Introduction to

Generative Adversarial Networks

Summer School on Generative Models

Morten Hannemose, mohan@dtu.dk

August 13th, 2019

Program for the day

- Now: Lecture
- Lunch
- 13:00 PyTorch Concepts (10 minutes)
- Exercise
- 15:15 CycleGAN walkthrough
- Resume exercise

- Dinner

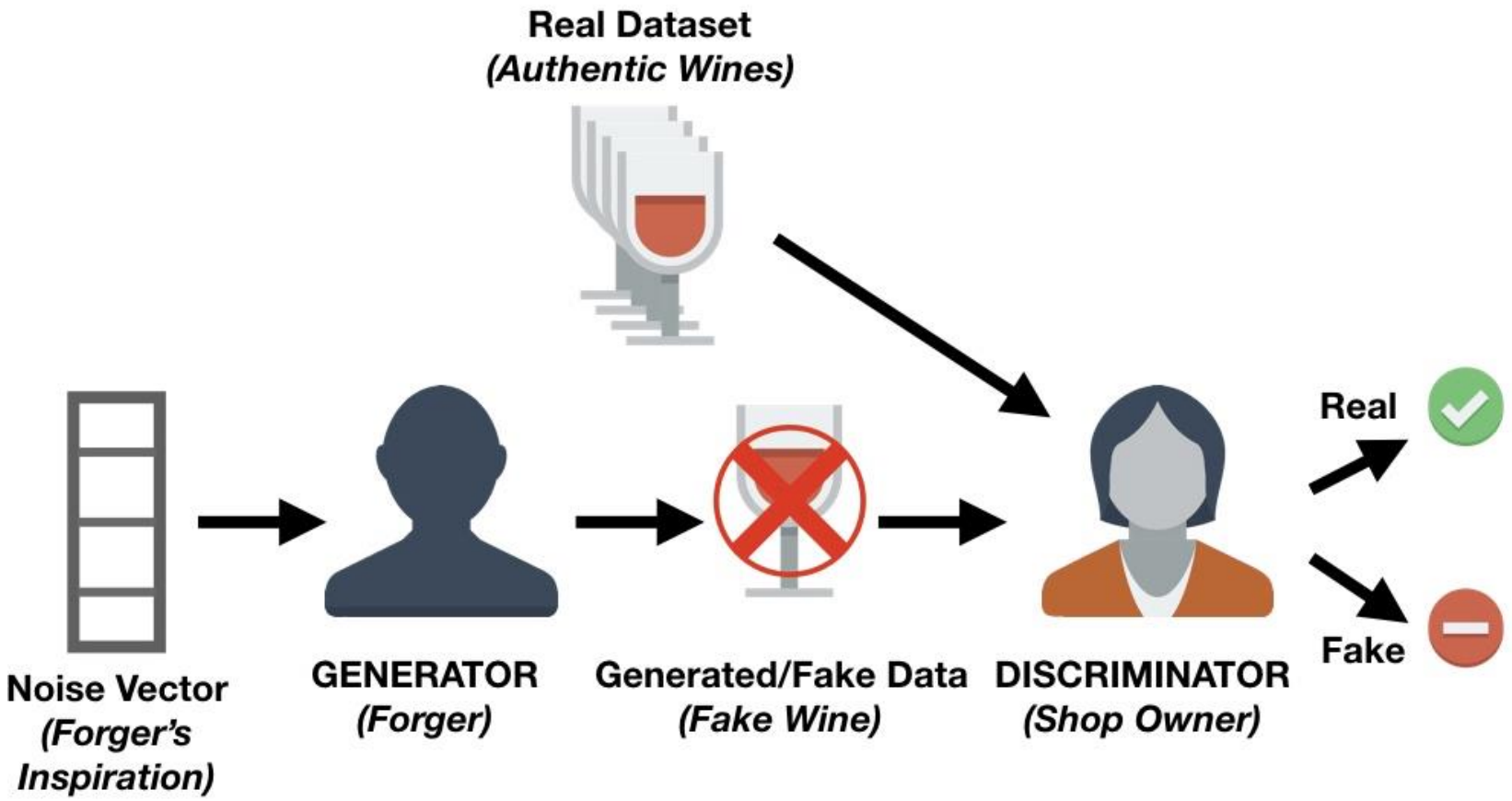
Outline – What you're going to see

- Introduction
- Applications
- How to train a GAN?
- Variations of GANs
- Exercise time!

Who am I

- Morten Hannemose
- PhD student
  - Image Analysis and Computer Graphics Section
  - Technical University of Denmark
- Work mostly with:
  - Image-based optimization
  - 3D reconstruction
  - Deep learning

Conceptual example



What?

- Introduced in 2014 by Ian Goodfellow
- Generator learns a mapping from one probability distribution to another
  - Commonly from a low dimensional Gaussian distribution to the distribution of images you train it on

Examples

- These images are generated from random noise (and conditioned to be a specific class)
  - BigGAN [2018]

Examples

- 4.5 years of GAN progress on face generation

Source: https://twitter.com/goodfellow_ian/status/1084973596236144640/

Which face is real?

- Cool website
  - http://www.whichfaceisreal.com

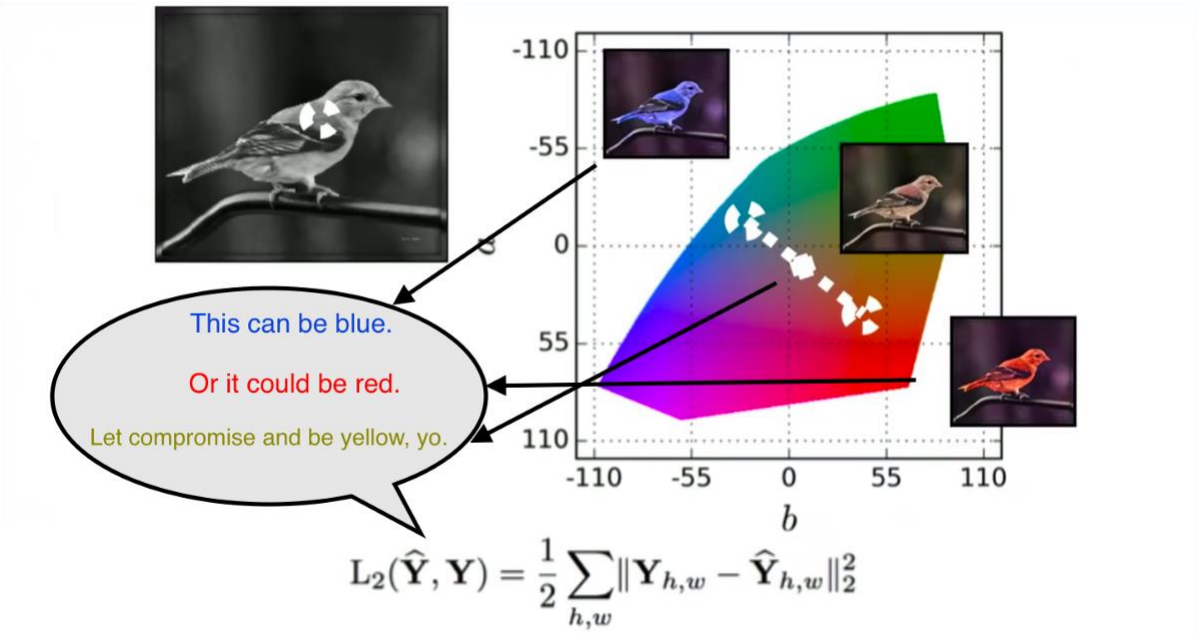
Why?

- Data without labels is abundant – we want to use it
- Being able to learn the distribution of your data is useful

- Many applications

Outputting images

- You want the network to output an image
  - L2 loss (mean squared error) gives blurry images
  - L1 loss (mean absolute error) gives sharper images
  - Both are very sensitive to pixel changes that don't mean anything perceptually



Applications

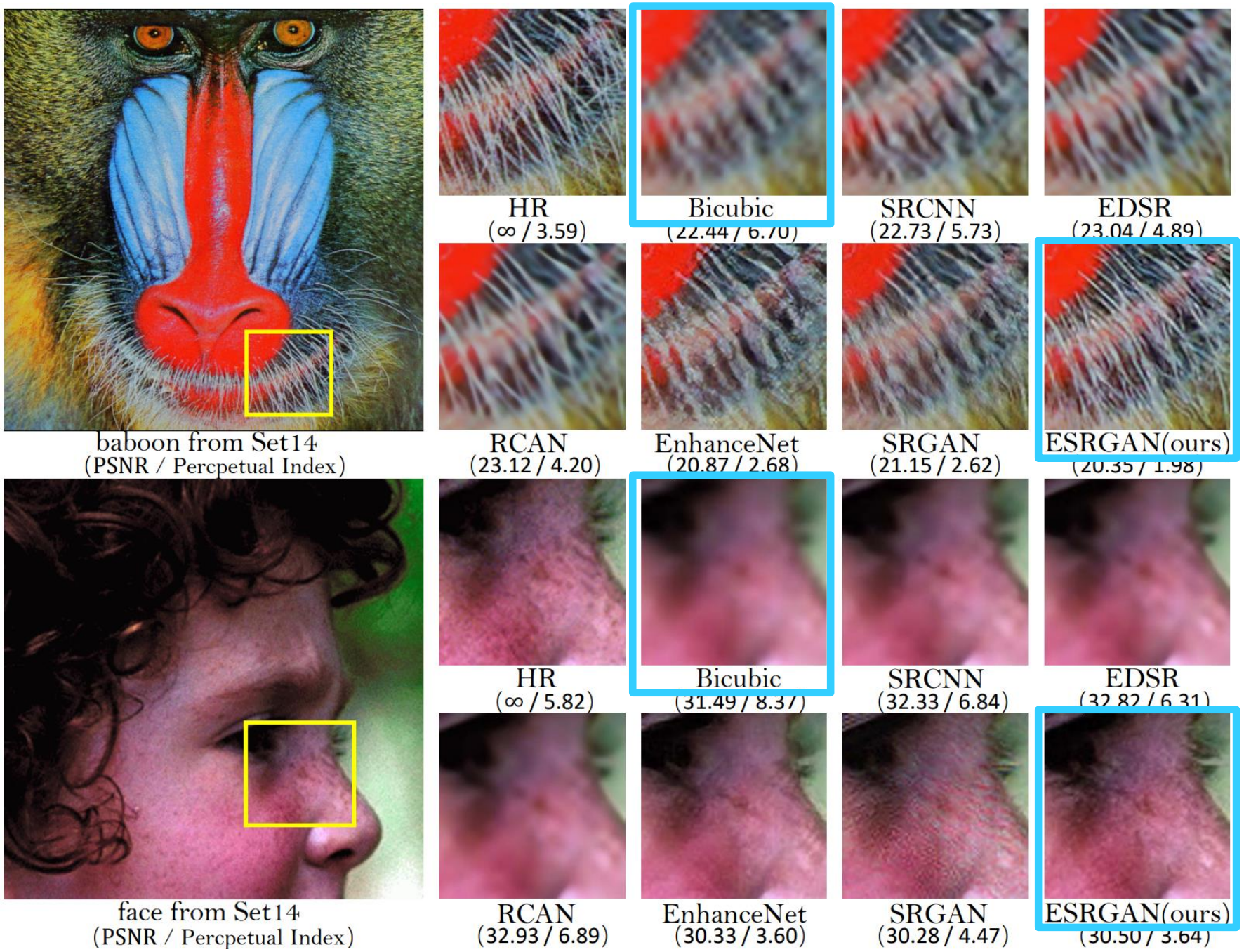
- Super resolution
- Colorization
- Inpainting
- Domain-transfer
- Generating additional training data
- And more (on Thursday)

Generating additional training data

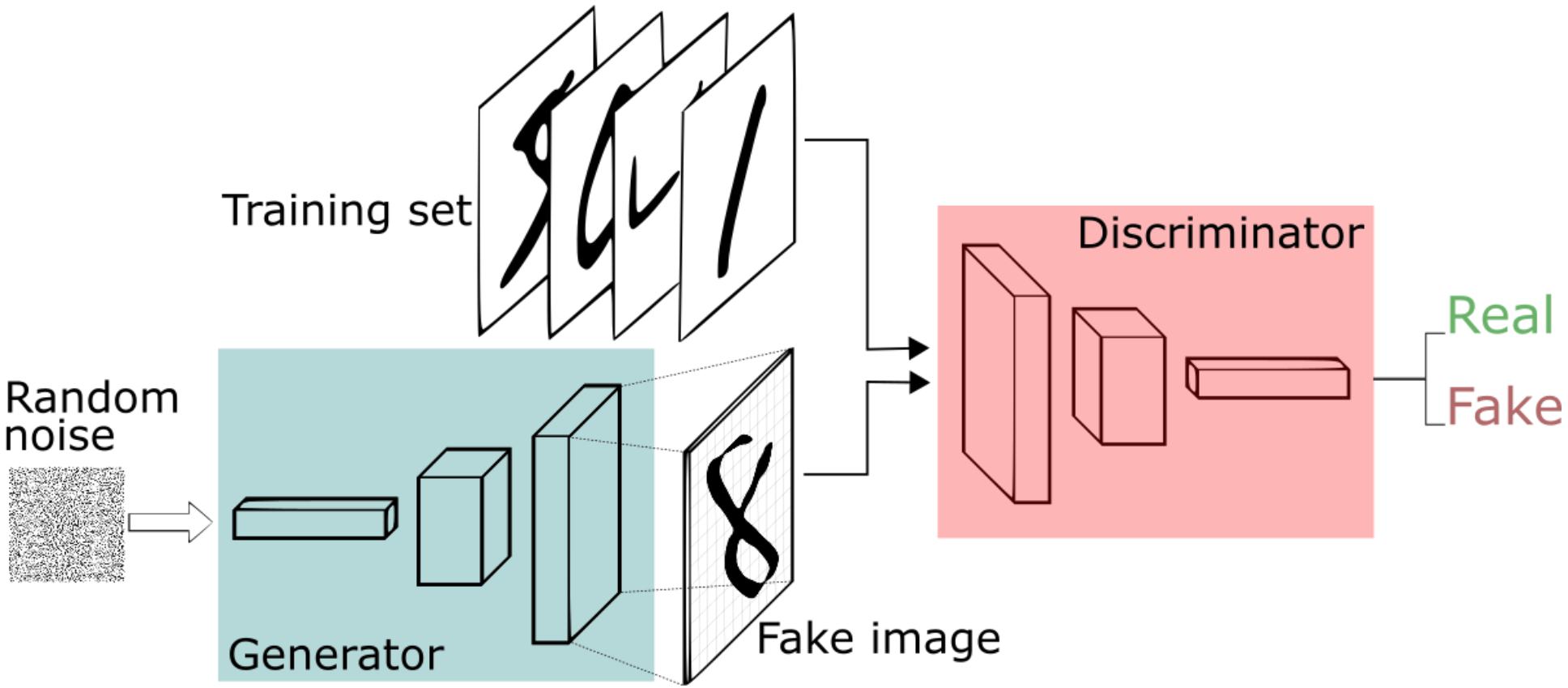
- Making rendered images look like real images
  - But because they are rendered, we have ground truth labels

[2016] Ashish Shrivastava et al. Learning from Simulated and Unsupervised Images through Adversarial Training
https://arxiv.org/abs/1612.07828

Super resolution example (ESRGAN)



Conceptual recap



How?

- Fully connected
- Many different losses possible
- Train generator and discriminator in an alternating fashion
  - Train discriminator for k iterations (can be k=2) (or k=1 and higher LR for D)
  - Then train generator once
  - Repeat
- Adam $\beta_1 = 0.5$, learning rate = 0.0002
  - Default parameter of $\beta_1 = 0.9$ doesn't work well (sometimes)
- Shorthand:
  - G: Generator
  - D: Discriminator

How to do GANs?

- Training GANs is still very hard
  - Many problems exist
  - Non-convergence
    - The models never converge and worse they become unstable.
  - Mode collapse
    - The generator produces a single or limited modes.
      - i.e. the images are not as diverse as the true data.

- Many tricks exist

2014 - GAN

Goodfellow et al. *Generative Adversarial Nets*

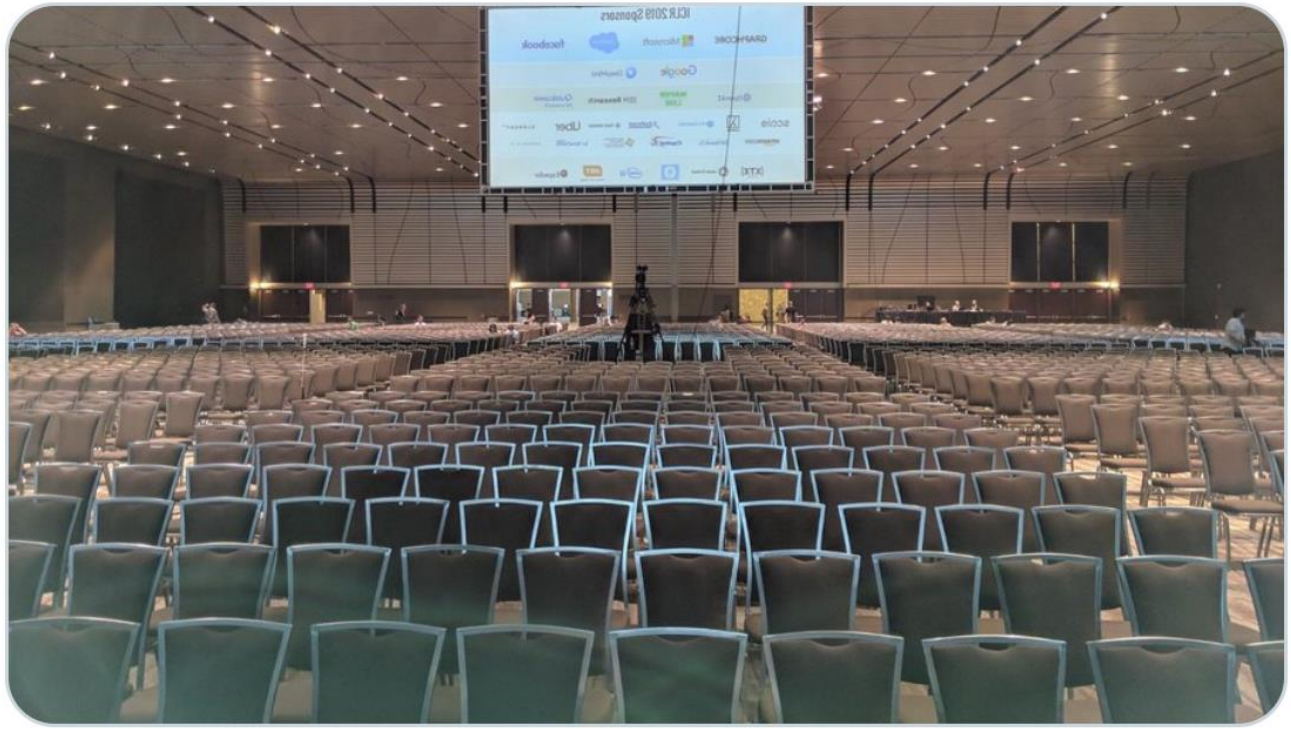
- Original GAN paper
- Uses only fully connected layers
  - Limited to generating small images
- Discriminator
  - Binary cross entropy loss

GANs

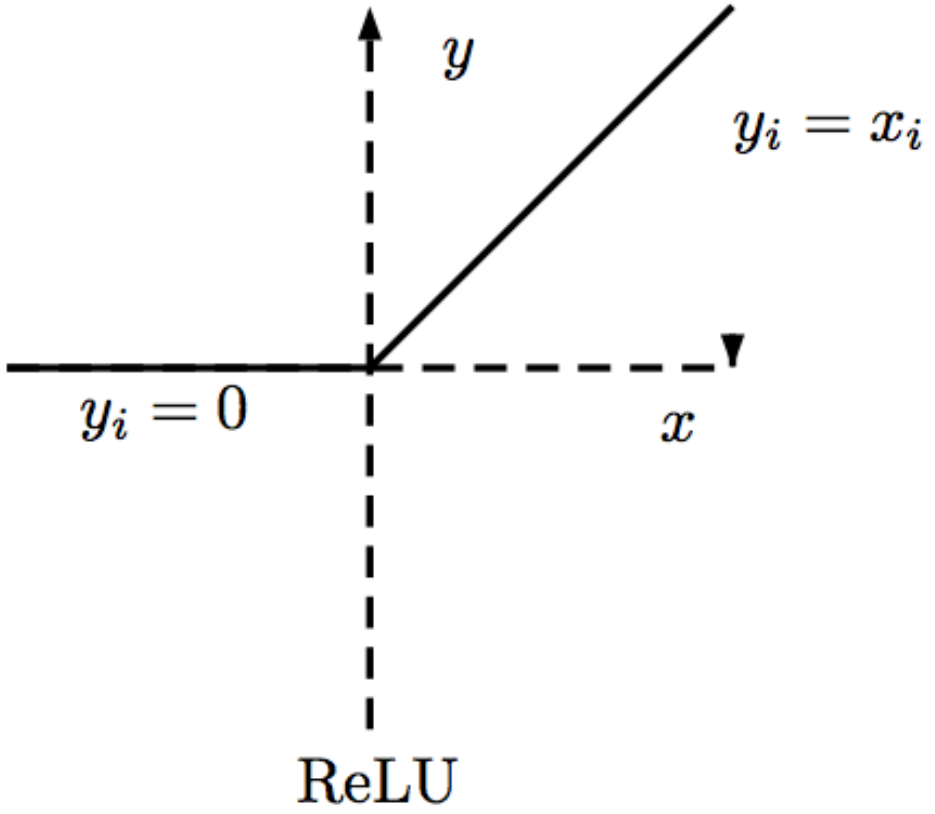
this is a joke slide

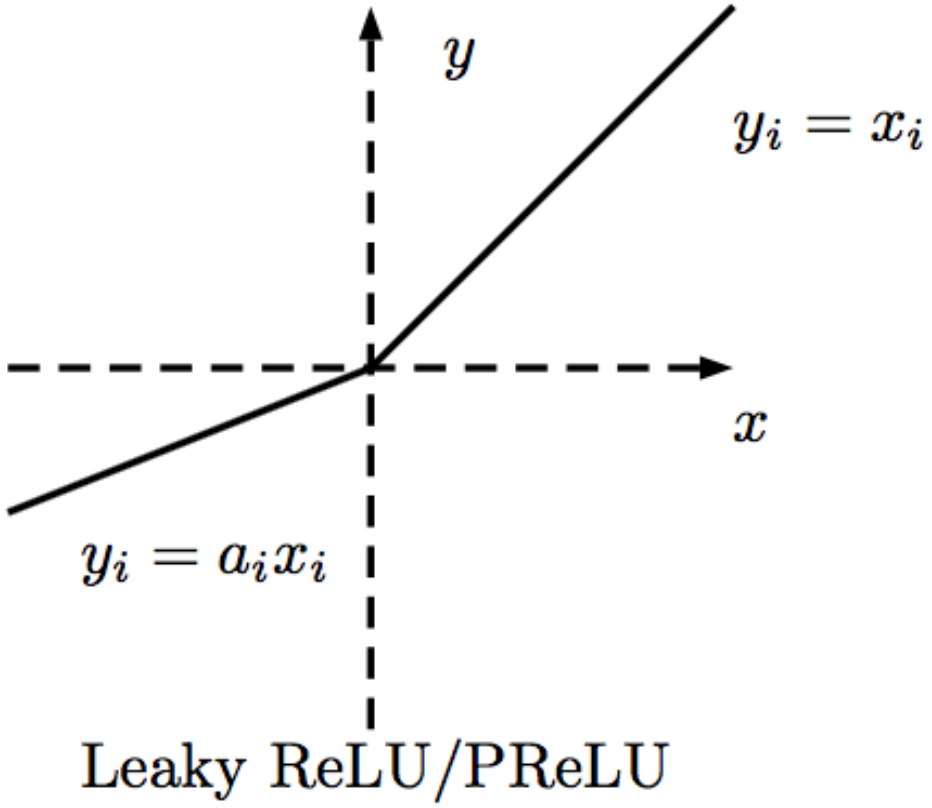
Ian Badfellow @badfellow_ian · May 7

Ready to blow the roof off this sucker. Free GANs for everyone in the first 10 rows!



5 17 327



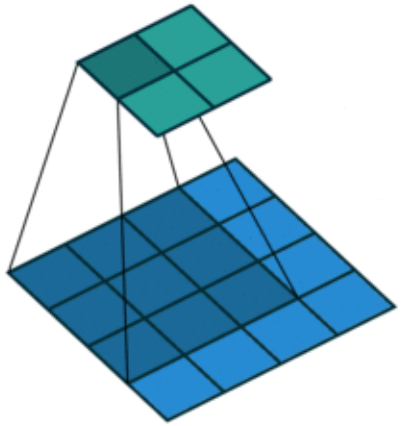


LeakyReLU

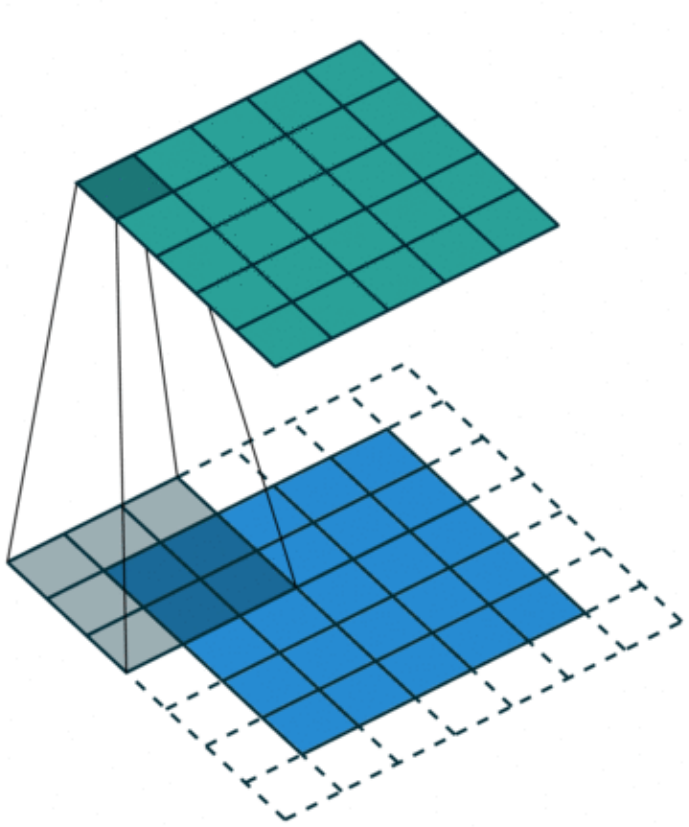
Often used with $\alpha = 0.2$

Convolution Recap – Blue is input

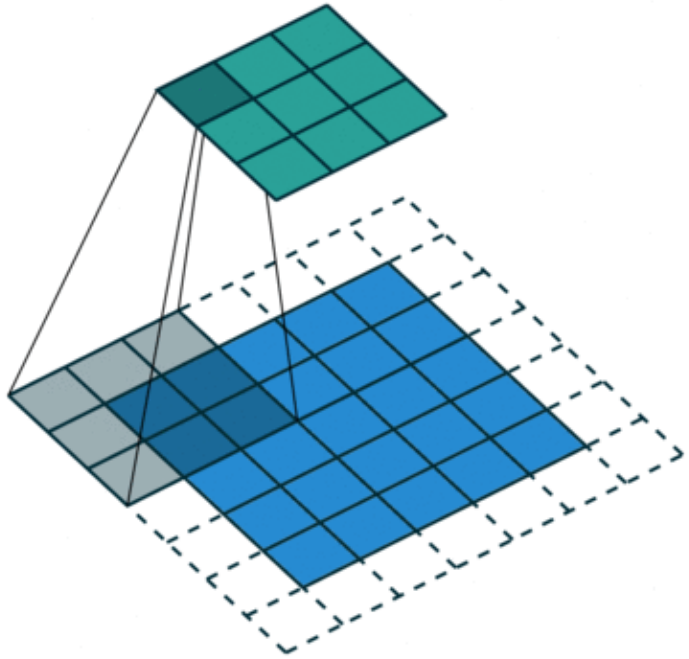
padding=0, stride=1



padding=1, stride=1



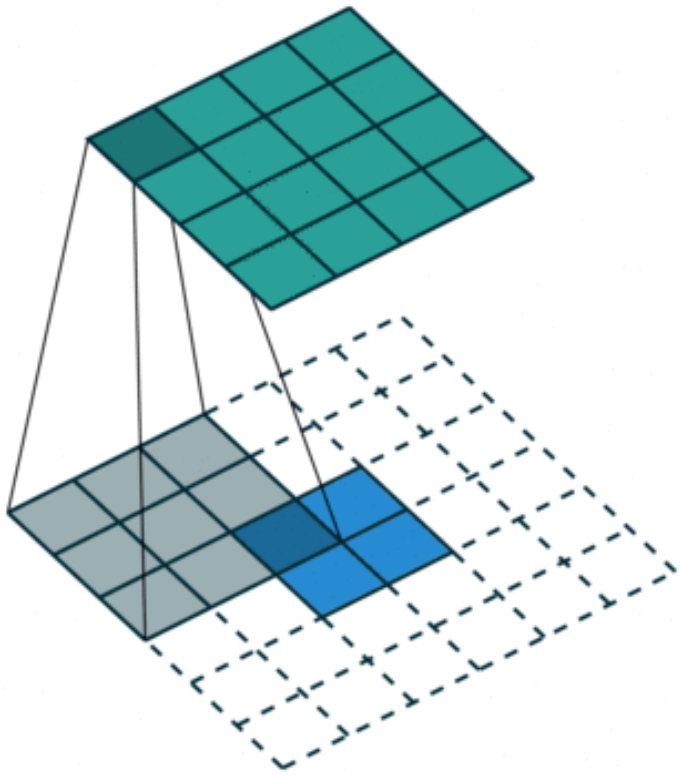
padding=1, stride=2



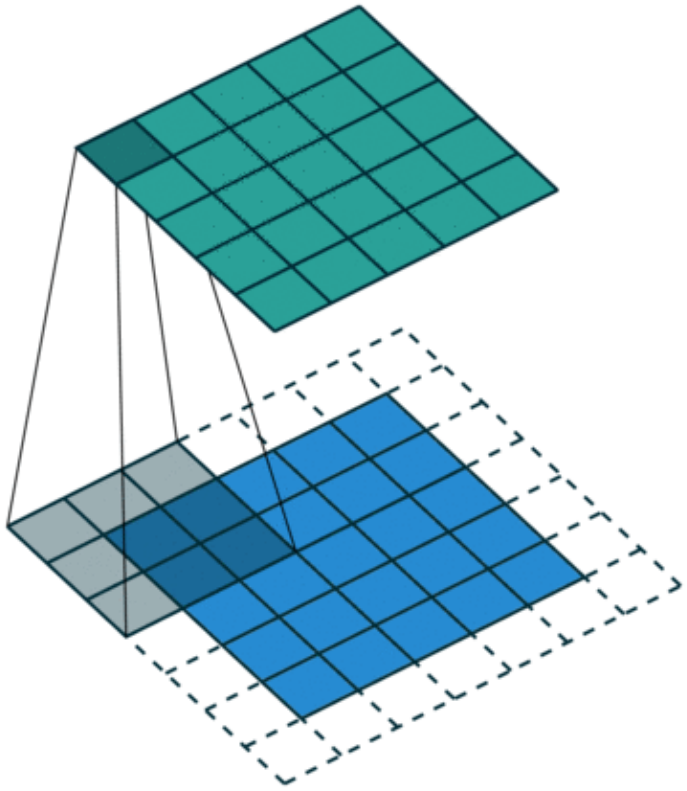
Transposed Convolution Recap – Blue is input

Also known as: fractionally strided convolution/deconvolution

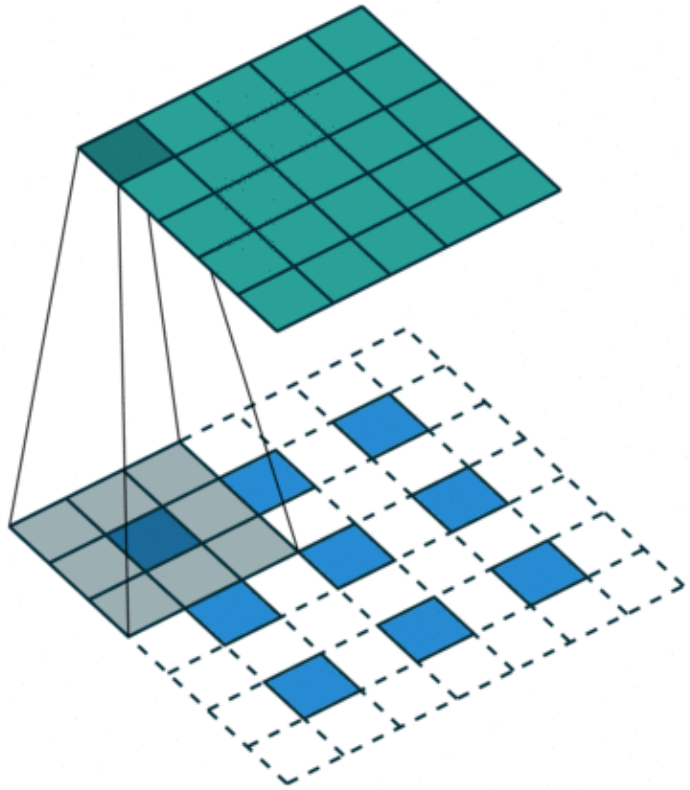
padding=0, stride=1



padding=1, stride=1



padding=1, stride=2



2015 - DCGAN

Radford et al. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*

- How to do GANs with convolutional layers?
  - Replace any pooling layers with strided convolutions (discriminator) and transposed-strided convolutions (generator)
  - Use batchnorm in both the generator and the discriminator
  - Use LeakyReLU activation in the discriminator for all layers
  - Use ReLU activation in generator for all layers except for the output, which uses Tanh
    - Later on people recommend using LeakyReLU in both G and D

General tips

Avoid sparse gradients (max pool, ReLU)

Use higher learning rate for discriminator (Two Time-Scale Update Rule)

Don't mix real and generated content in batches

Construct separate batches for real and generated content respectively

Don't assume you have a good training schedule

Visualize generated samples periodically.

GAN variations

Vanilla GAN

- GANs are a two player game
  - Which game do they play?
  - G tries to minimize
  - D tries to maximize
- Original loss:

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

For G → Rather than training G to minimize $\log(1 - D(G(z)))$ we can train G to maximize $\log D(G(z))$.

Minimizes the Jensen-Shannon divergence between p_d and p_g.

WGAN

Arjovsky et al. *Wasserstein GAN*
Gulrajani et al. *Improved Training of Wasserstein GANs*

- WGAN
  - Optimize approximation of Wasserstein-1 distance
  - Discriminator must be Lipschitz continuous
    - Otherwise it can push them arbitrarily far apart without becoming more discriminative
  - Introduce weight clipping in discriminator to enforce Lipschitz continuous
    - "Weight clipping is a clearly terrible way to enforce a Lipschitz constraint"

      -Original WGAN paper
- WGAN-GP
  - Enforce Lipschitz continuous D by penalizing on norm of gradients in D

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

LSGAN

Mao et al. *Least Squares Generative Adversarial Networks*

- Uses a least square loss instead
- Simple to implement

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$

$a=-1, b=1$ and $c=0$ minimizes Pearson χ^2 distance between $p_d + p_g$ and $2p_g$.
$a=0, b=1$ and $c=1$ is also a good choice.

GAN-SN

Miyato et al. *Spectral Normalization for Generative Adversarial Networks*

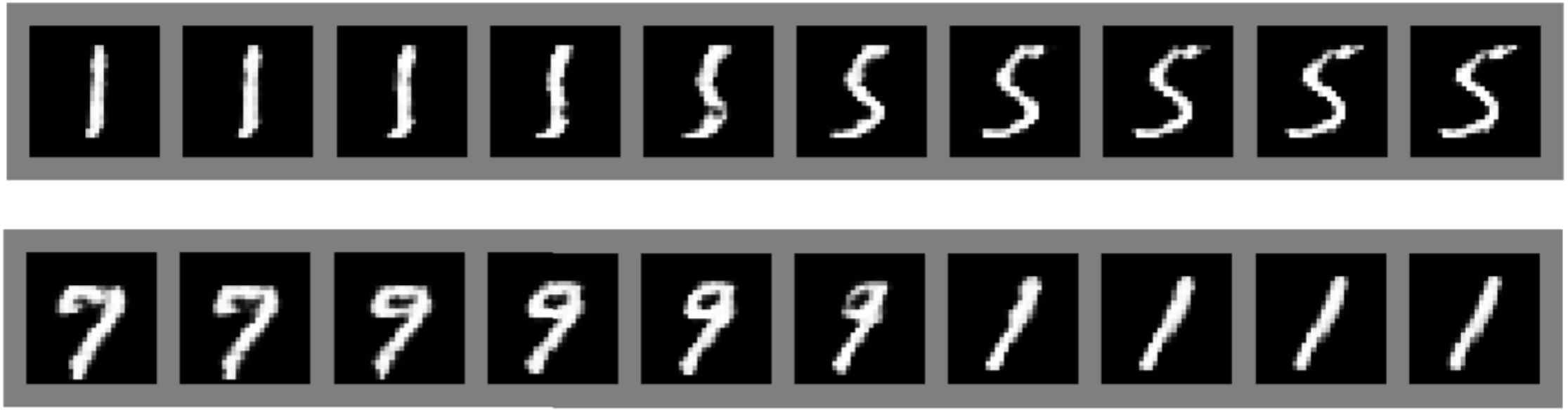
- Spectral Normalization
  - Normalization to prevent vanishing or exploding gradients
  - Enforces Lipschitz continuity in computationally efficient way
- Normalize each layer by it’s spectral norm
  - For a fully connected layer the spectral norm is the largest eigenvalue of the weight matrix
  - Computed efficiently with the power method
- Originally applied only to D but works well for G and D

Important note

- You should not have an activation function on the last layer of your discriminator when using WGAN or LSGAN
  - The discriminator should be able to output any value

Latent space interpolations

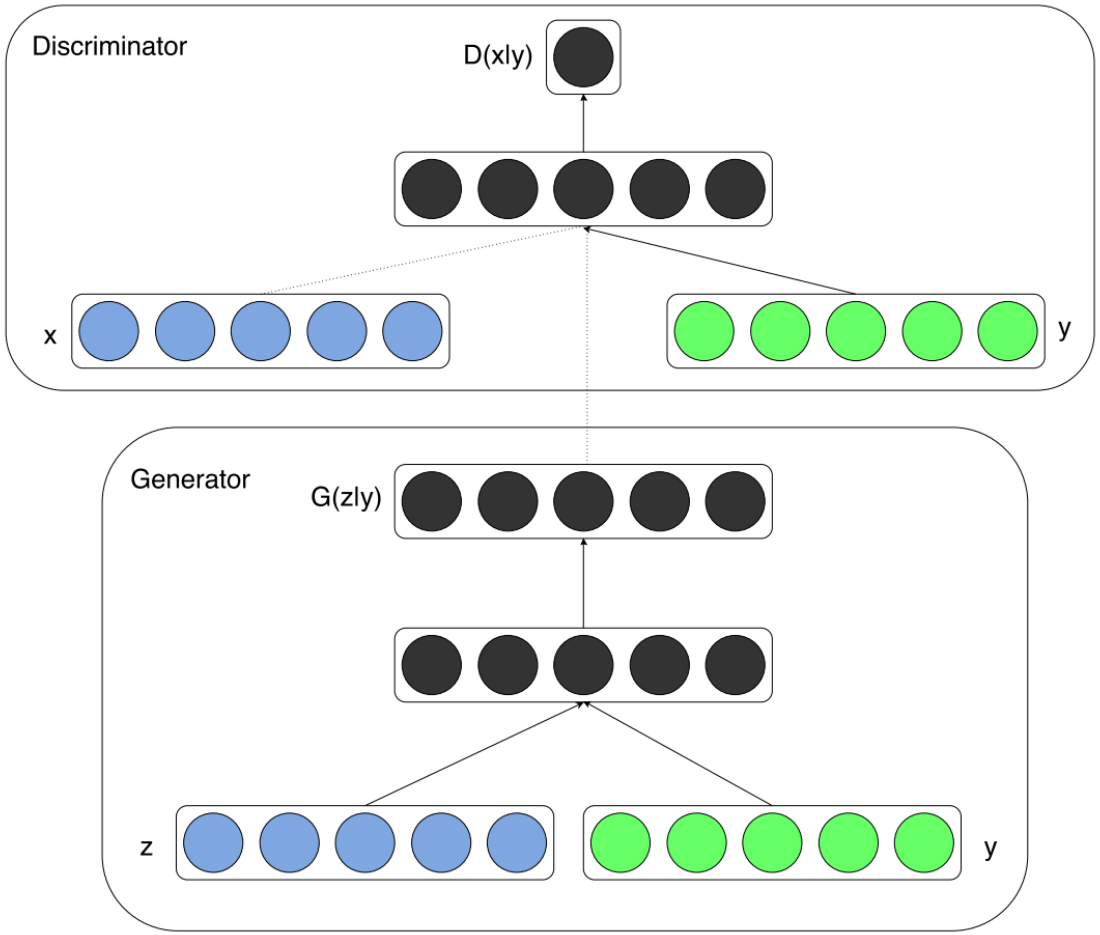
- Showing examples generated by moving between two datapoints in the latent space



cGAN

Mirza et al. *Conditional Generative Adversarial Nets*

- Conditional GAN
- Conditions the generated image on additional information (y)
  - e.g. class information
    - Can also condition on image



Abbreviations

- GAN: Generative Adversarial Network
- DCGAN: Deep Convolutional Generative Adversarial Network
- CGAN: Conditional Generative Adversarial Network
- WGAN: Wasserstein Generative Adversarial Network
  - WGAN-GP: Wasserstein GAN – Gradient Penalty

- Not covered here:
- CoGAN: Coupled GAN
- SAGAN: Self-Attention Generative Adversarial Networks
- ProGAN: Progressive Growing of GANs

Well known networks

- Later today:
- Pix2pix
- CycleGAN

- Not covered here (but you have seen examples):
- SRGAN
  - ESRGAN
- BigGAN
- StyleGAN

StyleGAN example

Karras et al. *A Style-Based Generator Architecture for Generative Adversarial Networks*

Thanks for listening

- A generator learns a mapping from one probability distribution to another
  - Often used to output images
- Training GANs is hard
  - Many different tricks have been employed