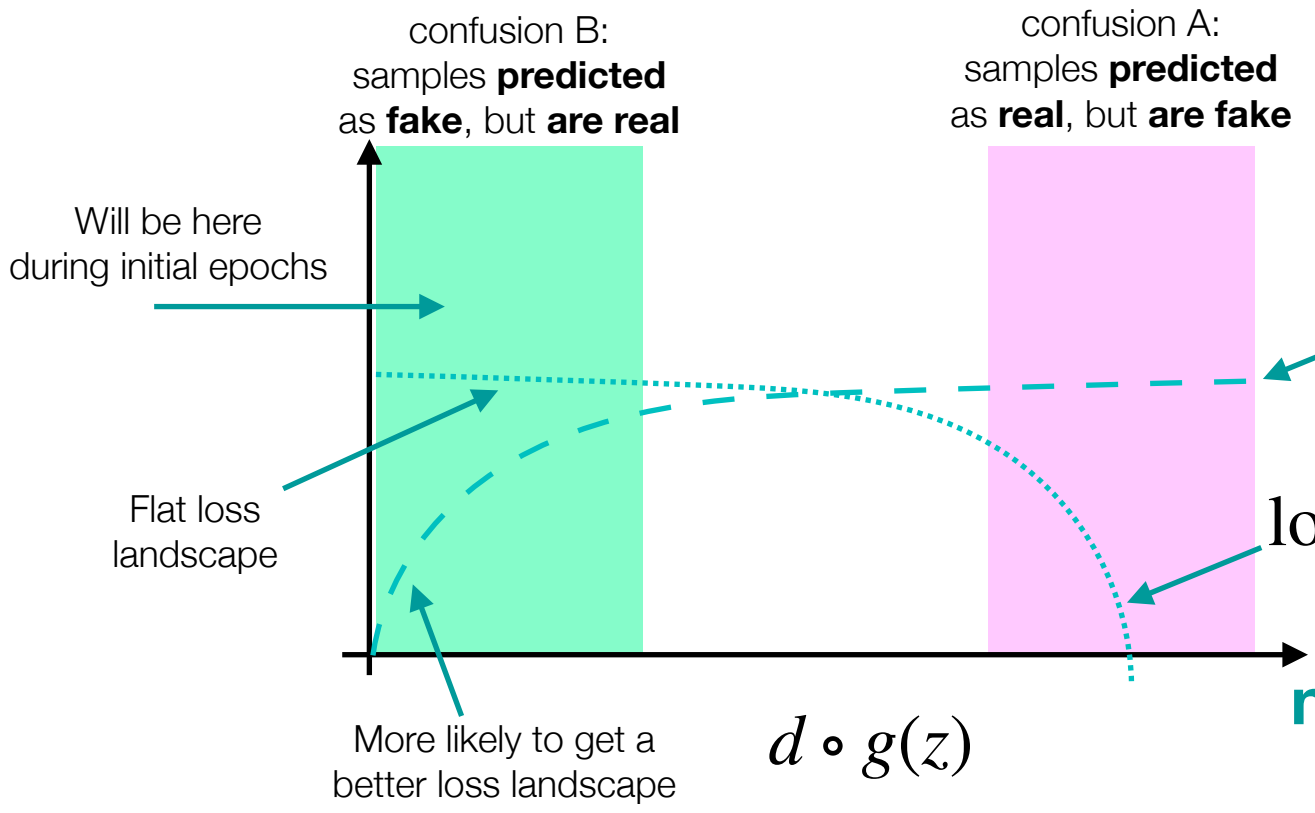
$$\max_{g} \left(\mathscr{L}^{G}_{gen} = \mathbf{E}\left[\log(d \circ g(z))\right] \right)$$

Do Not Update Discriminator Weights

Same as minimizing the binary cross entropy of "mislabeled" generated data=1!

Intuition for why this really helps…

- Training two networks is inherently unstable, need heuristic
- Let's assume generator is not any good for initial epochs!



This is a heuristic not a mathematical proof

DC-GAN

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Alec Radford & Luke Metz
indico Research
Boston, MA
[alec,luke]@indico.io

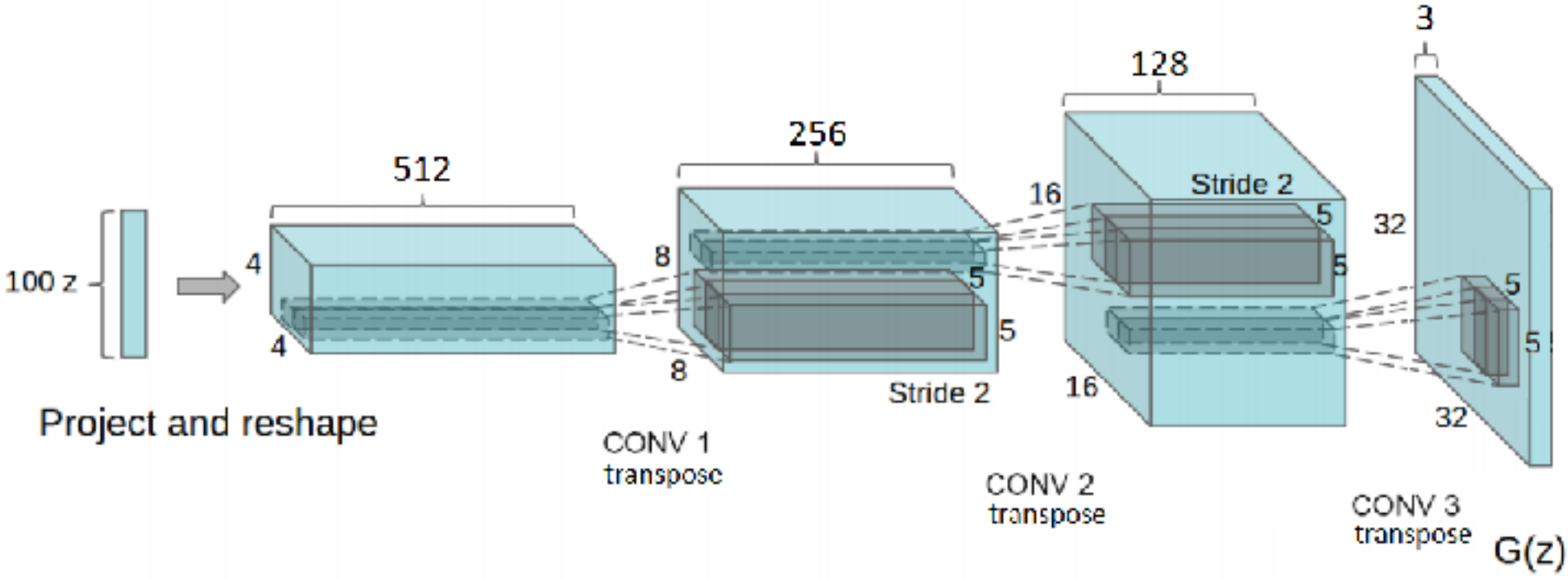
Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

Abstract

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, we use the learned features for novel tasks - demonstrating their applicability as general image representations.

Deep Convolutional GANs

- Just need to reshape and use upsampling



https://sthalles.github.io/intro-to-gans/

Some Results of Generation (not conv)

Goodfellow et al. "Generative Adversarial Networks" (NeurIPS 2014)

Highlight: Nearest Training Sample

So

Radford, Ale ersarial
networks." *a*

A bag of tricks from F. Chollet

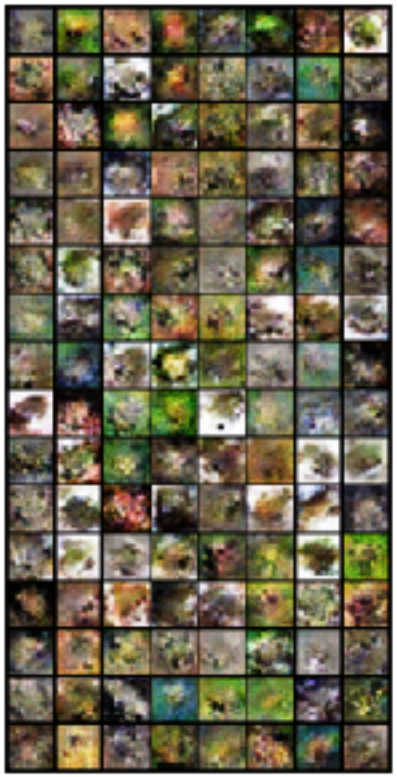
The process of training GANs and tuning GAN implementations is notoriously difficult. There are a number of known tricks you should keep in mind. Like most things in deep learning, it's **more alchemy than science**: **these tricks are heuristics, not theory-backed guidelines**. They're supported by a level of intuitive understanding of the phenomenon at hand, and they're known to work well empirically, although not necessarily in every context.

—Francois Chollet, DL with Python

- Use tanh for generator output, not a sigmoid
  - *We typically normalize inputs to a NN to be from [-1, 1], generator needs to mirror this squashing*
- Sample from Normal Distribution
  - *Everyone else is doing it (practically whatever we do here should help to create a latent space easy to sample from)*
- Random is more robust in optimizer and labels
  - *GANs get stuck a lot, label noise can help move optimization*
- Sparse gradients are **not** your friend here (discriminator should be more transparent, so generator can backprop through it)
  - *No max pooling, no ReLU* 😳
- Make decoder upsampling multiple of stride…
  - *Unequal pixel coverage (checkerboards)*

GANs Demo

Master Repository:
`07c GANsWithKeras.ipynb`



GANs in Keras (optional)

Implementation from Book on
"Frogs from CIFAR"

Demo by Francois Chollet

https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.5-introduction-to-gans.ipynb

GANs Loss

Meta-meta-learning for Neural Architecture Search through arXiv Descent

Antreas Antoniou
MetaMind
aa@mm.ai

Nick Pawlowski
Google x^2
nick@x.x

Jack Turner
slow.ai
jack@slow.ai

James Owers
Facebook AI Research Team
jim@fart.org

Joseph Mellor
Institute of Yellow Jumpers
joe@anditwasall.yellow

Elliot J. Crowley
ClosedAI
elliot@closed.ai

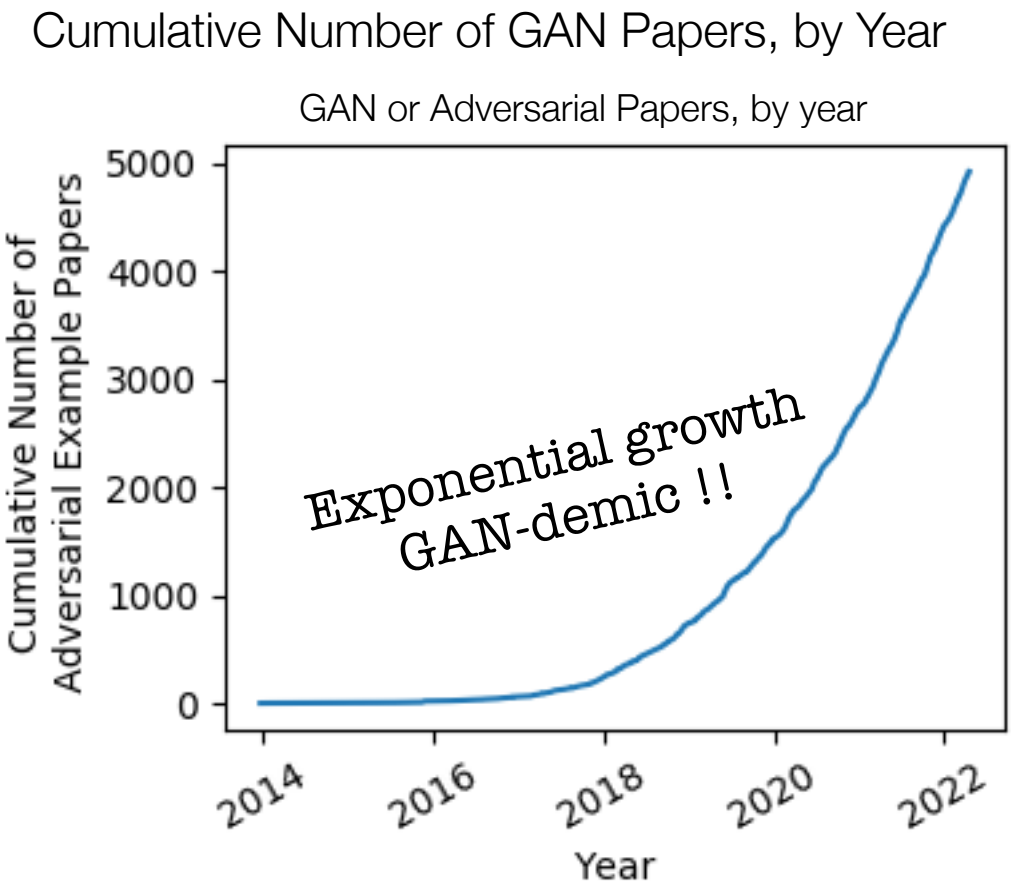
Abstract

Recent work in meta-learning has set the deep learning community alight. From minute gains on few-shot learning tasks, to discovering architectures that are slightly better than chance, to solving intelligence itself[1], meta-learning is proving a popular solution to every conceivable problem ever conceivably conceived ever.

In this paper we venture deeper into the computational insanity that is meta-learning, and potentially risk exiting the simulation of reality itself, by attempting to meta-learn at a third learning level. We showcase the resulting approach—which we call *meta-meta-learning*—for neural architecture search. Crucially, instead of *meta-learning* a neural architecture differentiably as in DARTS (Liu et al., 2018) we *meta-meta-learn* an architecture by searching through arXiv. This *arXiv descent* is GPU-free and only requires a handful of graduate students. Further, we introduce a regulariser, called *college-dropout*, which works by randomly removing a single graduate student from our system. As a consequence, procrastination levels decrease significantly, due to the increased workload and sense of responsibility each student attains.

The code for our experiments is publicly available at [redacted].
Edit: we have decided not to release our code as we are concerned that it may be used for malicious purposes.

There were a bunch go GANs proposed

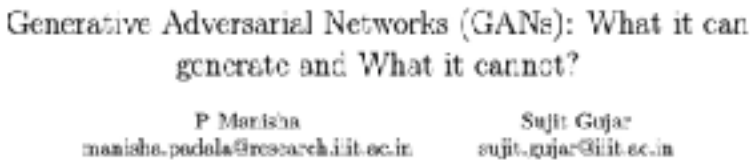


https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

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-Criti
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: A
AffGAN — Amortised MAP Inf
AIM — Generating Informative
AL-CGAN — Learning to Gen
ALI — Adversarially Learned I
AlignGAN — AlignGAN: Learn
AlphaGAN — AlphaGAN: Gen
AM-GAN — Activation Maximi
AmbientGAN — AmbientGAN
AMC-GAN — Video Predictio
AnoGAN — Unsupervised An
APD — Adversarial Distillation
APE-GAN — APE-GAN: Adve
ARAE — Adversarially Regula
ARDA — Adversarial Represe
ARIGAN — ARIGAN: Syntheti
ArtGAN — ArtGAN: Artwork S
ASDL-GAN — Automatic Steg
ATA-GAN — Attention-Aware
Attention-GAN — Attention-G
AttGAN — Arbitrary Facial Att
AttnGAN — AttnGAN: Fine-Gr
AVID — AVID: Adversarial Vis

Why are GANs so difficult to train?

- Why did we need to add noise to labels?
- Why were sparse gradients needed?
- Does using least squares really solve the problem?
- Formalization:
  - GANs will not converge
  - GAN outputs all might be similar (**mode collapse**)
  - Slow training: gradient vanishes, sometimes not recoverable


Generative Adversarial Networks (GANs): What it can generate and What it cannot?

P Manisha
manisha.padala@research.iiit.ac.in

Sujit Gujar
sujit.gujar@iiit.ac.in


TOWARDS PRINCIPLED METHODS FOR TRAINING GENERATIVE ADVERSARIAL NETWORKS

Martin Arjovsky
Courant Institute of Mathematical Sciences
martinarjovsky@gmail.com

Léon Bottou
Facebook AI Research
leonb@fb.com

The Optimal Discriminator

- Discriminator is maximizing:

$$\max_d \mathbf{E}\left[\log d(x)\right] + \mathbf{E}\left[\log(1 - d \circ g(z))\right]$$

… math …

$$d(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

-**Conclusion**: Discriminator is optimal when this condition occurs

In principle, it would seem like **training the discriminator fully** for each update of the generator. Then the generator could simply find a good update to itself based on the optimal discriminator…
…but that is not what we observe. Subsequent updates to the generator (as discriminator is better) do not seem to keep up…

As the discriminator gets more and more confident (i.e., more optimal), it pushes the real distribution of the data away from the generator latent space… gradients vanish

What this means for the Generator

- It can be shown that the generator objective function (according to the Goodfellow loss) is optimal when it maximizes this:

$$C(G) = -\log(4) + KL\left(p_{data} \parallel \frac{p_{data} + p_g}{2}\right) + KL\left(p_g \parallel \frac{p_{data} + p_g}{2}\right)$$

$$C(G) = -\log(4) + 2.JSD(p_{data} \parallel p_g)$$ Jenson-Shannon Divergence

- Which helps explain the **Vanishing gradients**: if $p_{data} \| p_g \approx 0$ then JSD saturates and the gradient is basically zero. Optimization has no idea what direction to move in…
  - **Indirect Solution:** Perhaps we can just use tips/tricks to get around the vanishing gradient problem?
    - Not covering these methods this semester …
  - **Direct Solution**: get rid of the loss function from Goodfellow
    - **Use a better objective: Wasserstein Distance**

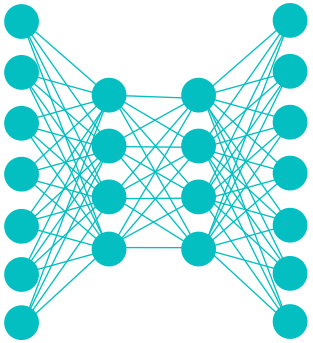
Lecture Notes for

Neural Networks and Machine Learning

Simple GANs

Next Time:
Wasserstein GANs

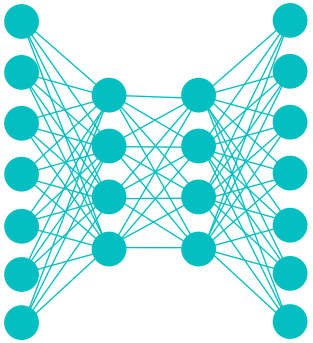
Reading: WGAN Paper



Lecture Notes for

Neural Networks and Machine Learning

Wasserstein GANs



Logistics and Agenda

- Logistics
  - **Student Presentation**: None
- Agenda
  - Quick revisiting: GAN Optimality
  - Wasserstein GAN
  - WGAN-GP
  - Demo
  - BigGAN

Last Time:

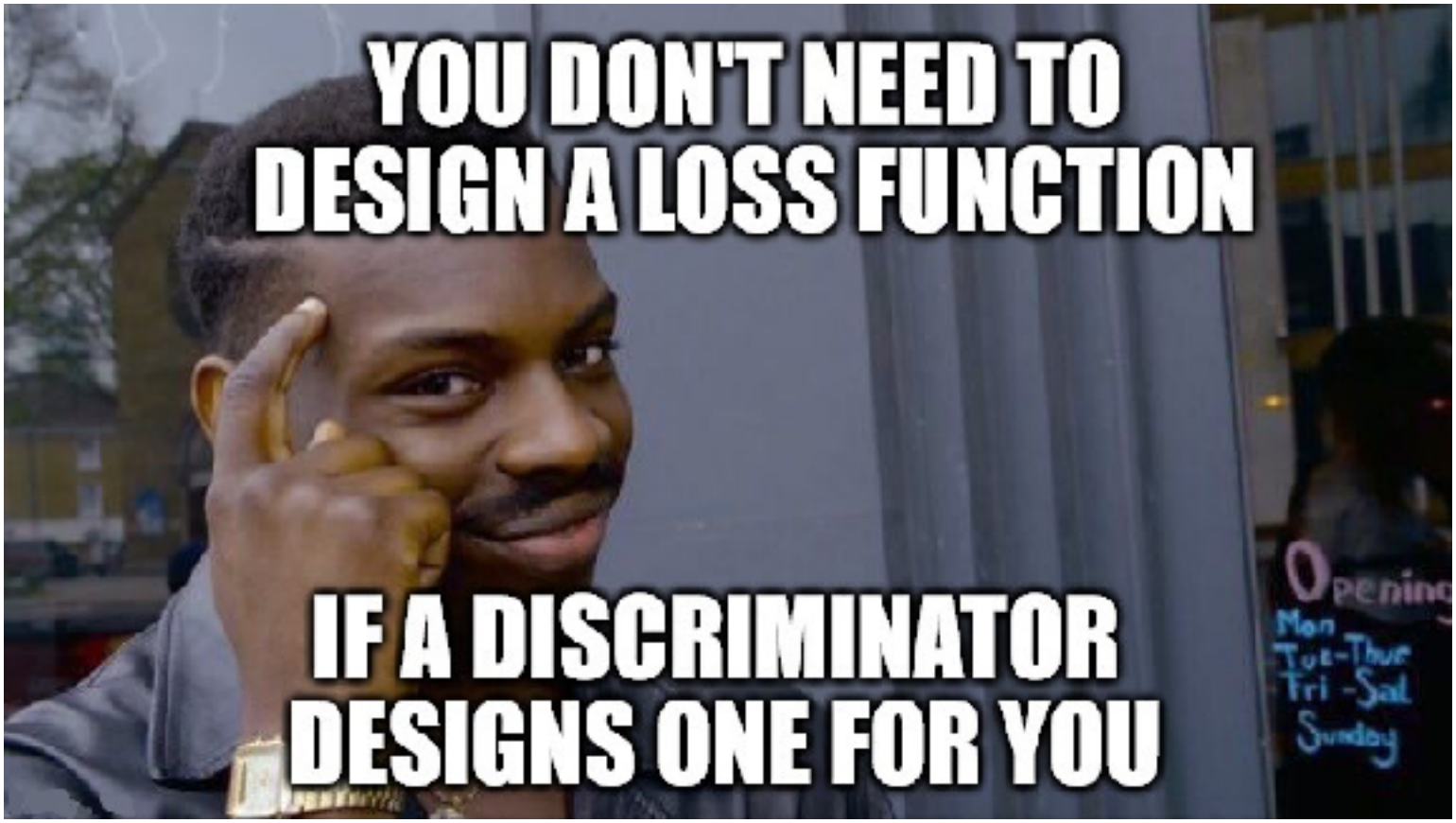
- It can be shown that the generator objective function (according to the Goodfellow loss) is optimal when it maximizes this:

$$C(G) = -\log(4) + KL\left(p_{data} \parallel \frac{p_{data} + p_g}{2}\right) + KL\left(p_g \parallel \frac{p_{data} + p_g}{2}\right)$$

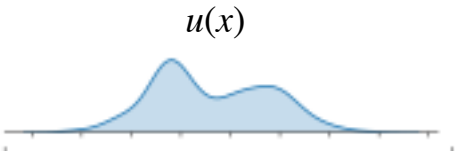
$$C(G) = -\log(4) + 2.JSD(p_{data} \parallel p_g)$$ Jenson-Shannon Divergence

- Which helps explain the **Vanishing gradients**: if $p_{data} \| p_g \approx 0$ then JSD saturates and the gradient is basically zero. Optimization has no idea what direction to move in…
  - **Indirect Solution:** Perhaps we can just use tips/tricks to get around the vanishing gradient problem?
    - Not covering these methods this semester …
  - **Direct Solution**: get rid of the loss function from Goodfellow
    - **Use a better objective: Wasserstein Distance**

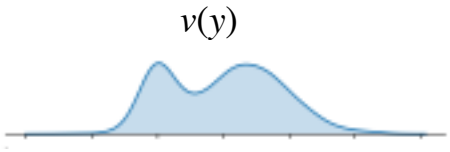
Wasserstein GANs

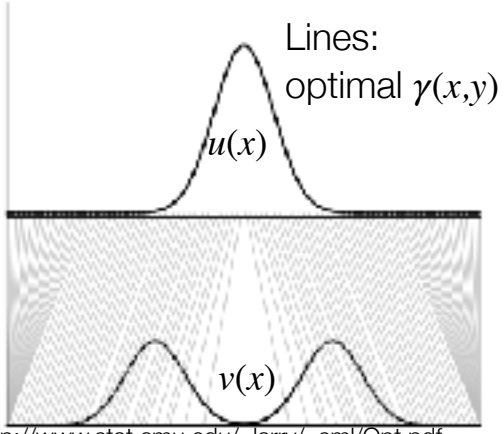


Wasserstein Distance (Earth Mover)



how to move dirt from u to v ?

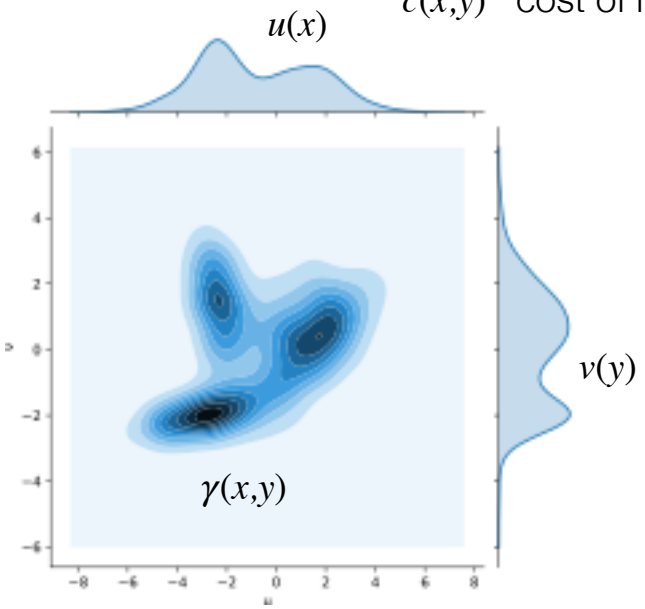




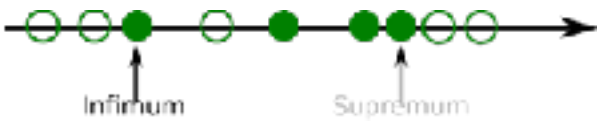
http://www.stat.cmu.edu/~larry/=sml/Opt.pdf

$\gamma(x,y)$ one plan to move u to v

$c(x,y)$ cost of moving u to v



$$\int\int c(x,y)\cdot\gamma(x,y)\,\partial x\partial y$$ Total cost of plan γ

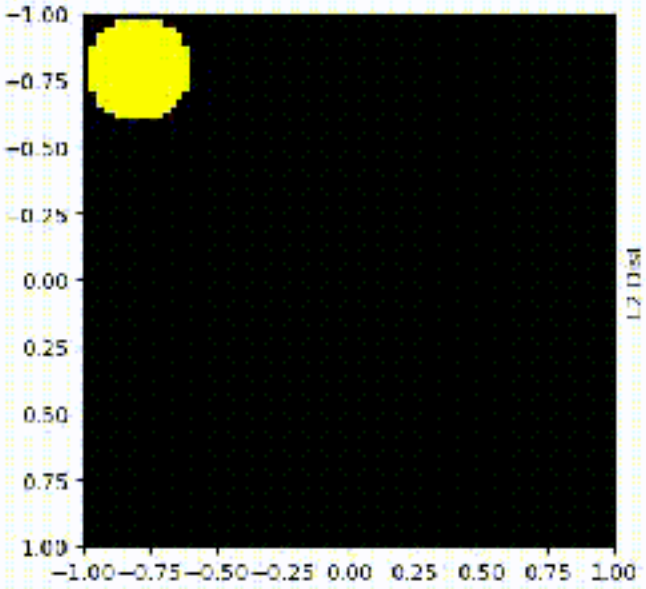


$$\inf_{\gamma\in\Pi}\int\int c(x,y)\cdot\gamma(x,y)\,\partial x\partial y$$ Optimal plan has greatest lower bound (minimum cost in set Π)

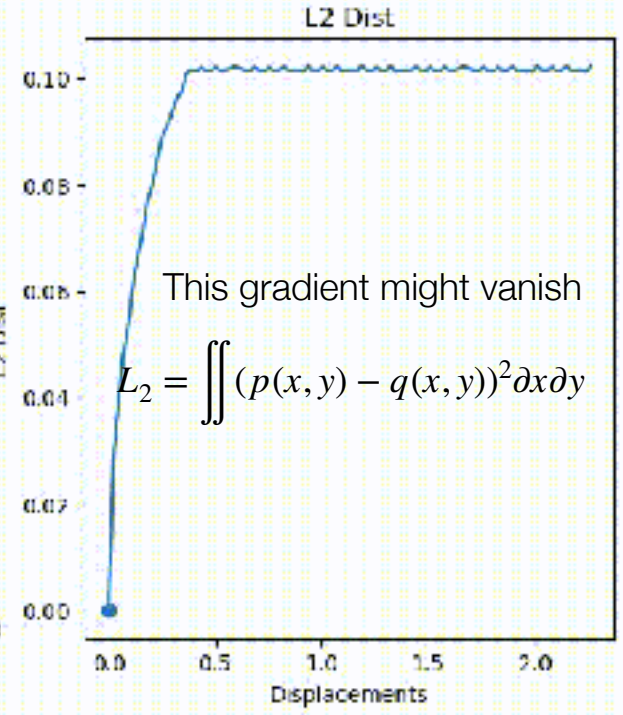
$$\inf \mathbf{E}_{x,y\leftarrow\gamma}\left[|x-y|\right] = \inf_{\gamma\in\Pi}\int\int |x-y|\cdot\gamma(x,y)\,\partial x\partial y$$ If cost is difference to move from x to y

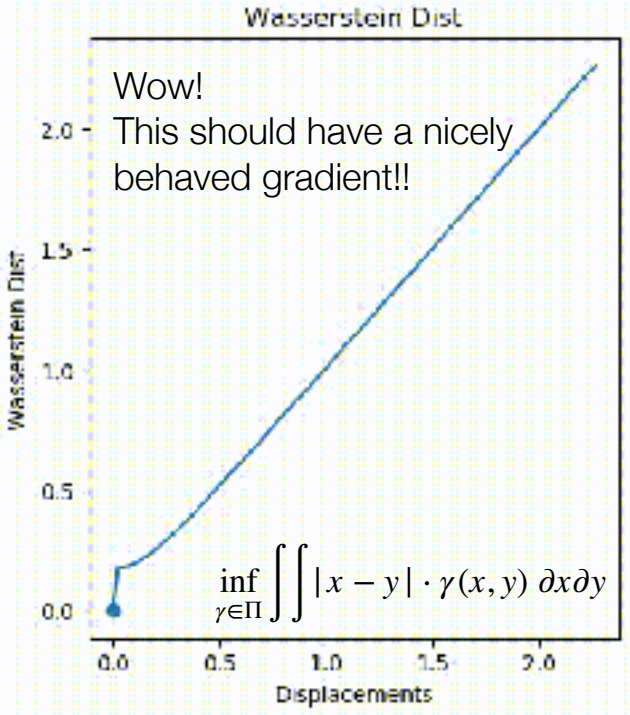
image: wikipedia

Wasserstein Distance



Two 2D Normal Distributions





- *Wasserstein GAN adds few tricks to allow the Discriminator to approximate Wasserstein (aka Earth Mover's) distance between real and model distributions. Wasserstein distance roughly tells "how much work is needed to be done for one distribution to be adjusted to match another" and is remarkable in a way that it is defined even for non-overlapping distributions.*

https://gist.github.com/ctralie/66352ae6ab06c009f02c705385a446f3

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490