Deep Learning

Chapter 10: Generative Adversarial Networks (GANs)

Mina Rezaei

Department of Statistics – LMU Munich

Winter Semester 2020

LECTURE OUTLINE

Generative Adversarial Networks (GANs)

GAN Training

Challenges for GAN Optimization

Loss-variant GANs

Architecture-variant GANs

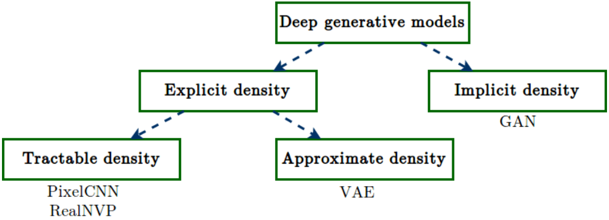
GAN Evaluation

Generative Adversarial Networks (GANs)

GENERATIVE ADVERSARIAL NETWORKS (GANS)

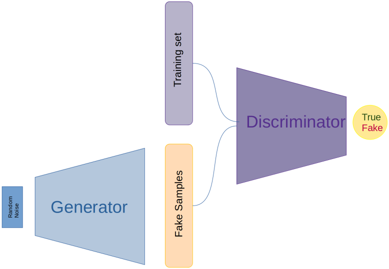
Generative adversarial networks

- define the generative model as in VAEs,
- but approach problem of learning a directed generative model $p(\mathbf{x}|\mathbf{z})$ from a totally different perspective.



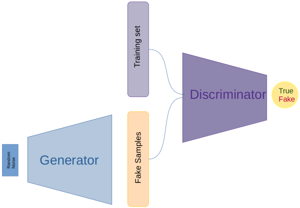
Taxonomy of Generative Model

WHAT IS A GAN?



- Vanilla GAN consists of two deep neural networks:
  - the generator
  - the discriminator
- The generator is fed a random noise vector which it transforms into a fake sample in a given domain.
- The discriminator is fed both real and fake samples and outputs a number between 0 and 1 indicating the probability of the input being real.

WHAT IS A GAN?



- The goal of the generator is to fool the discriminator into thinking that the synthesized samples are real.
- The goal of the discriminator is to accurately recognize real samples and not be fooled by the generator.
- This sets off an arms race. As the generator gets better at producing realistic samples, the discriminator is forced to get better at detecting the fake samples which in turn forces the generator to get even better at producing realistic samples and so on.

FAKE CURRENCY ILLUSTRATION

The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles.

-Ian Goodfellow

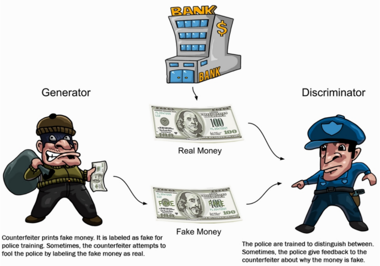


Image created by Mayank Vadsola

GAN Training

MINIMAX LOSS FOR GANS

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

- $p_{\text{data}(\mathbf{x})}$ is our target, the data distribution.
- Recall, that we defined the generator to be neural network mapping a latend random vector $\mathbf{z}$ to generated samples $G(\mathbf{z})$. Thus even if the generator is a determinisic function, we have random outputs, i.e. variability.
- $p(\mathbf{z})$ is usually a uniform distribution or an isotropic Gaussian. It is typically fixed and not adapted during training.

MINIMAX LOSS FOR GANS

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

- $G(\mathbf{z})$ is the output of the generator for a given state $\mathbf{z}$ of the latent variables.
- $D(\mathbf{x})$ is the output of the discriminator for a real sample $\mathbf{x}$.
- $D(G(\mathbf{z}))$ is the output of the discriminator for a fake sample $G(\mathbf{z})$ synthesized by the generator.

MINIMAX LOSS FOR GANS

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

- Roughly speaking, $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}(\mathbf{x})}}[\log D(\mathbf{x})]$ is the log-probability of correctly classifying real data points as real.
- $\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$ is the log-probability of correctly classifying fake samples as fake.
- Therefore, with each gradient update, the discriminator tries to push $D(\mathbf{x})$ toward 1 and $D(G(\mathbf{z})))$ toward 0. This is the same as maximizing V(D,G).
- The generator, on the other hand, only has control over $D(G(\mathbf{z}))$ and tries to push that toward 1 with each gradient update. This is the same as minimizing V(D,G).

GAN TRAINING : PSEUDOCODE

Algorithm Minibatch stochastic gradient descent training of GANs. The number of steps ,k to apply to the discriminatoris a hyperparameter

for number of training iterations **do**
    for k steps **do**
        Sample minibatch of m noise samples $\{\mathbf{z}^{(1)} \dots \mathbf{z}^{(m)}\}$ from the noise prior $p_g(\mathbf{z})$
        Sample minibatch of m examples $\{\mathbf{x}^{(1)} \dots \mathbf{x}^{(m)}\}$ from the data generating distribution $p_{\text{data}}(\mathbf{x})$.
        Update the discriminator by ascending its stochastic gradient:

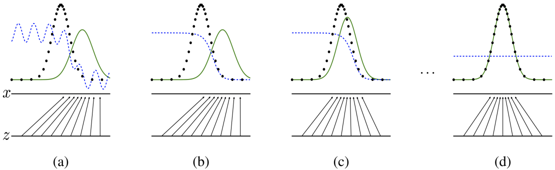
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)}))) \right]$$

    end for
    Sample minibatch of m noise samples $\{\mathbf{z}^{(1)} \dots \mathbf{z}^{(m)}\}$ from the noise prior $p_g(\mathbf{z})$
    Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)})))$$

end for

GAN TRAINING: ILLUSTRATION



GANs are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black,dotted line) p_x from those of the generative distribution $p_g(G)$ (green, solid line).Source: Goodfellow et al (2017),

- For k steps, G's parameters are frozen and one performs **gradient ascent** on D to increase its accuracy.
- Finally, D's parameters are frozen and one performs **gradient descent** on G to increase its generation performance.
- Note, that G gets to peek at D's internals (from the back-propagated errors) but D does not get to peek at G.

Challenges for GAN Optimization

ADVERSARIAL TRAINING

Deep Learning models (in general) involve a single player!

- The player tries to maximize its reward (minimize its loss),
- Use SGD (with backprob) to find the optimal parameters,
- SGD has convergence guarantees (under certain conditions).
- However, with non-convexity, we might convert to local minima!

GAN instead involve two players

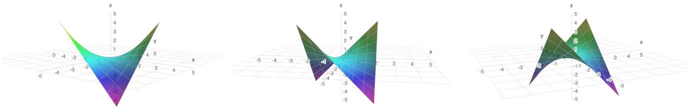
- Discriminator is trying to maximize its reward,
- Generator is trying to minimize Discriminator's reward

(*) SGD was not designed to find the Nash equilibrium of a game!

(**) Therefore, we might not converge to the Nash equilibrium at all!

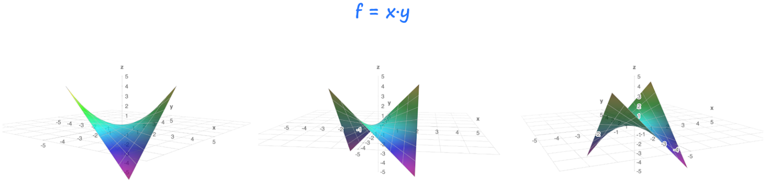
ADVERSARIAL TRAINING -EXAMPLE

$f = x \cdot y$



- Consider the function $f(x, y) = xy$, where x and y are both scalars.
- Player A can control x and Player B can control y.
- The loss:
  - Player A: $L_A(x, y) = xy$
  - Player B: $L_B(x, y) = -xy$
- This can be rewritten as $L(x, y) = \min_x \max_y xy$
- What we have here is a simple zero-sum game with its characteristic minimax loss.

POSSIBLE BEHAVIOUR #1: CONVERGENCE

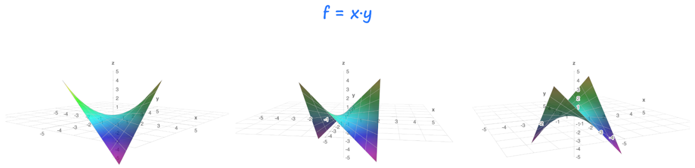


- The partial derivatives of the losses are:

$$\frac{\partial L_A}{\partial x} = y \ , \ \frac{\partial L_B}{\partial y} = -x$$

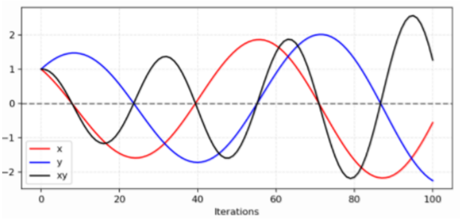
- In adversarial training, both players perform gradient descent on their respective losses.
- To perform simultaneous gradient descent, we update x with $x - \alpha \cdot y$ and y with $y + \alpha \cdot x$ simultaneously in one iteration, where α is the learning rate.

POSSIBLE BEHAVIOUR #1: CONVERGENCE



- In order for simultaneous gradient descent to converge to a fixed point, both gradients have to be simultaneously 0.
- They are both (simultaneously) zero only for the point (0,0).
- This is a saddle point of the function $f(x, y) = xy$.
  - The fixed point for a minimax game is typically a saddle point.
  - Such a fixed point is an example of a Nash equilibrium.
- In adversarial training, convergence to a fixed point is **not** guaranteed.

POSSIBLE BEHAVIOUR #2: CHAOTIC BEHAVIOUR



Credit: Lilian Weng

Figure: A simulation of our example for updating x to minimize xy and updating y to minimize -xy. The learning rate α = 0.1. With more iterations, the oscillation grows more and more unstable.

- Once *x* and *y* have different signs, every following gradient update causes huge oscillation and the instability gets worse in time, as shown in the figure.

POSSIBLE BEHAVIOUR #3: CYCLES

Credit: Goodfellow

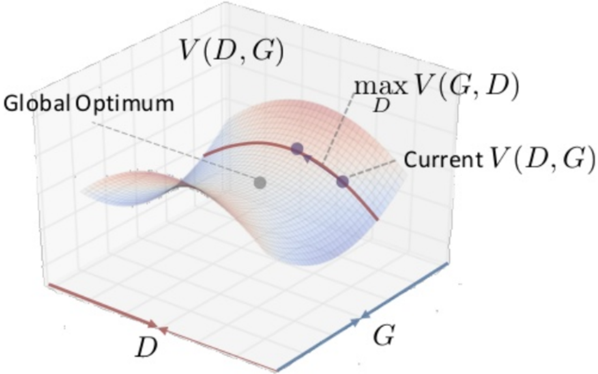
Figure: Simultaneous gradient descent with an infinitesimal step size can result in a circular orbit in the parameter space.

- A Discrete Example: A never-ending game of Rock-Paper-Scissors where player A chooses 'Rock' $\rightarrow$ player B chooses 'Paper' $\rightarrow$ A chooses 'Scissors' $\rightarrow$ B chooses 'Rock' $\rightarrow$ ...
- **Takeaway:** Adversarial training is highly unpredictable. It can get stuck in cycles or become chaotic.

NON-STATIONARY LOSS SURFACE

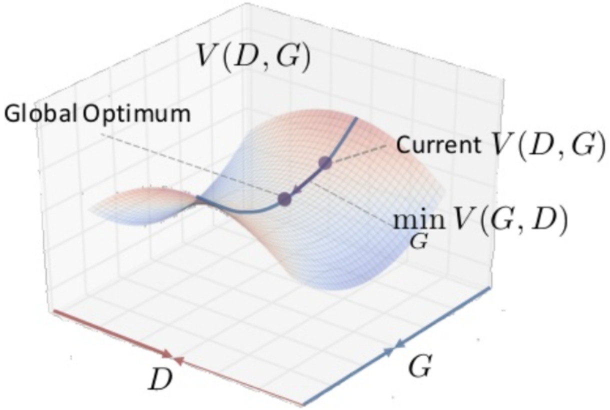
- Once again, it is extremely important to note that from the perspective of one of the players, the loss surface changes every time the other player makes a move.
- This is in stark contrast to the (full batch) gradient descent case where the loss surface is stationary no matter how many iterations of gradient descent are performed.

ILLUSTRATION OF CONVERGENCE



Credit: Mark Chang

ILLUSTRATION OF CONVERGENCE: FINAL STEP



Credit: Mark Chang

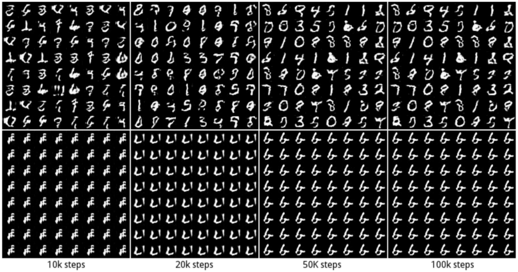
Such convergence is not guaranteed, however.

CHALLENGES FOR GAN TRAINING

- Non-convergence: the model parameters oscillate, destabilize and never converge,
- Mode collapse: the generator collapses which produces limited varieties of samples,
- Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing,
- Unbalance between the generator and discriminator causing overfitting,
- Highly sensitive to the hyperparameter selections.

MODE COLLAPSE

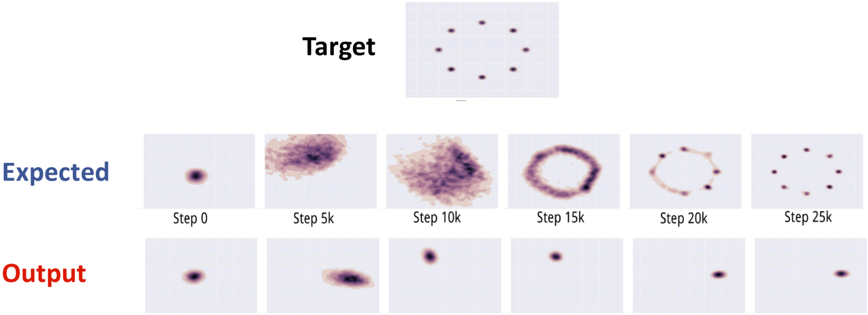
Real-life data distributions are multimodal. For example, in MNIST, there are 10 major modes from digit ‘0’ to digit ‘9’. The samples below are generated by two different GANs. The top row produces all 10 modes while the second row creates a single mode only (the digit “6”). This problem is called mode collapse when only a few modes of data are generated.



Credit: Luke Metz et al.(2017)

MODE COLLAPSE- EXAMPLE

Generator fails to output diverse samples on Toy dataset



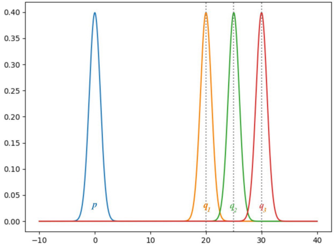
Metz, Luke, et al. **"Unrolled Generative Adversarial Networks."** arXiv preprint arXiv:1611.02163 (2016).

SOLUTIONS TO TACKLE MODE COLLAPSE

- Mini-batch GAN:
  - Problem: Generator produces good samples but a very few of them, therefore discriminator can not tag them as fake!
  - Solution: Let the discriminator look at the entire batch instead of single example, If there is lack of diversity mark them fake!
  - Result: Generator will be force to produce diverse samples!
- Train with labels
  - Label information of real data might help
  - Here, we train D with all categories instead of binary label!

VANISHING GRADIENTS IN JS-DIVERGENCE

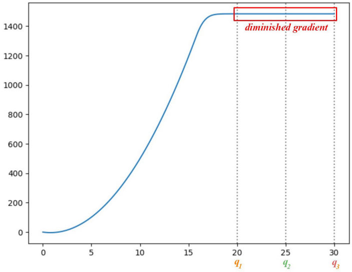
- Recall that when the discriminator is optimal, the objective function for the generator is:
  $\min_{G} V(D^*, G) = 2D_{JS}(p_r||p_g) - 2\log 2$
- The problem happens when the data distribution q of the generator's images does not match with the ground truth p for the real images.
- Let's consider q with different means to study the gradient of $JS(p, q)$



Credit: Jonathan Hui.(2017)

VANISHING GRADIENTS IN JS-DIVERGENCE

- Here, we plot the JS-divergence $JS(p, q)$ between p and q with means of q ranging from 0 to 30.
- As shown below, the gradient for the JS-divergence vanishes from q_1 to q_3.
- Results: The GAN generator will learn extremely slow to nothing when the cost is saturated in those regions. In early training, p and q are very different and the generator learns very slow.



Credit: Jonathan Hui.(2017)

DIVERGENCE MEASURES

- Recall that the goal of generative modeling is to learn $p_{\text{data}}(\mathbf{x})$.
- In order to understand the differences between different generative models, it is sometimes helpful to consider them from the angle of **divergence measures**.
- A divergence measure quantifies the distance between two distributions. It is a measure of how different one distribution is from another.
- There are many different divergence measures that one can use here (such as the Kullback-Leibler divergence).
- One thing that all such measures have in common is that they are 0 if and only if the two distributions are equal to each other (otherwise, they are all positive).

DIVERGENCE MEASURES

- One approach to training generative models, then, is to explicitly minimize the distance between $p_{\text{data}}(\mathbf{x})$ and the model distribution $p_\theta(\mathbf{x})$ according to some divergence measure.
- If our generator has the capacity to model $p_{\text{data}}(\mathbf{x})$ perfectly, the choice of divergence does not matter much because they all achieve their minimum (that is 0) when $p_g(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$.
- However, it is not likely that that the generator, which is parametrized by the weights of a neural network, is capable of perfectly modelling an arbitrary $p_{\text{data}}(\mathbf{x})$.
- In such a scenario, the choice of divergence measure matters, because the parameters that miniminize the various divergence measures differ.

IMPLICIT DIVERGENCE MEASURE OF GANS

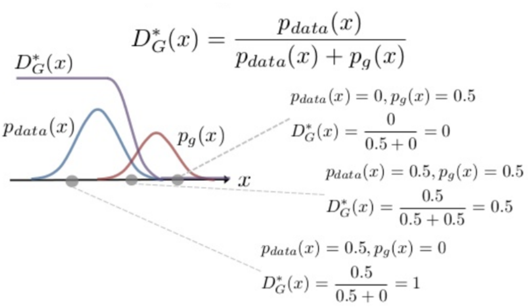
- GANs do not explicitly minimize any divergence measure.
- However, (under some assumptions!) optimizing the minimax loss is equivalent to implicitly minimizing a divergence measure.
- That is, if the optimal discriminator is found in every iteration, the generator minimizes the **Jensen-Shannon divergence (JSD)** (theorem and proof are given by the original GAN paper (Goodfellow et al, 2014)):

$$JS(p_{\text{data}}||p_g) = \frac{1}{2}KL(p_{\text{data}}||\frac{p_{\text{data}} + p_g}{2}) + \frac{1}{2}KL(p_g||\frac{p_{\text{data}} + p_g}{2})$$

$$KL(p_{\text{data}}||p_g) = E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[log\frac{p_{\text{data}}(\mathbf{x})}{p_g(\mathbf{x})}]$$

OPTIMAL DISCRIMINATOR

For G fixed, the optimal discriminator D_G^* is:

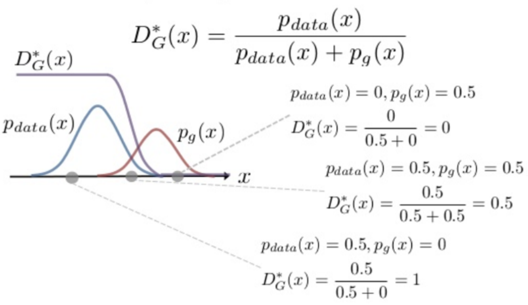


Credit: Mark Chang

- The optimal discriminator returns a value greater than 0.5 if the probability to come from the data ($p_{data}(x)$) is larger than the probability to come from the generator ($p_g(x)$).

OPTIMAL DISCRIMINATOR

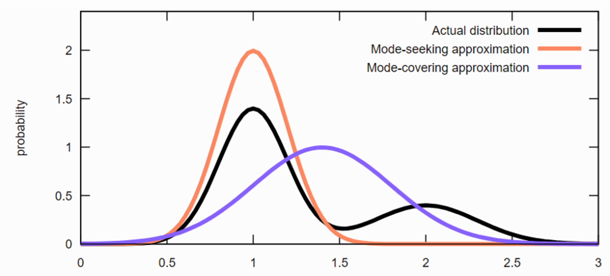
For G fixed, the optimal discriminator D_G^* is:



Credit: Mark Chang

- Note: The optimal solution is almost never found in practice, since the discriminator has a finite capacity and is trained on a finite amount of data.
- Therefore, the assumption needed to guarantee that the generator minimizes the JSD does usually not hold in practice.

TRADE-OFFS MADE BY SOME COMMON DIVERGENCES



Credit: Aiden NIbali

- In this simplified 1-dimensional example, $p_{\text{data}}(x)$ is a bimodal distribution, but $p_\theta(x)$ only has the modelling capacity of a single Gaussian.
- Therefore, based on the divergence measure, $p_\theta(x)$ can either fit a single mode really well, i.e. be 'mode-seeking' (e.g. JSD), or attempt to cover both modes, i.e. be 'mode-covering' (e.g. KLD).

ALTERNATIVE DIVERGENCES

Many common divergences, such as KL-divergence, Hellinger distance, and total variation distance, are special cases of f-divergence, coinciding with a particular choice of f. The following table lists many of the common divergences between probability distributions and the f function.

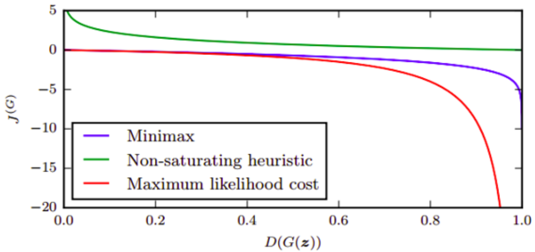
Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$	$1+\log\frac{p(x)}{q(x)}$
Reverse KL	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int\frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)}-1)$
Squared Hellinger	$\int\left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}}-1)\cdot\sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$	$\log\frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$	$\log\frac{p(x)}{p(x)+q(x)}$

Credit: Aiden NIbali

- We train a distribution Q and an approximate the divergence with $T^*(x)$

Loss-variant GANs

NON-SATURATING LOSS

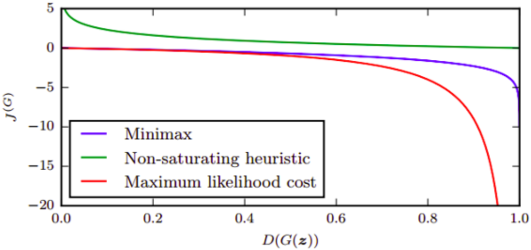


Credit: Daniel Seita

Figure: Various generator loss functions ($J^{(G)}$).

- It was discovered that a relatively strong discriminator could completely dominate the generator.
- When optimizing the minimax loss, as the discriminator gets good at identifying fake images, i.e. as $D(G(\mathbf{z}))$ approaches 0, the gradient with respect to the generator parameters vanishes.

NON-SATURATING LOSS



Credit: Daniel Seita

Figure: Various generator loss functions ($J^{(G)}$).

- Solution: Use a non-saturating generator loss instead: $J^{(G)} = -\frac{1}{2}\mathbb{E}_{\vec{z}\sim p(\vec{z})}[\log D(G(\mathbf{x}))]$
- In contrast to the minimax loss, when the discriminator gets good at identifying fake images, the magnitude of the gradient of $J^{(G)}$ increases and the generator is able to learn to produce better images in successive iterations.

OTHER LOSS FUNCTIONS

Various losses for GAN training with different properties have been proposed:

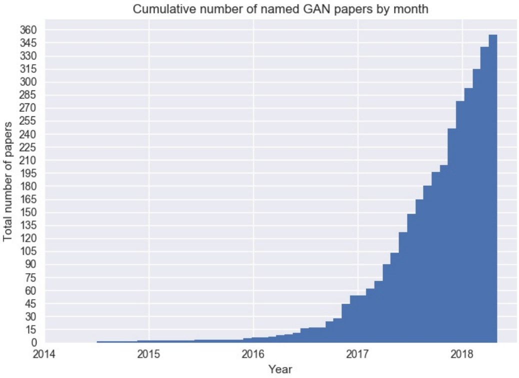
GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_{\text{D}}^{\text{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\text{D}}^{\text{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\text{D}}^{\text{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\text{G}}^{\text{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\text{D}}^{\text{WGANGP}} = \mathcal{L}_{\text{D}}^{\text{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\|\nabla D(\alpha x + (1-\alpha\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\text{D}}^{\text{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x)-1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\text{G}}^{\text{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x}-1)^2]$
DRAGAN	$\mathcal{L}_{\text{D}}^{\text{DRAGAN}} = \mathcal{L}_{\text{D}}^{\text{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\text{D}}^{\text{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \text{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\text{G}}^{\text{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$

Source: Lucic et al. 2016

Architecture-variant GANs

ARCHITECTURE-VARIANT GANS

Motivated by different challenges in GAN training procedure described, there have been several types of architecture variants proposed. Understanding and improving GAN training is a very active area of research.



Credit: hindupuravinash

GAN APPLICATION

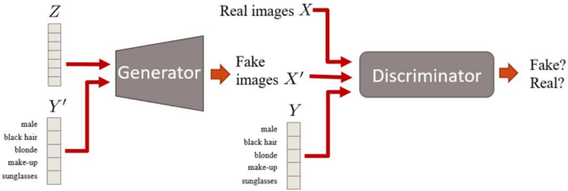
What kinds of problems can GANs address?

- Generation
- Conditional Generation
- Clustering
- Semi-supervised Learning
- Representation Learning
- Translation
- Any traditional discriminative task can be approached with generative models

CONDITIONAL GANS: MOTIVATION

- In an ordinary GAN, the only thing that is fed to the generator are the latent variables $\mathbf{z}$.
- A conditional GAN allows you to condition the generative model on additional variables.
- E.g. a generator conditioned on text input (in addition to $\mathbf{z}$) can be trained to generate the image described by the text.

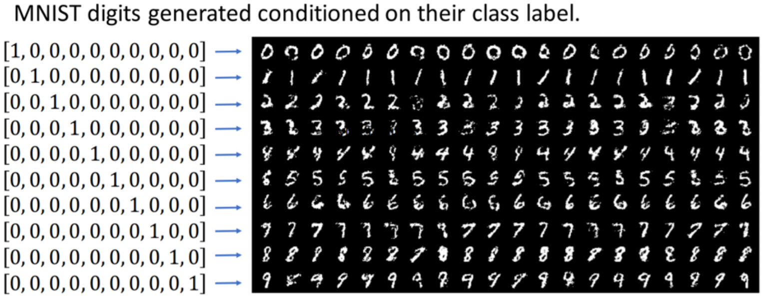
CONDITIONAL GANS: ARCHITECTURE



Credit: Guim Perarnau

- In a conditional GAN, additional information in the form of vector $\vec{y}$ is fed to both the generator and the discriminator.
- $\vec{z}$ can then encode all variations in $\vec{z}$ that are not encoded by $\vec{y}$.
- E.g. $\vec{y}$ could encode the class of a hand-written number (from 0 to 9). Then, $\vec{z}$ could encode the style of the number (size, weight, rotation, etc).

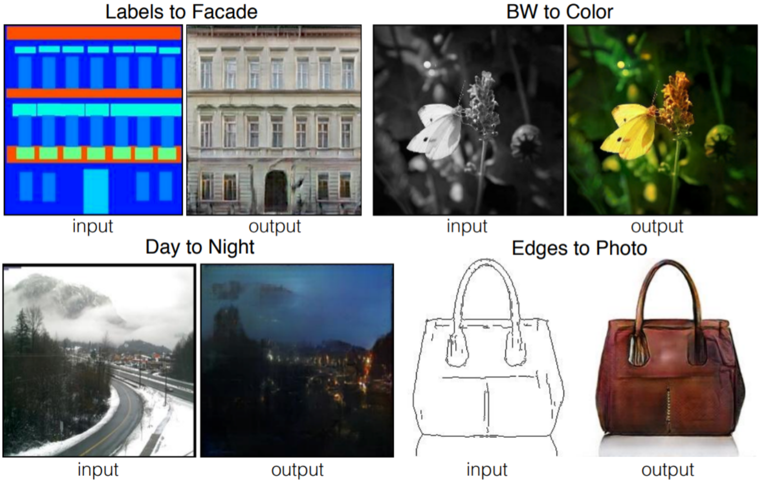
CONDITIONAL GANS: EXAMPLE



Source: Mirza et al. 2014

Figure: When the model is conditioned on a one-hot coded class label, it generates random images that belong (mostly) to that particular class. The randomness here comes from the randomly sampled **z**. (Note : **z** is implicit. It is not shown above.)

CONDITIONAL GANS: MORE EXAMPLES



Source: Isola et al. 2016

Figure: Conditional GANs can translate images of one type to another. In each of the 4 examples above, the image on the left is fed to the network and the image on the right is generated by the network.

MORE GENERATIVE MODELS

- Today, we learned about two kinds of (directed) generative models:
  - Variational Autoencoders (VAEs)
  - Generative Adversarial Networks (GANs).
- There are other interesting generative models, e.g.:
  - autoregressive models
  - restricted Boltzmann machines.
- Note:
  - It is important to bear in mind that generative models are not a solved problem.
  - There are many interesting hybrid models that combine two or more of these approaches.

GAN Evaluation

GAN EVALUATION

What makes a good generative model?

- Each generated sample is indistinguishable from a real sample
- Generated samples should have variety

Fake faces generated by ThisPersonDoesNotExist.com. | Image: The Verge

GAN EVALUATION

How to evaluate the generated samples?

- Cannot rely on the models' loss
- Human evaluation
- Use a pre-trained model

	Measure	Objective
Quantitative	Inception score (Salimans et al., 2016)	$IS(G) = exp\left(E_{x\ P_{data}(x)} D_{KL}(p(y\|x)\|p(y))\right)$
	Fréchet inception distance (Heusel et al., 2017)	$\text{FID} = \|\mu_x - \mu_g\| + (\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g))$
	The wasserstein critic (Gulrajani et al., 2017)	$W(P_r, P_g) = \inf_{\eta \in \Pi(P_r, P_g)} E_{(x,y)\eta} \|[x - y]\|$
	Reconstruction error (Xiang & Li, 2017)	
	Geometry score (Lucic et al., 2018)	Geometrical comparisons between real data and generated data
	Precision-recall (Khrulkov & Oseledets, 2018)	Quantify the degree of overfitting in GANs, often over dataset.
Qualitative	Preference judgment (Chen et al., 2016)	Ask from participant to rank models in terms of the fidelity of their generated images.
	Mode drop and collapse (Srivastava et al., 2017)	Over datasets with known modes (e.g., a GMM or a labeled dataset), modes are computed as by measuring the distances of generated data to mode centers.
	Network internal (Zhang et al., 2017)	Illustrating the internal representation and dynamics of models (e.g., space continuity) as well as visualizing learned features.

source: M. Rezaei (2020)

GAN EVALUATION- INCEPTION SCORE

The inception score measures the quality of generated samples.

- They used the Inception model (Szegedy et al., 2015) trained on ImageNet
- Given generated image x, assigned the label y by model p:
  $P(y|x) \rightarrow$ low entropy (one class)
- The distribution over all generated images should be spread (evaluating mode collapse)
  $\int P(y|x = G(z))dz \rightarrow$ high entropy (many classes)
- The distribution over all generated images should be spread: (evaluating mode collapse)
  $exp(E_x KL(p(y|x)||p(y)))$

REFERENCES

- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014)
  Generative Adversarial Networks
  https://arxiv.org/abs/1406.2661
- Santiago Pascual, Antonio Bonafonte, Joan Serra (2017)
  SEGAN: Speech Enhancement Generative Adversarial Network
  https://arxiv.org/abs/1703.09452
- Ian Goodfellow (2016)
  NIPS 2016 Tutorial: Generative Adversarial Networks
  https://arxiv.org/abs/1701.00160
- Lilian Weng (2017)
  From GAN to WGAN
  https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

REFERENCES

Mark Chang (2016)
Generative Adversarial Networks
https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks

Lucas Theis, Aaron van den Oord, Matthias Bethge (2016)
A note on the evaluation of generative models
https://arxiv.org/abs/1511.01844

Aiden Nibali (2016)
The GAN objective, from practice to theory and back again
https://aiden.nibali.org/blog/2016-12-21-gan-objective/

Mehdi Mirza, Simon Osindero (2014)
Conditional Generative Adversarial Nets
https://arxiv.org/abs/1411.1784

REFERENCES

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros (2016)
Image-to-Image Translation with Conditional Adversarial Networks
https://arxiv.org/abs/1611.07004

Guim Perarnau (2017)
Fantastic GANs and where to find them
https://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them