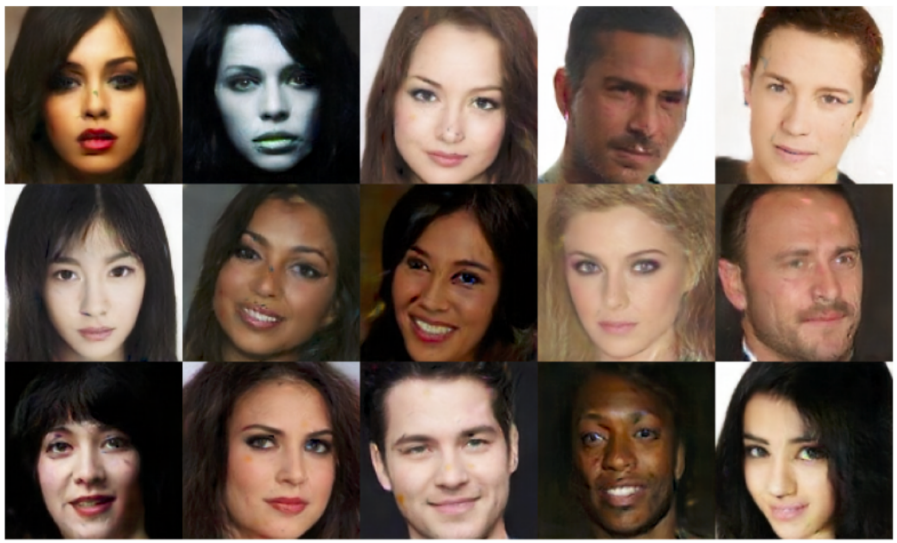
COMPSCI 689
Lecture 22: Generative Adversarial Networks

Benjamin M. Marlin

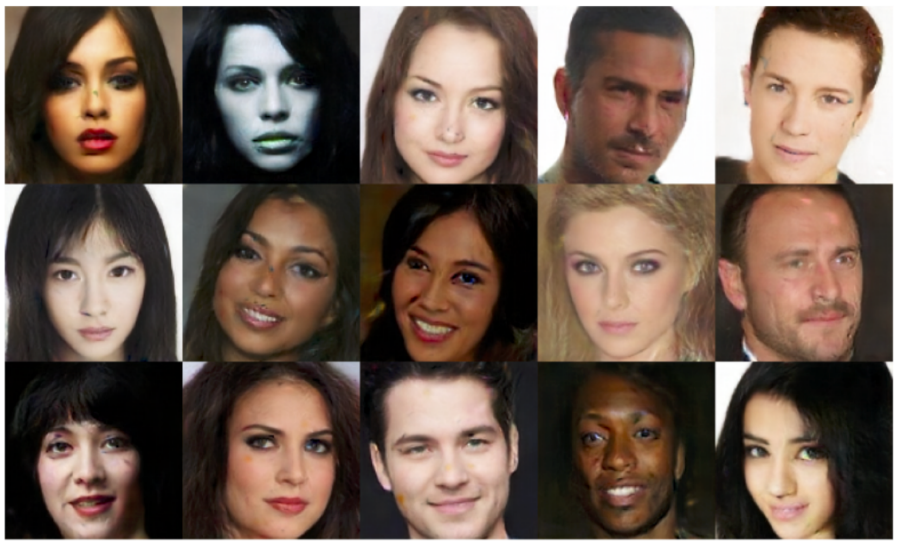
College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

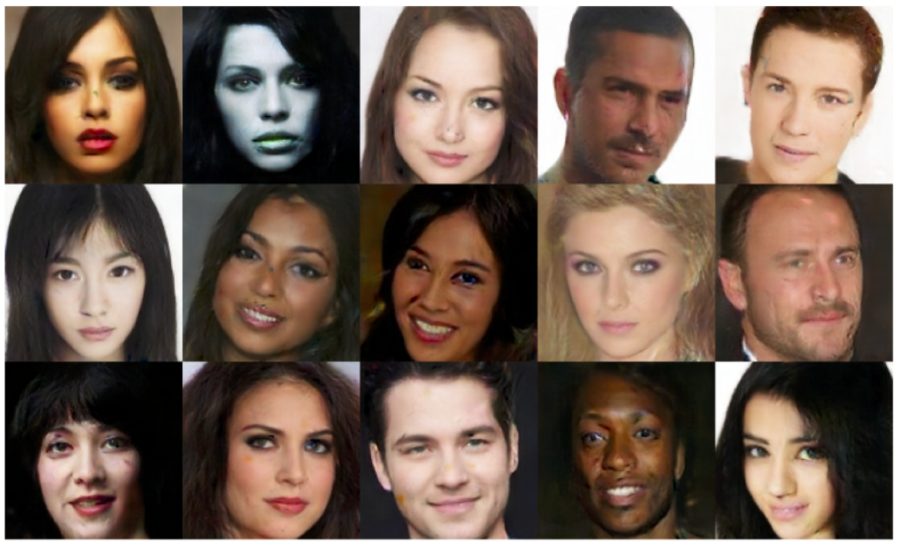
Which Images are Real?

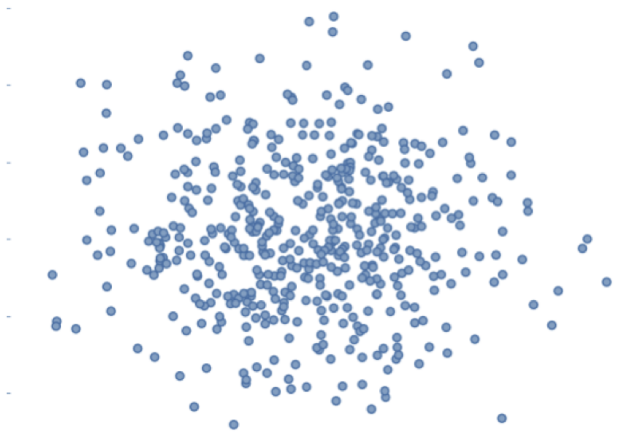


Which Images are Real?



Which Images are Real?

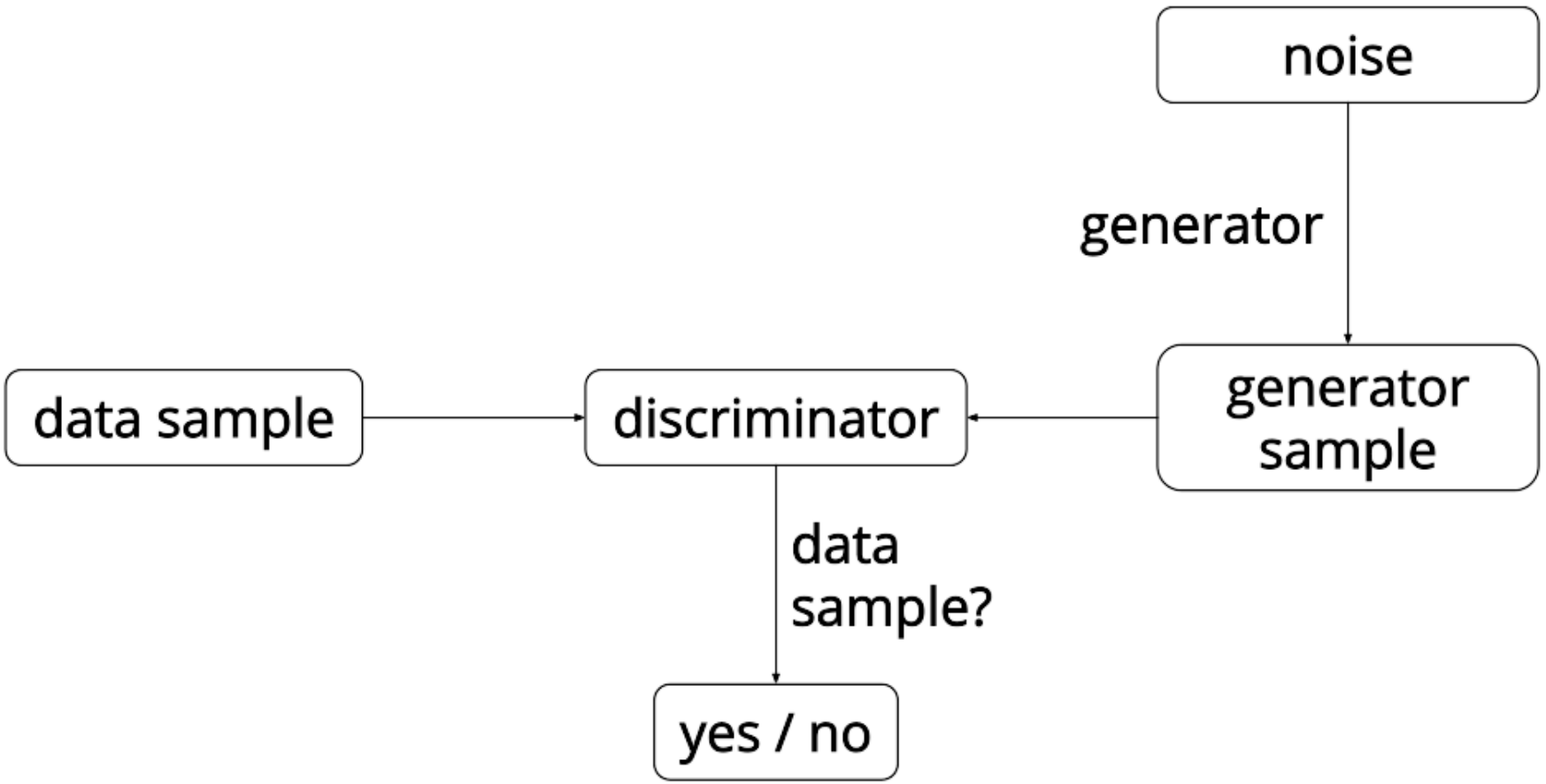




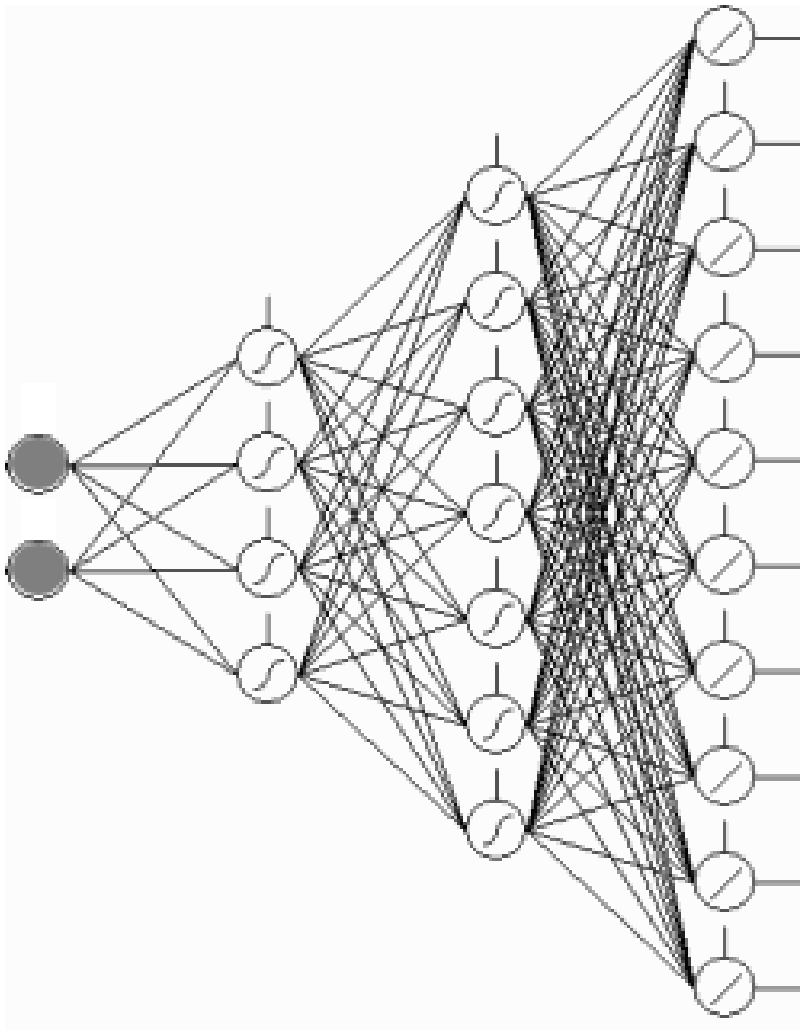
Generative Adversarial Networks

- Generative adversarial networks are an approach to learning generative models for complex data like images.
- GANs seek to avoid the typical problems associated with learning complex hierarchical generative models by transforming random noise into samples from the model.
- GANs do not provide an explicit likelihood over the data space, and they require optimization to infer latent variables/codes.
- They are sometimes referred to as *implicit* probabilistic models.

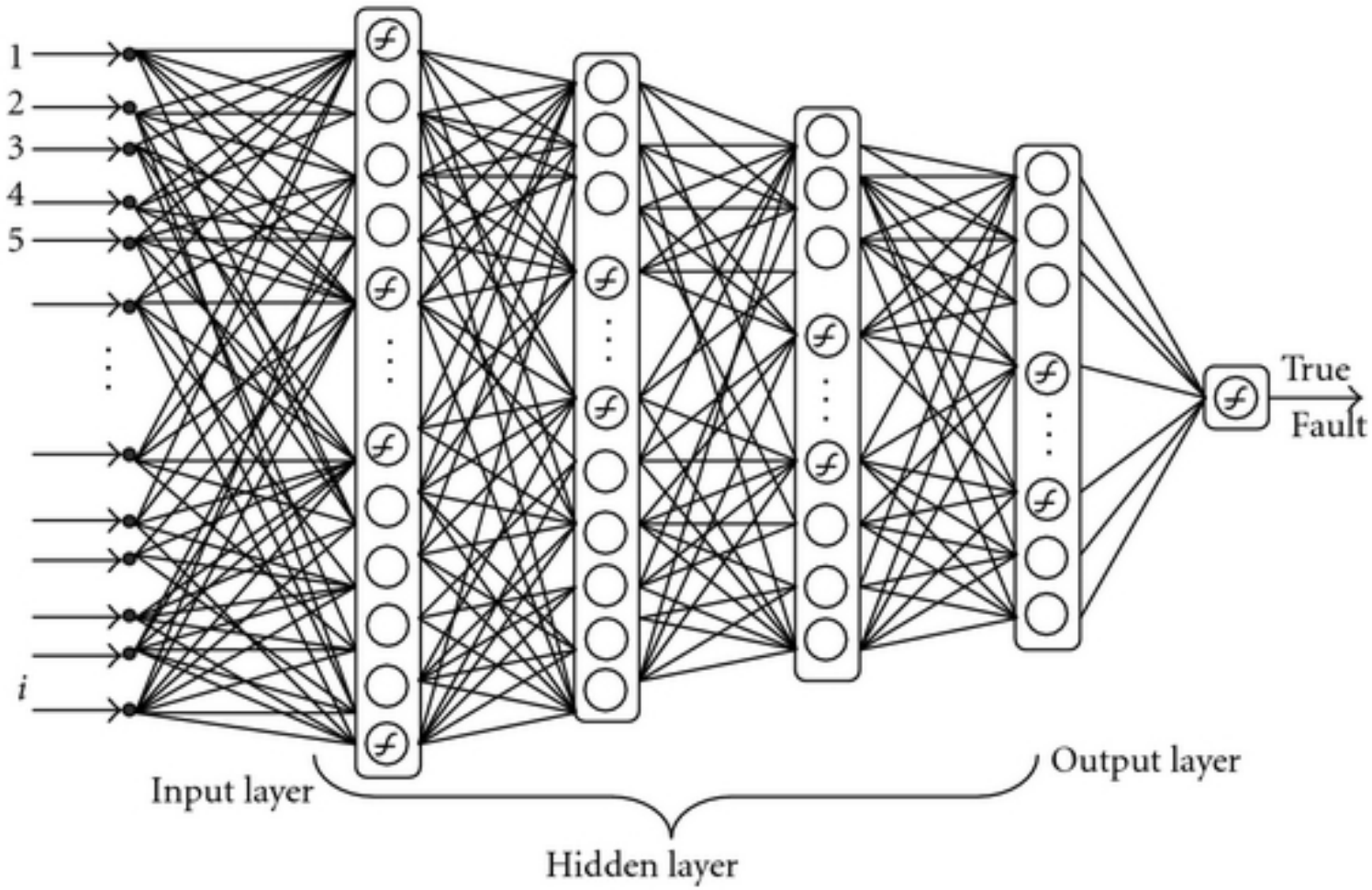
GAN Overview



GAN Generator



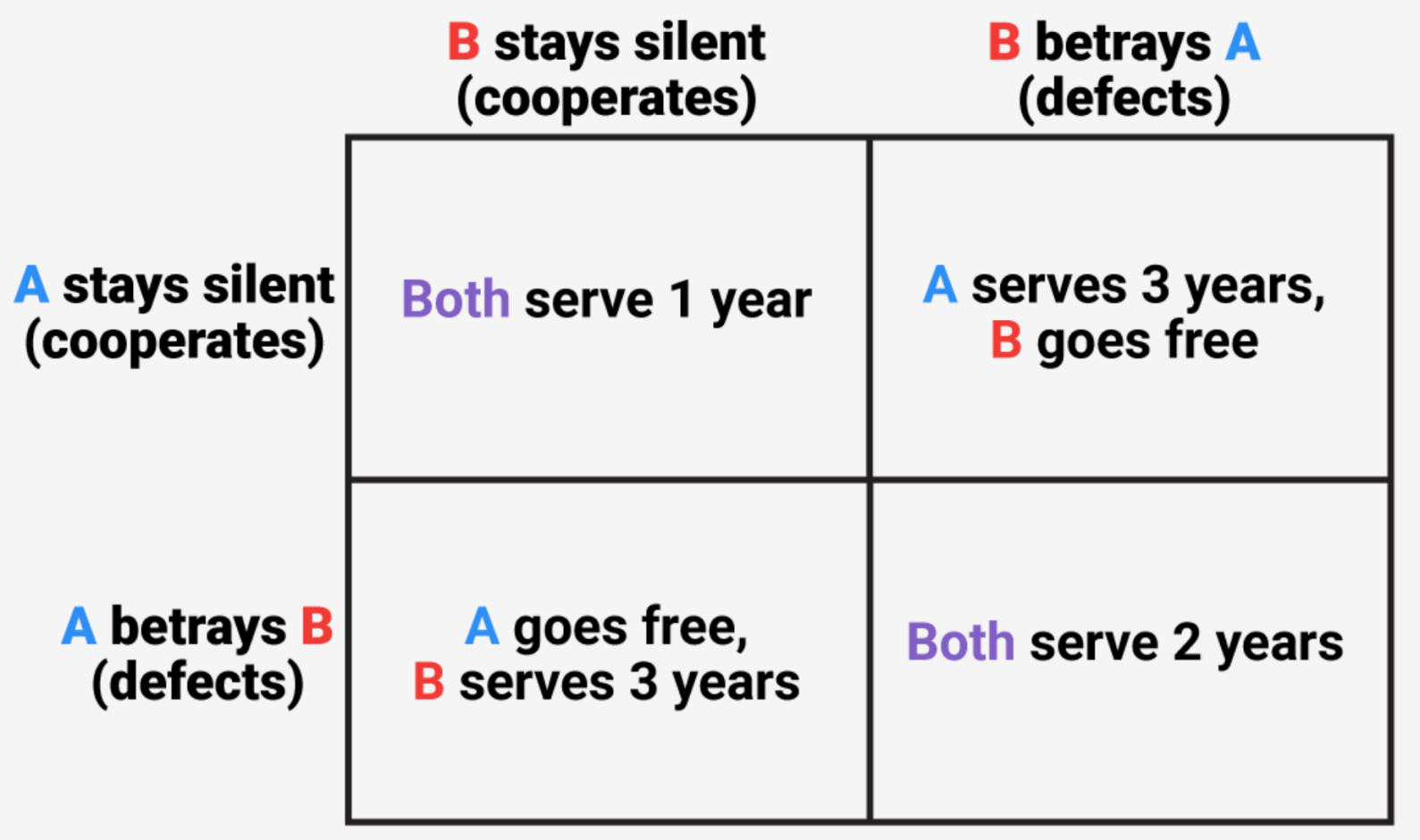
GAN Discriminator



GAN Learning

- The job of the the generator is to generate data samples from the model.
- The job of the discriminator is to tell real data samples from generated samples.
- The learning procedure for GANs does not follow any of the principles we've seen to date.
- The learning problem is framed as a *game* between the generator and discriminator.

Game Theory: Example



GAN Learning

- The value function for the discriminator in the classical GAN formulation is given below. The generator receives the negative of this function as its reward.

$$\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- This function is basically the cross entropy between the true labels indicating whether data examples are real or generated and the corresponding probabilities that data examples are real or generated according to the discriminator.
- The larger this value is, the better the discriminator is at telling generated data from real data. The smaller it is, the better the generator is at fooling the discriminator.

GAN Learning

- Learning then takes the form of solving a minimax game as shown below.

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- Solutions to the game take the form of parameters for the discriminator and the generator that are in *equilibrium*.
- If no player can improve their value by changing strategies while the other players keep theirs unchanged, then the current set of strategies and the corresponding payoffs constitutes a classical (Nash) equilibrium.

GAN Learning

- Learning algorithms for GANs try to reach such an equilibrium by alternately maximizing over the discriminator parameters and minimizing over the generator parameters.
- The optimization problem for the discriminator is a standard binary classification problem:

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- The optimization problem for the generator is to minimize the number of times the discriminator is correct:

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

GAN Learning

- Due to issues with the scaling of the gradients during learning, having the generator maximize the number of times the discriminator is incorrect works better in practice:

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

GAN Learning Algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

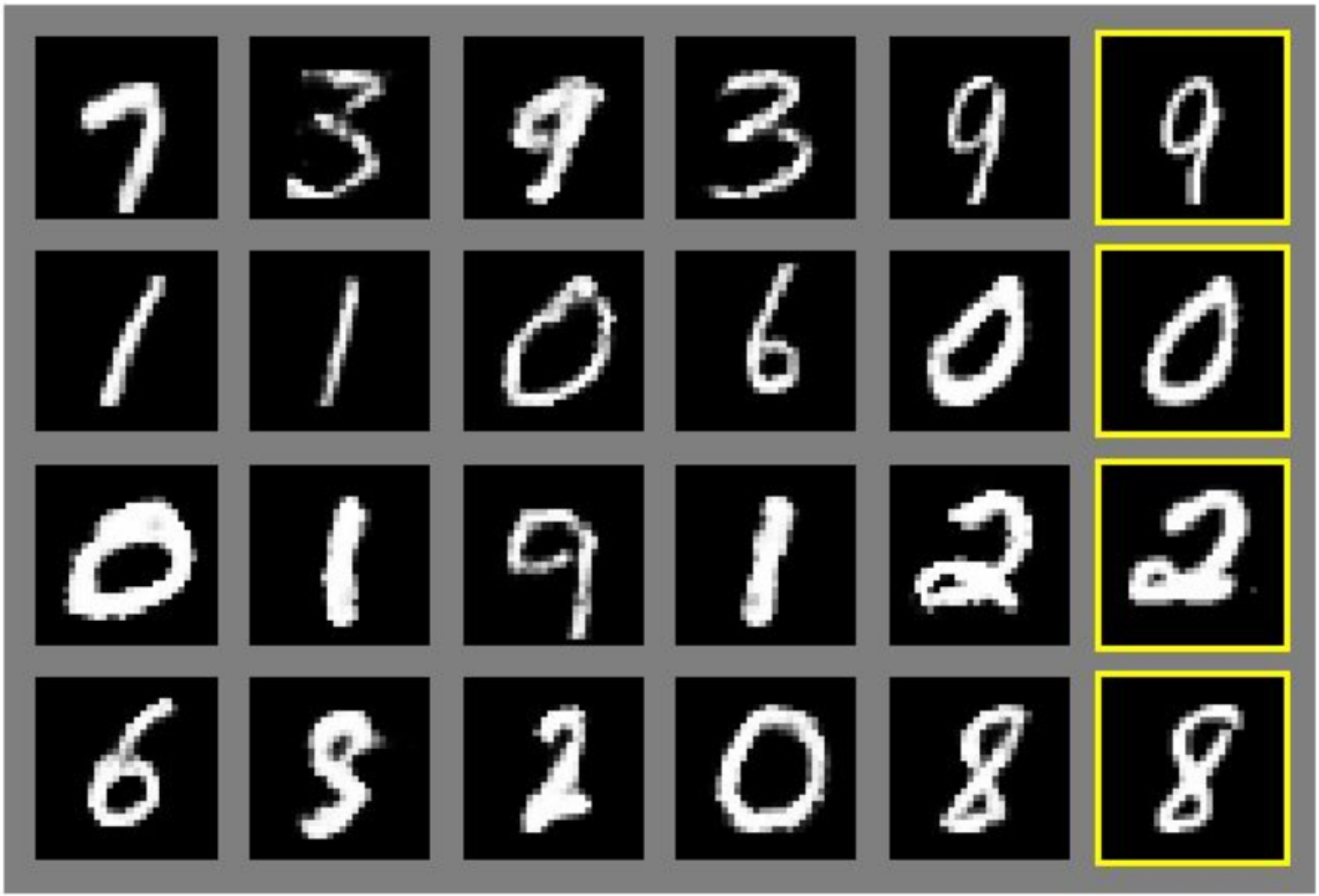
end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

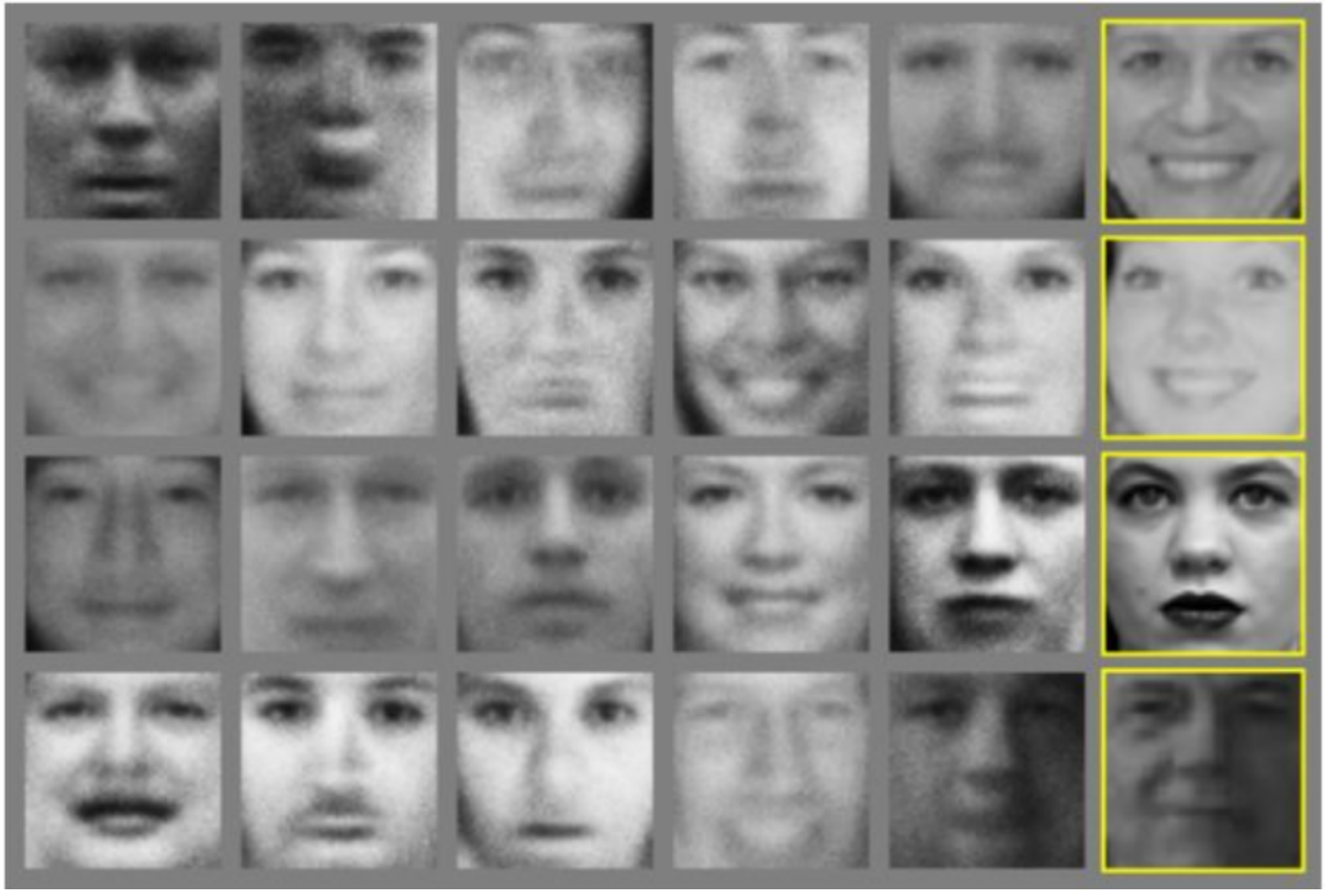
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Basic GAN Results: Digits



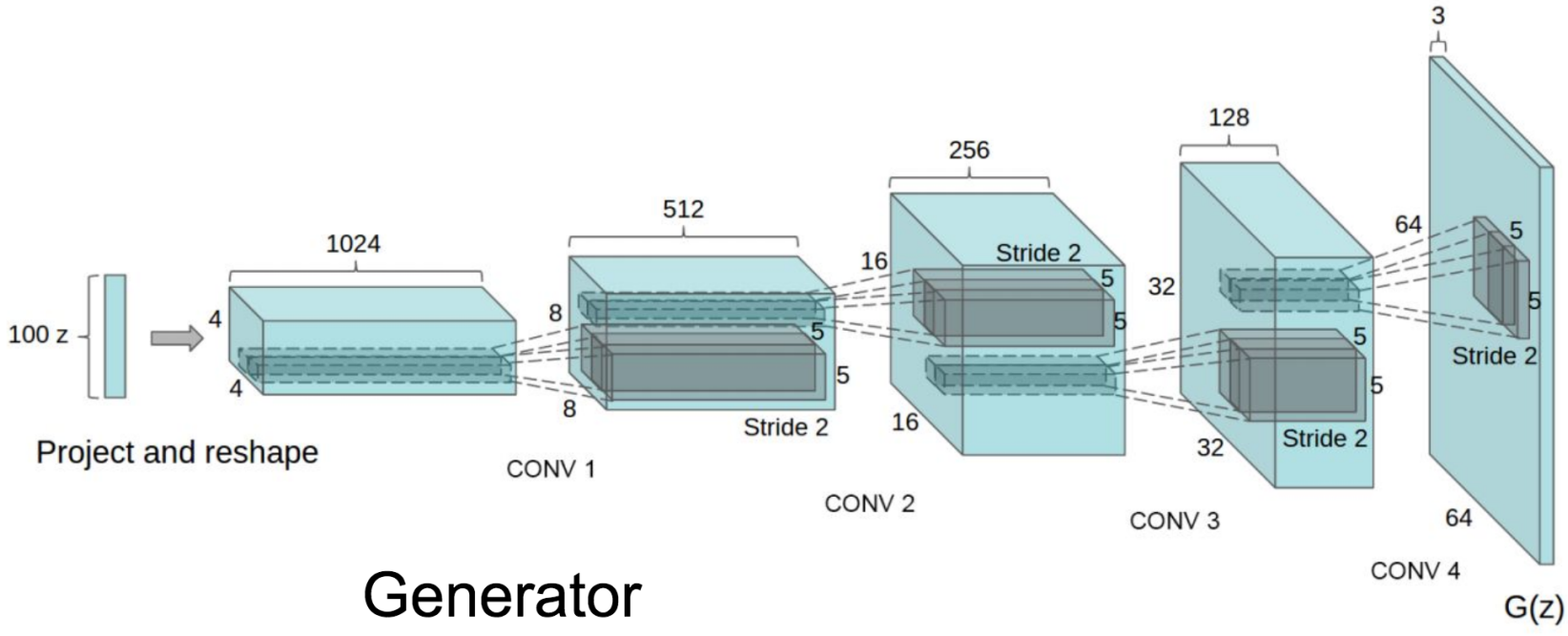
Basic GAN Results: Faces



Improving GANs for Images

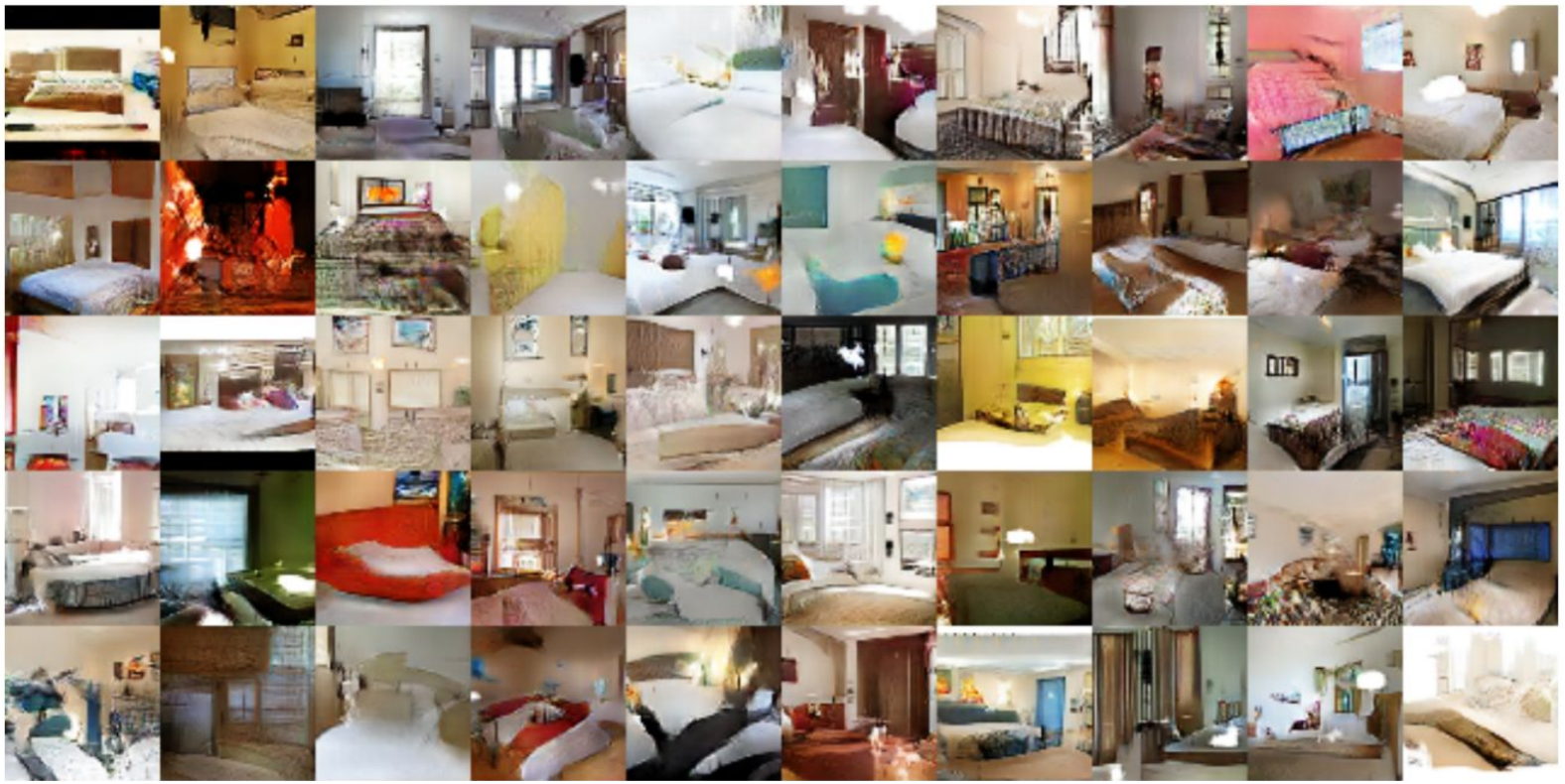
- To build a better GAN for images, we can leverage state-of-the-art deep (de-)convolutional architectures.
- The DC-GAN and BE-GAN are two different convolutional GAN architectures that have been fine-tuned to generate (small) photo-realistic outputs.

DC-GAN Architecture

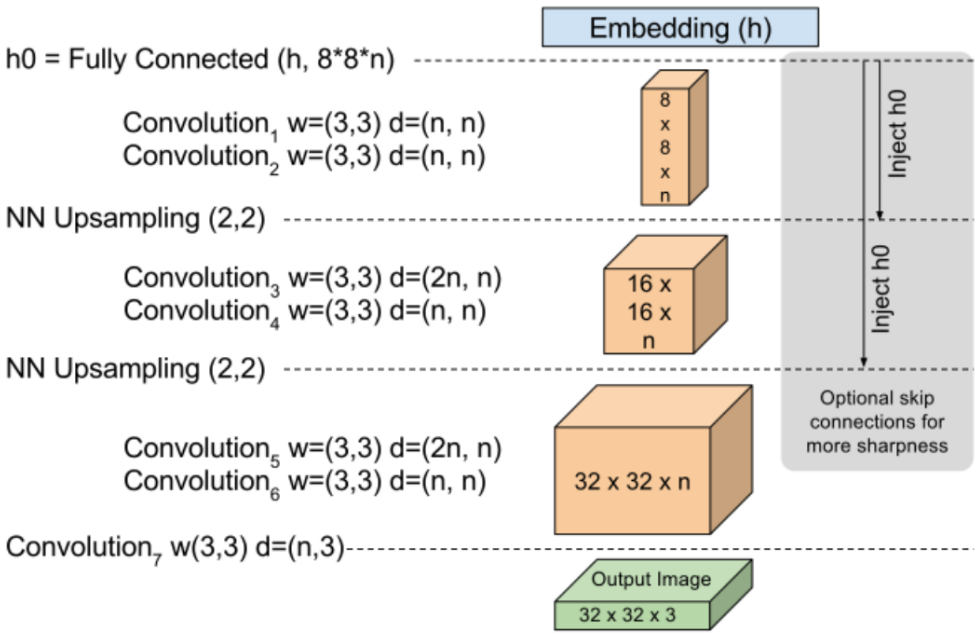


Generator

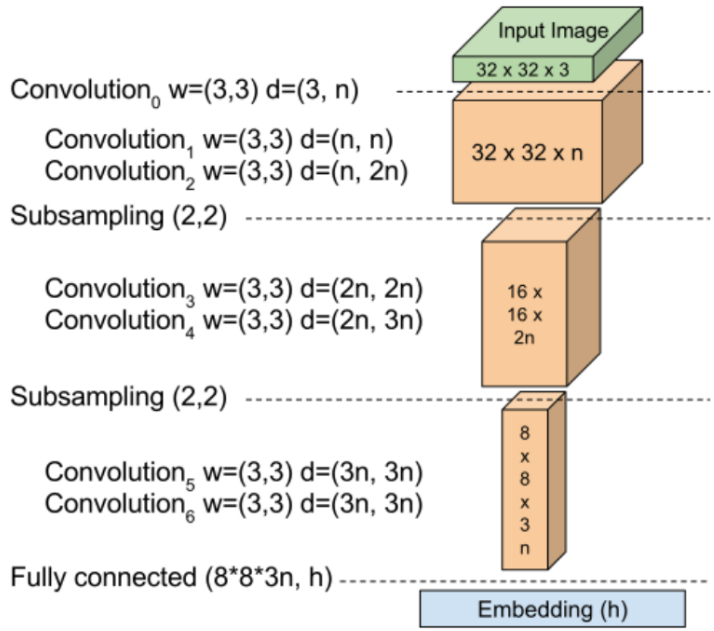
DC-GAN Examples



BE-GAN Architecture



(a) Generator/Decoder



(b) Encoder

BE-GAN Examples

BE-GAN Interpolations