Neural Networks: Generative Adversarial Networks

Mark Craven and David Page
Computer Sciences 760
Spring 2019

Goals for the lecture

you should understand the following concepts

- generative models
- Nash equilibrium
- generative adversarial networks
- the vanishing gradient problem

An application of a generative adversarial network (GAN)

- a *generative model* takes a training set and learns to represent an estimate of the distribution the training instances were drawn from
- in some applications, we may use the generative model to generate new instances
- e.g.

https://thispersondoesnotexist.com/
[Karras et al. 2019]

The Prisoner's Dilemma

- To understand the underlying theory of GANs, let's first consider the Prisoner's Dilemma
  - Scenario: the police have arrested two suspects for a crime.
    - They tell each prisoner they'll reduce his/her prison sentence if he/she betrays the other prisoner.
    - Each prisoner must choose between two actions:
    - cooperate with the other prisoner, i.e., don't betray him/her
    - defect (betray the other prisoner).
  - Payoff = – (years in prison):
  - Each player has only two strategies, each of which is a single action
  - Non-zero-sum
  - Imperfect information: neither player knows the other's move until after *both* players have moved

Prisoner's Dilemma

Agent 1 \ Agent 2	C	D
C	–2, –2	–5, 0
D	0, –5	–4, –4

Nash Equilibrium

- A strategy profile $s = (s_1, \ldots, s_n)$ is a **Nash equilibrium** if for every i,
  - s_i is a best response to S_{-i}, i.e., no agent can do better by unilaterally changing his/her strategy

S_{-i} represents all possible strategies opponents could take

- **Theorem (Nash, 1951)**: Every game with a finite number of agents and action profiles has at least one Nash equilibrium
- In the Prisoner's Dilemma, (D,D) is a Nash equilibrium
  - If either agent unilaterally switches to a different strategy, his/her expected utility goes below 1
- A dominant strategy equilibrium is always a Nash equilibrium

Prisoner's Dilemma

Agent 1 \ Agent 2	C	D
C	3, 3	0, 5
D	5, 0	1, 1

Minimax with simultaneous moves

- *minimax* is a decision rule that minimizes the possible loss for a worst case scenario

$$\overline{v_i} = \min_{s_{-i}} \max_{s_i} v_i(s_i, s_{-i})$$

Where:

- i is the index of the player of interest.
- $-i$ denotes all other players except player i

- a *zero-sum* game is one in which each player's gain or loss is exactly balanced by the others'
- in a zero-sum game, the minimax solution is the same as Nash Equilibrium

Generative Adversarial Networks (GANs)

Key idea: set up zero-sum game between two deep nets

Generator network: generate data that looks like training data

$$\boldsymbol{x} = G\left(\boldsymbol{z}; \theta^{(G)}\right)$$

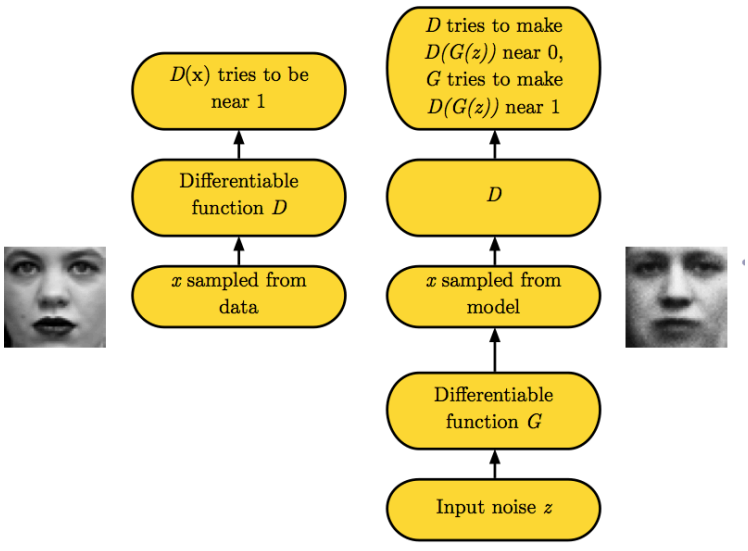
generated instance — random vector — trainable parameters of generator

Discriminator network: distinguish between real and generated data

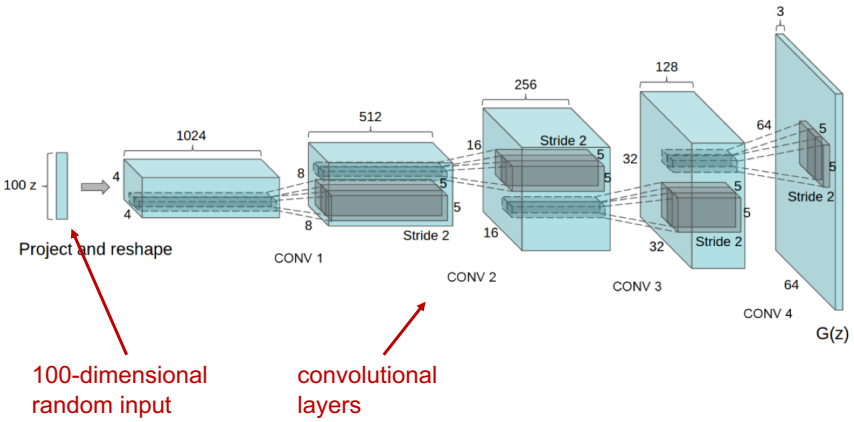
$$\boldsymbol{y} = D\left(\boldsymbol{x}; \theta^{(D)}\right)$$

real or generated? — instance — trainable parameters of discriminator

Overview of GAN training



A typical generator network [Radford et al. *ICLR* 2016]



GAN training

Use stochastic gradient descent on two minibatches simultaneously

- a minibatch of training instances
- a minibatch of generated instances

Set up a zero-sum game by using the following loss functions for the discriminator and the generator

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

$$E\left(\theta^{(G)}\right) = -E\left(\theta^{(D)}\right)$$

GAN training

Equivalently, the training objective function is:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

The Nash equilibrium corresponds to $G(\boldsymbol{z})$ recovering p_{data} exactly

GAN application

Generating images from text descriptions [Reed et al. 2016]

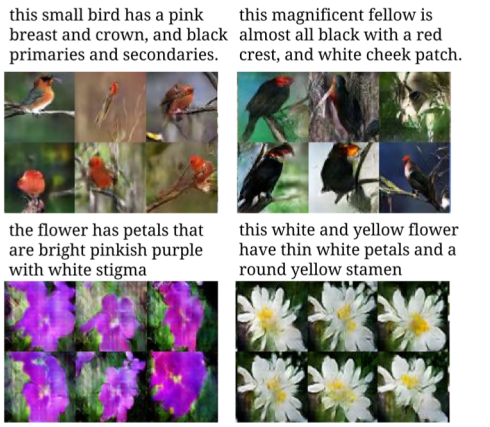
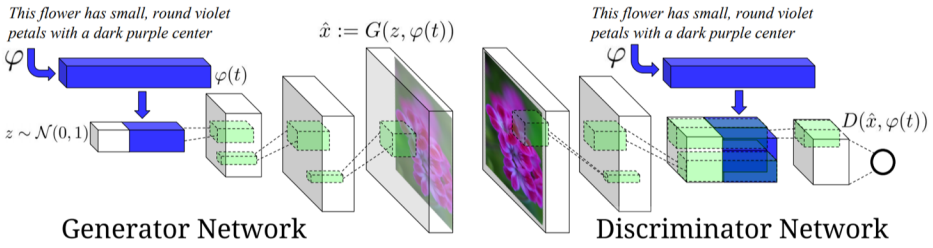


Figure 1. Examples of generated images from text descriptions. Left: captions are from zero-shot (held out) categories, unseen text. Right: captions are from the training set.

GAN application

Generating images from text descriptions [Reed et al. 2016]



GAN application

Increasing image resolution [Ledig et al. 2016]

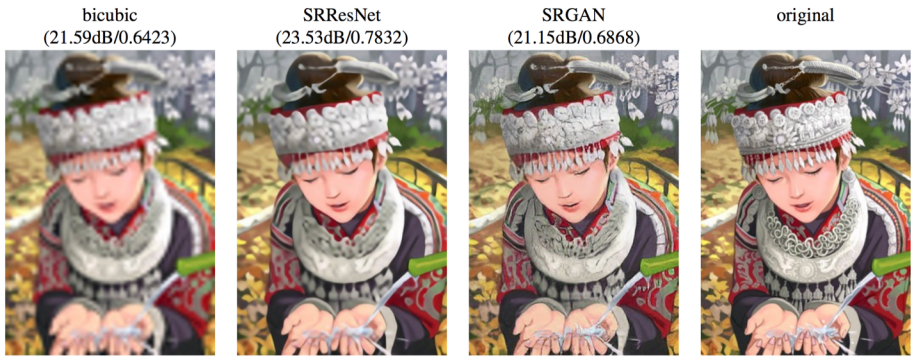


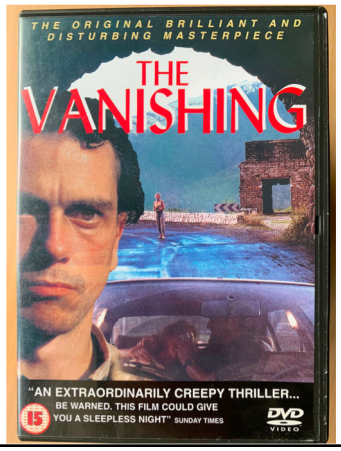
Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

GAN application

Image to image translation [Isola et al. 2016]

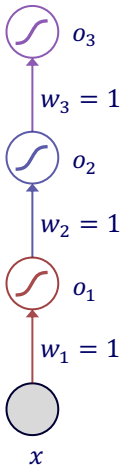
Vanishing gradients

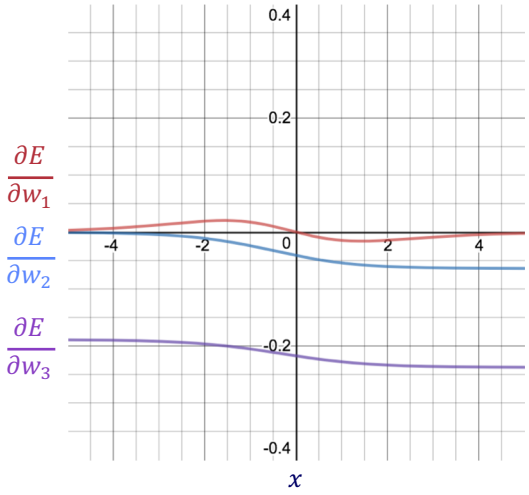
- the *vanishing gradient* problem refers to the fact that the gradients for weights in lower layers in a deep network can be vanishingly small
- especially a problem with traditional activation functions like sigmoid, tanh
- the implication is that it is hard to train these layers since weights are adjusted very slowly



Vanishing gradients

- consider the gradients for the simple network below when the target output is $y = 1$
- magnitude of gradient for w_1 is much smaller than w_3

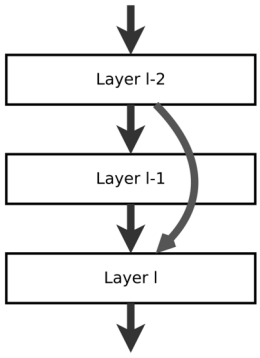




Vanishing gradients

Ways to alleviate the vanishing gradient problem

- activations like ReLU that have only saturate in one direction; have large derivatives otherwise
- training networks layer by layer (stacked autoencoders)
- sparsely connected networks like convolutional nets
- *residual* connections that skip layers



Comments on neural networks

- Many other architecture and training variants we haven't covered
- Given their flexibility, some have referred to these new approaches as "differential programming"

- deep networks have had much recent success due to a combination of tricks and factors
  - rectified linear units to handle the vanishing gradients problem
  - dropout and other regularization methods to avoid overfitting
  - sparsely connected architectures (e.g. convolutional networks) to incorporate task-specific bias
  - very large data sets and hardware to enable training with them

Comments on neural networks

- another style of deep networks, Restricted Boltzmann machines (RBMs), are based on undirected graphical models

- stochastic gradient descent often works well for very large data sets even with simple models (i.e. no hidden units)
  - one pass (or a few passes) through the data set may be sufficient to learn a good model