Generative Adversarial Networks

June course: Deep Learning in Computer Vision

Morten Hannemose, mohan@dtu.dk

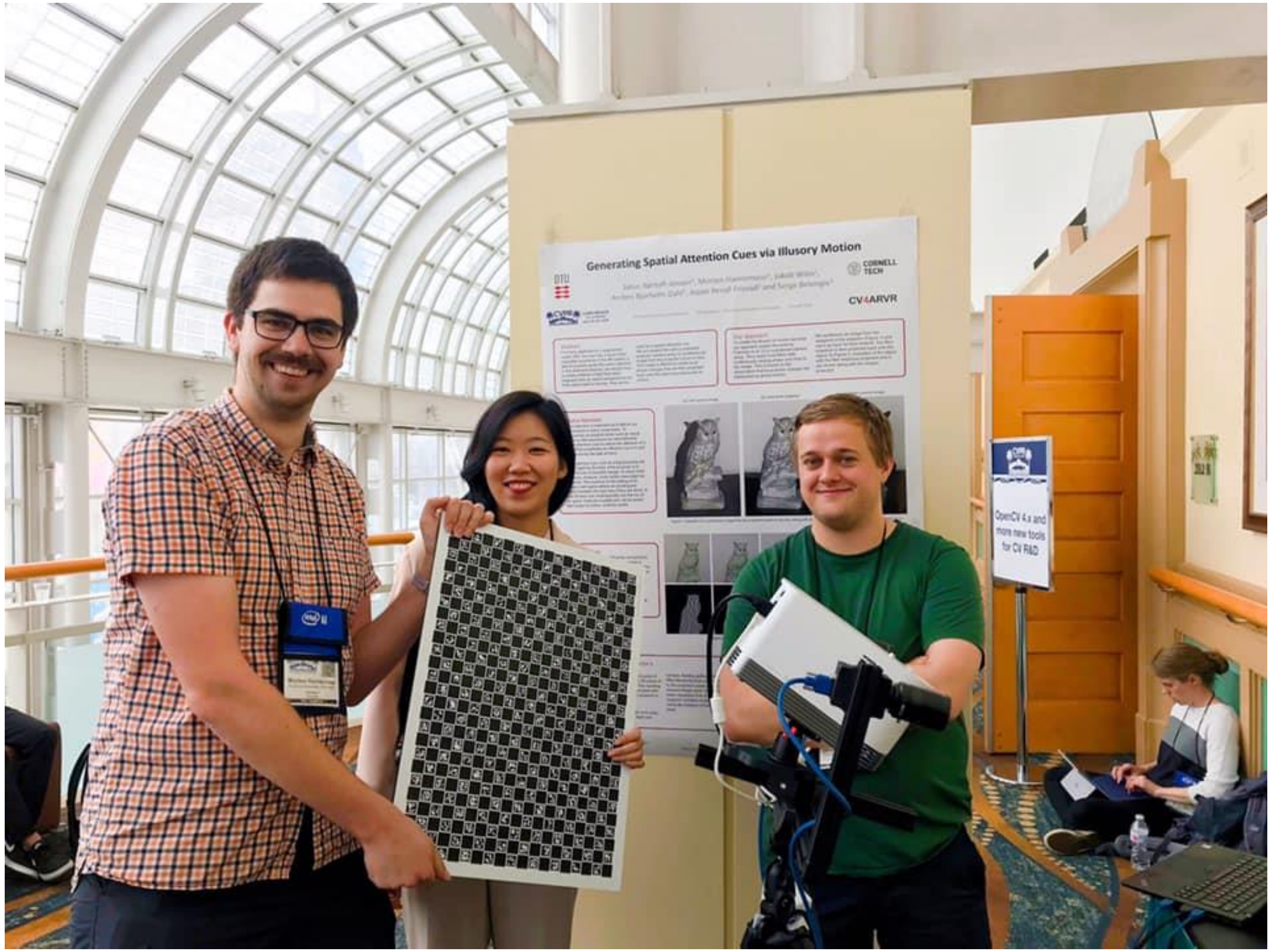
June 24th, 2019

Program for the day

- Lecture 9:00
- Exercise
- Lecture 13:00 (about assignment)
- Continue exercise

Where have I been? - CVPR

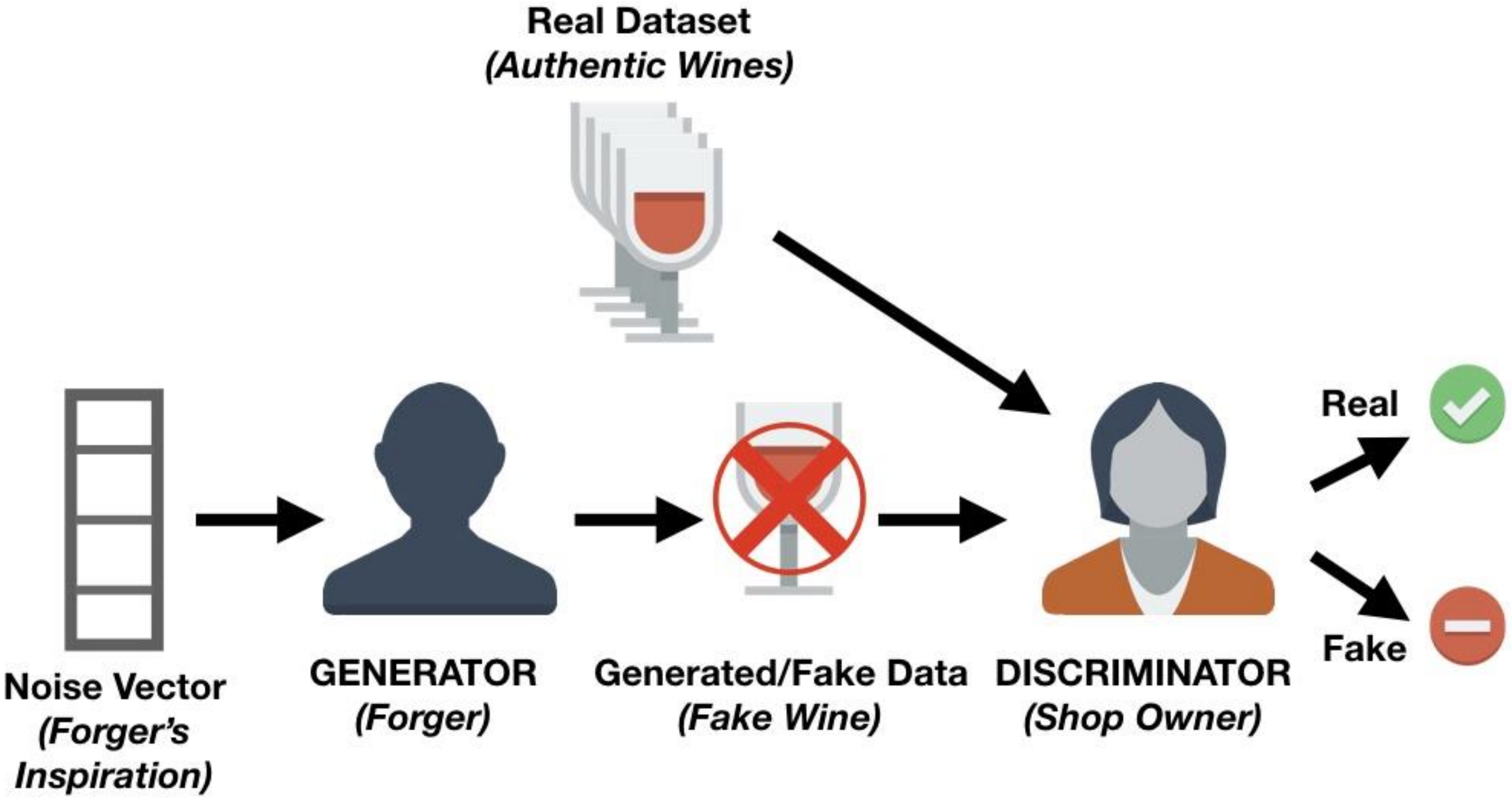
- 9300 participants
- Many cool papers



Outline – What you're going to see

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

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

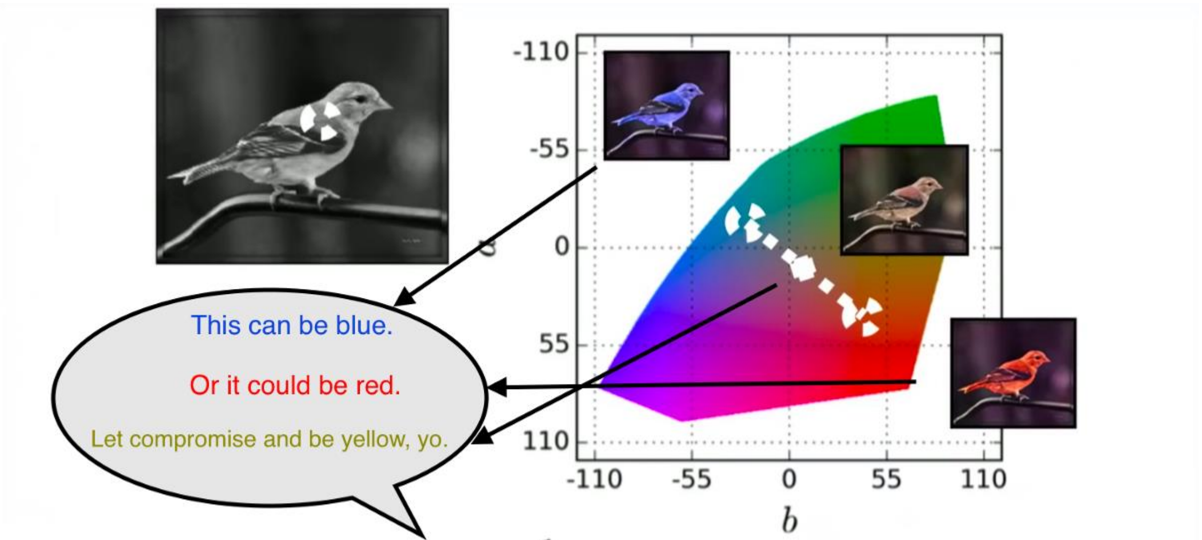
Why?

- 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



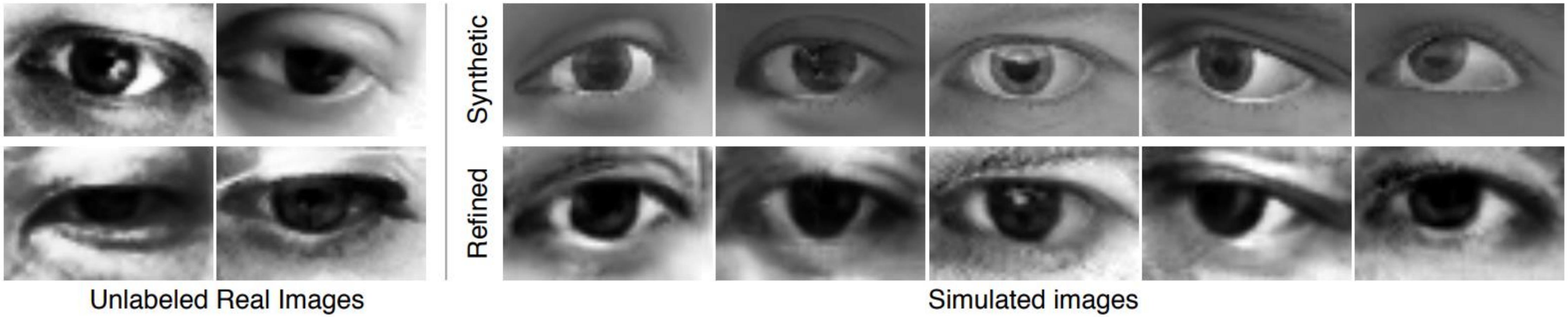
$$L_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \| \mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w} \|_2^2$$

Applications

- Super resolution
- Colorization
- Inpainting
- Domain-transfer
- Generating additional training data

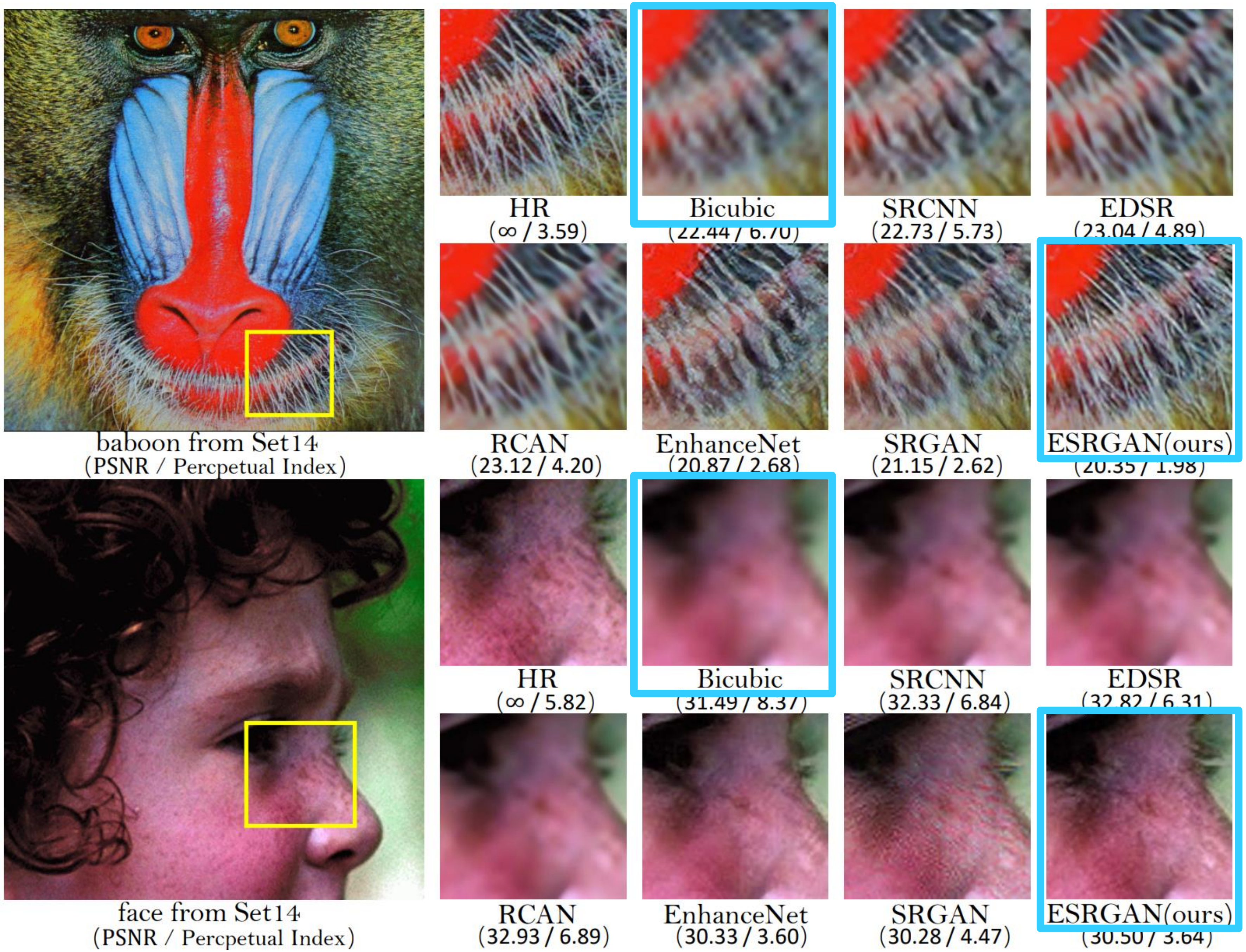
Generating additional training data

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

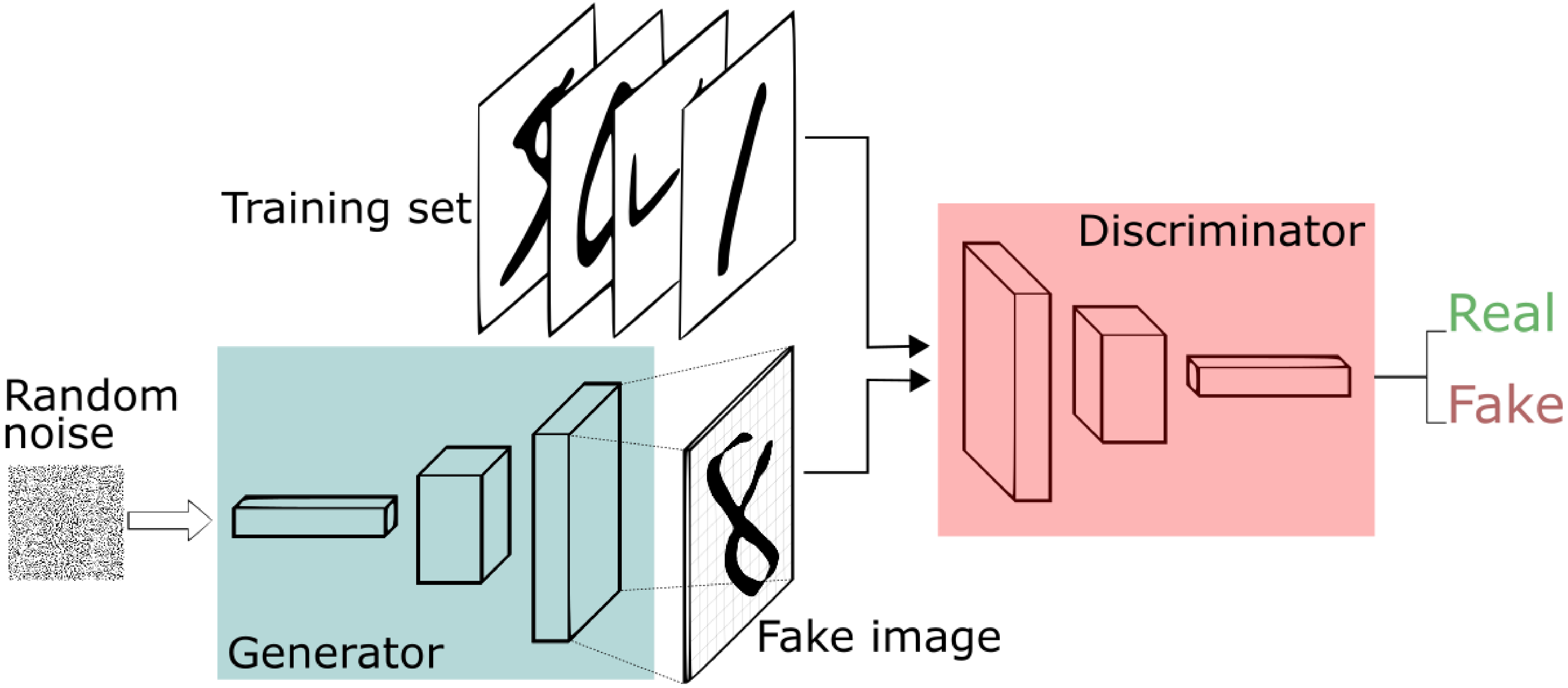


[2016] Ashish Shrivastava et al. Learning from Simulated and Unsupervised Images through Adversarial Training
https://arxiv.org/abs/1612.07828

Super resolutiuon 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*

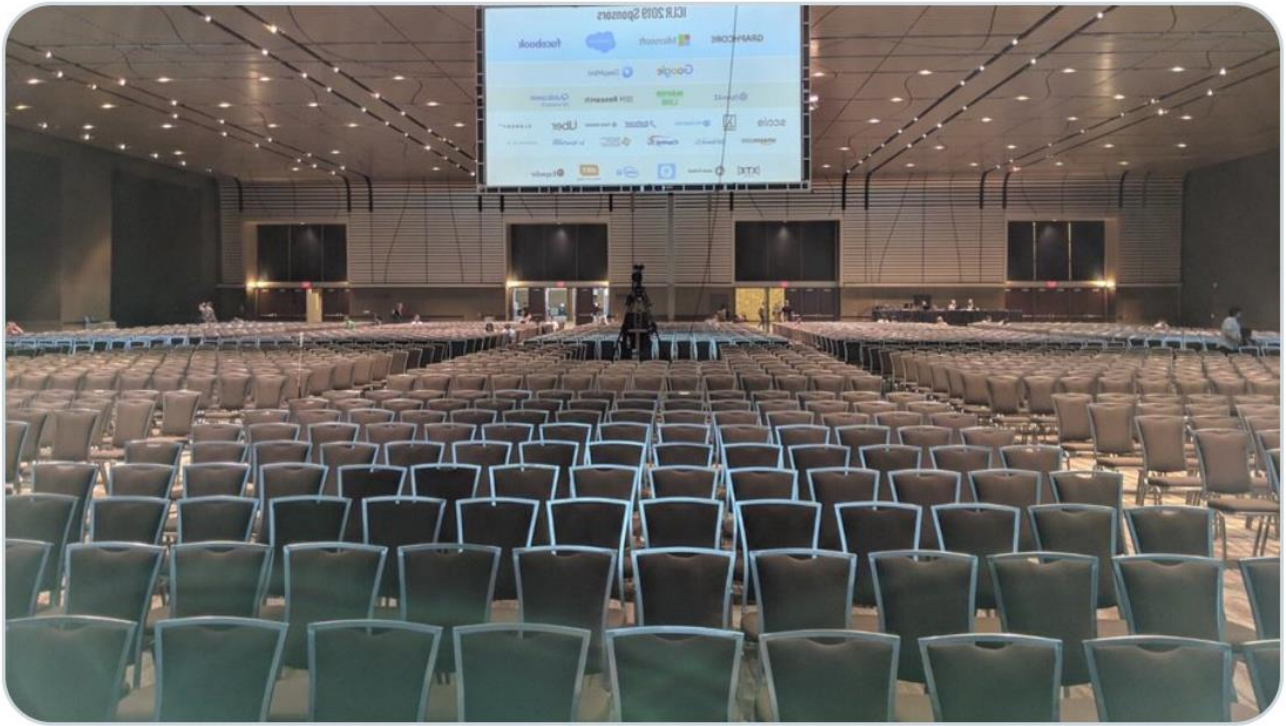
- Original GAN paper
- Uses only fully connected layers
  - Limited to generating small images
- Discriminator
  - Binary cross entropy loss

GANs

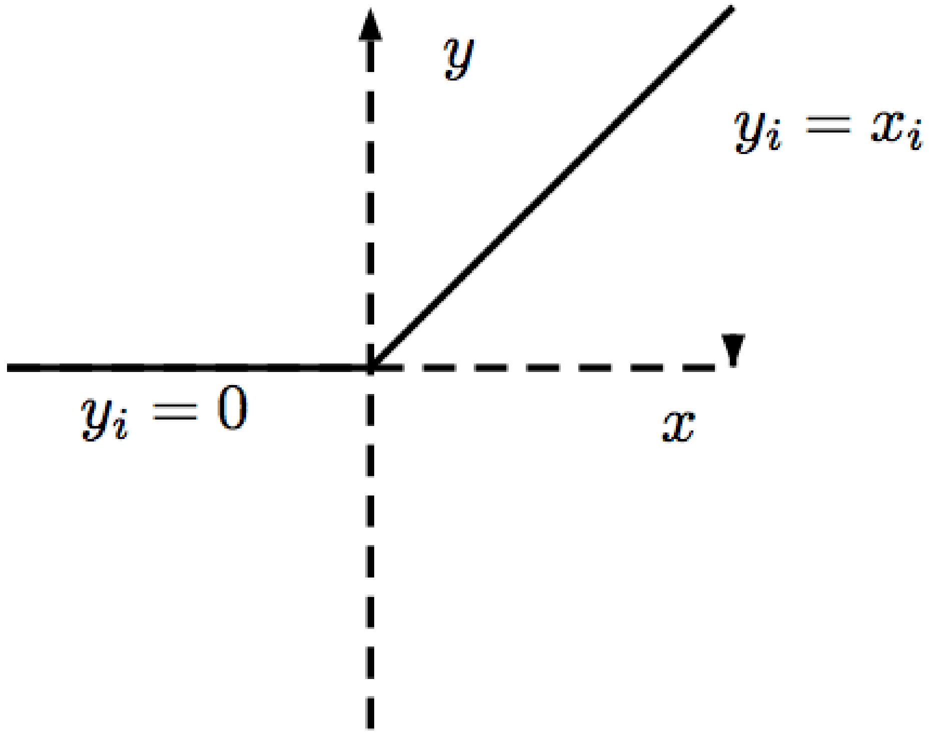
this is a joke slide

Ian Badfellow @badfellow_ian · May 7

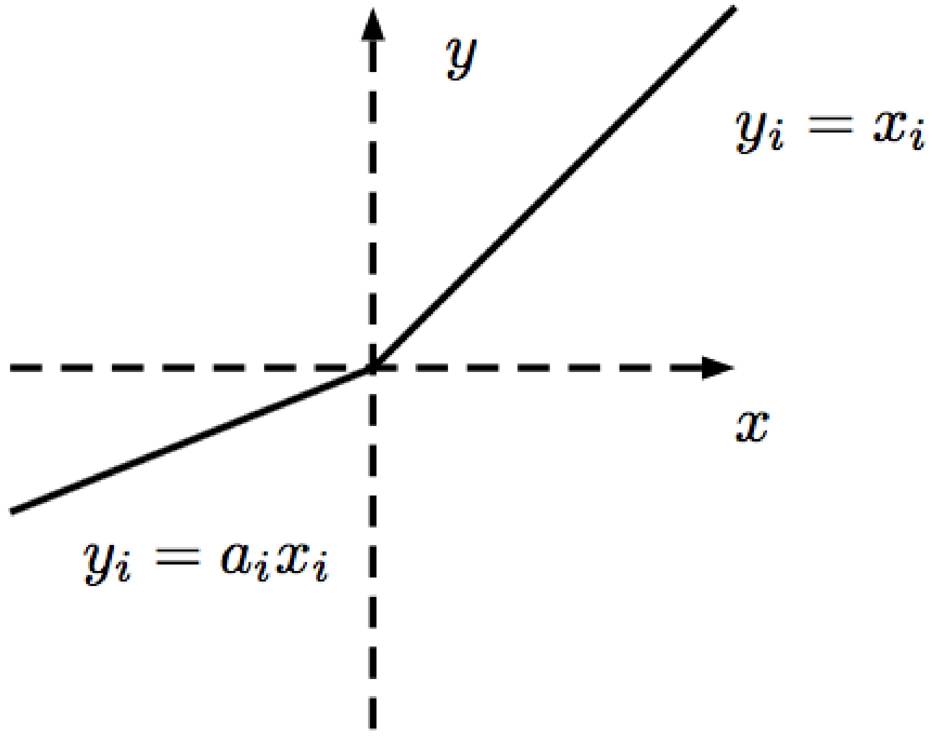
Ready to blow the roof off this sucker. Free GANs for everyone in the first 10 rows!



5 17 327



ReLU



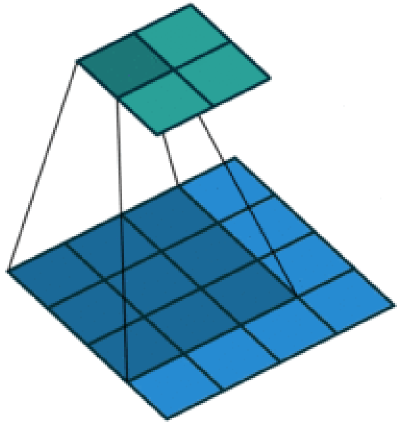
Leaky ReLU/PReLU

LeakyReLU – remember this guy?

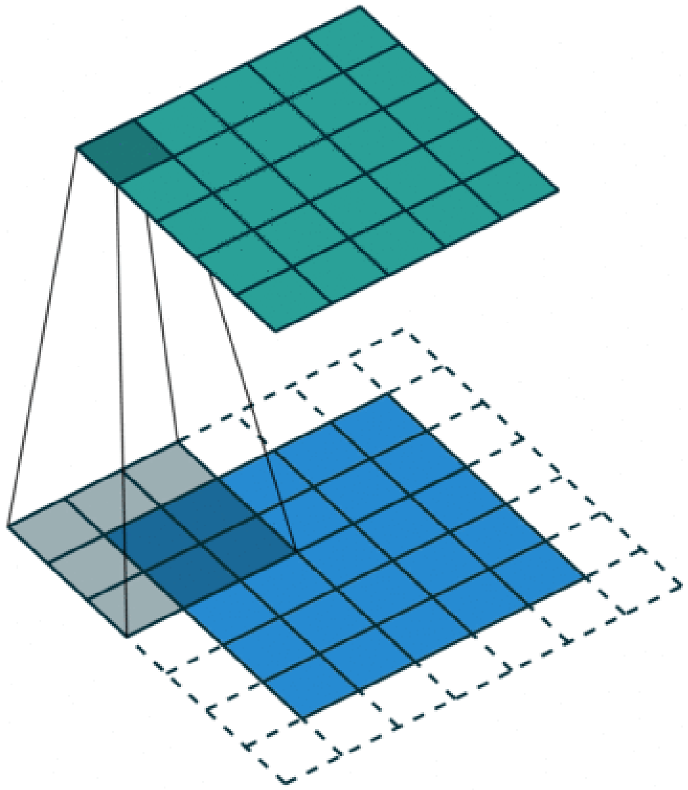
Often used with $\alpha = 0.2$

Convolution Recap – Blue is input

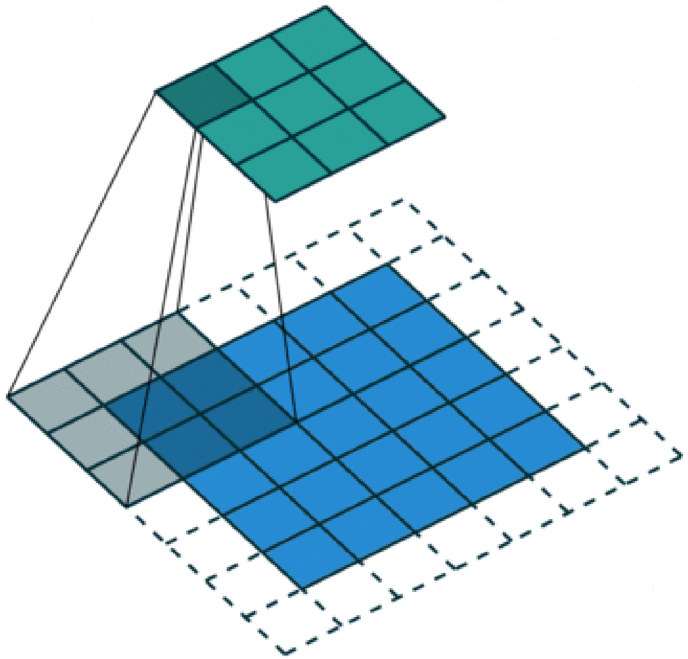
padding=0, stride=1



padding=1, stride=1



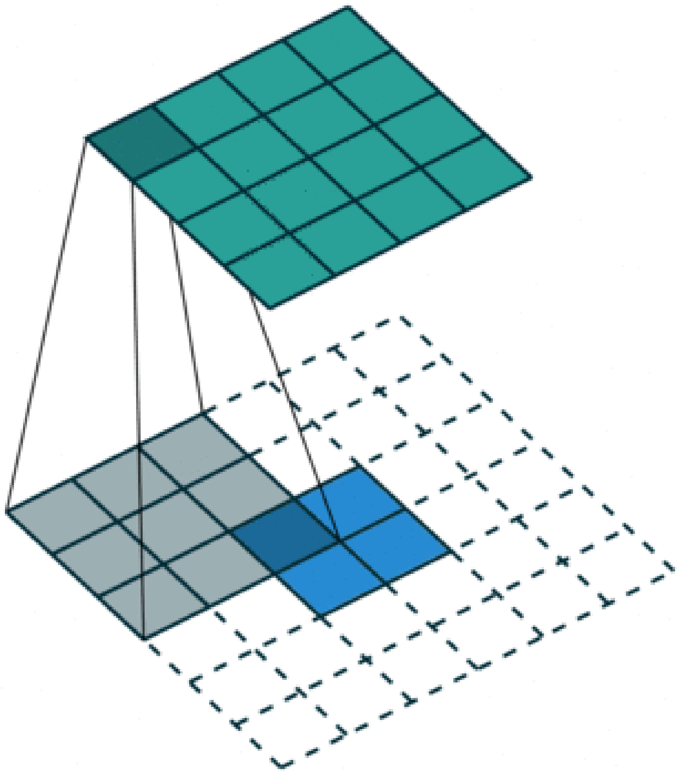
padding=1, stride=2



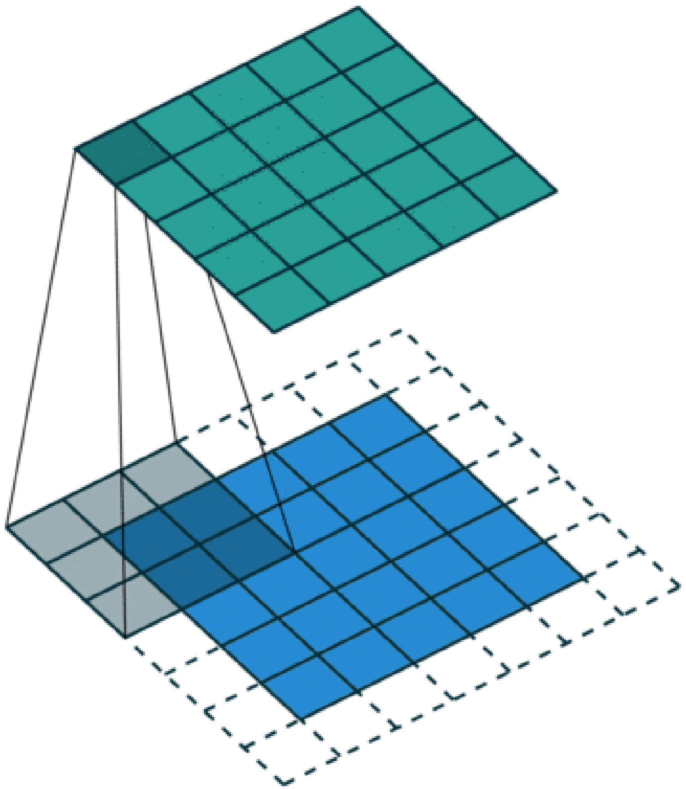
Transposed Convolution Recap – Blue is input

Also known as: fractionally strided convolution/deconvolution

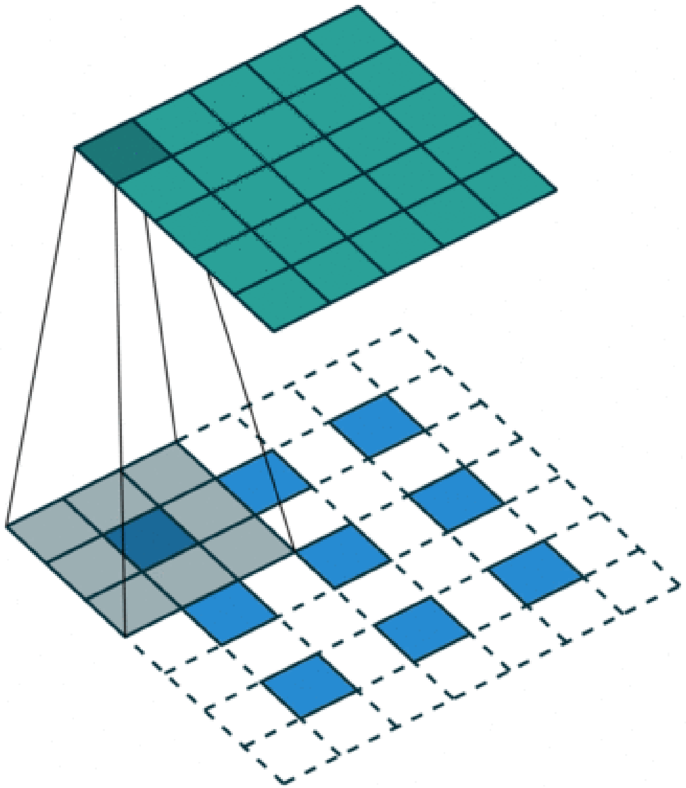
padding=0, stride=1



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

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 variants

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

LSGAN

Mao et al. *Least Squares Generative Adversarial Networks*

- Uses a least square loss instead
- Simple to implement

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

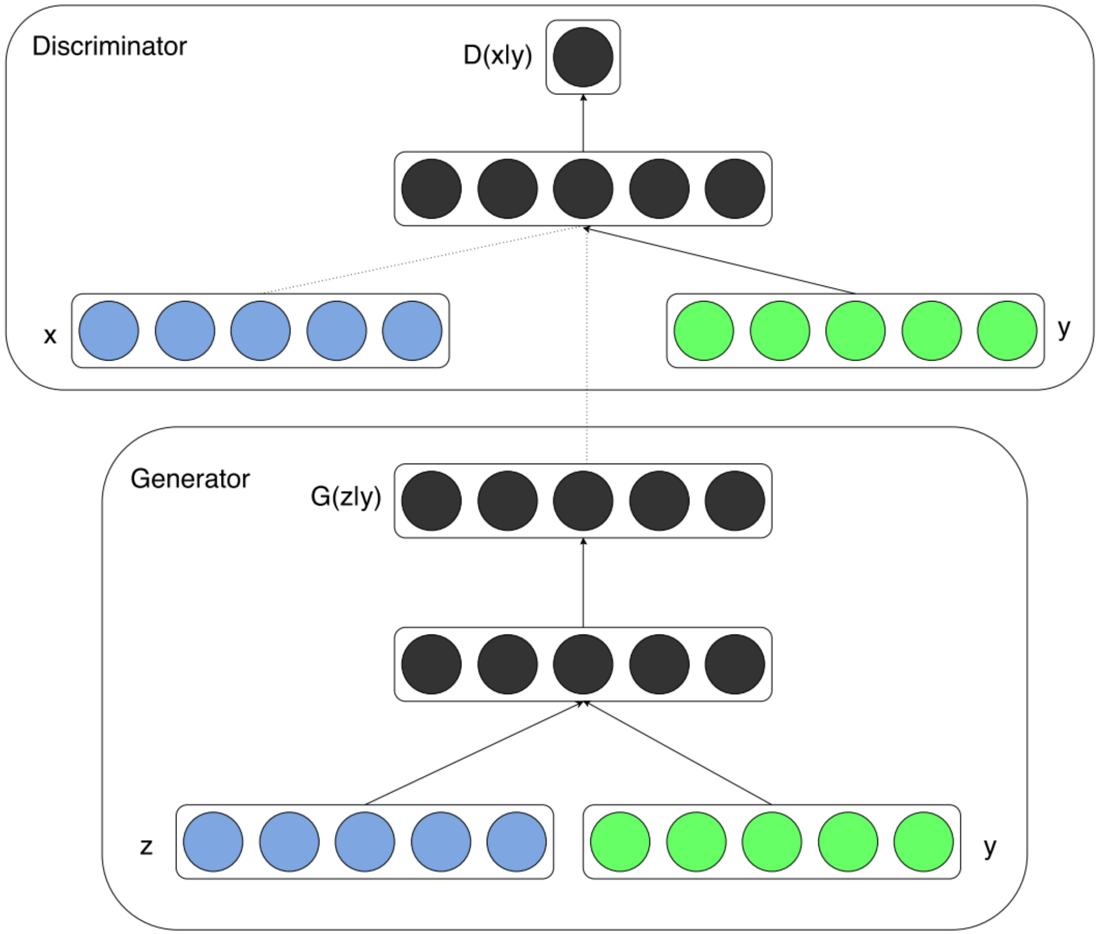
$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - c)^2\right],$$

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

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 in this course:
- CoGAN: Coupled GAN
- SAGAN: Self-Attention Generative Adversarial Networks
- ProGAN: Progressive Growing of GANs

Good networks

- Pix2pix
- CycleGAN

- Not covered in this course (but you’ll see examples):
- SRGAN
    - ESRGAN
- BigGAN
- StyleGAN
- MUNIT

StyleGAN example

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

Exercise now

- Generate MNIST digits using a vanilla GAN
- Additional tasks:
    - Implement another loss
    - Convert your network to a DCGAN
    - Generate images from FashionMNIST
    - Convert your architecture into a cGAN
    - Hard: Create a cGAN model to convert from SVHN to MNIST