# GENERATIVE ADVERSARIAL NETWORKS

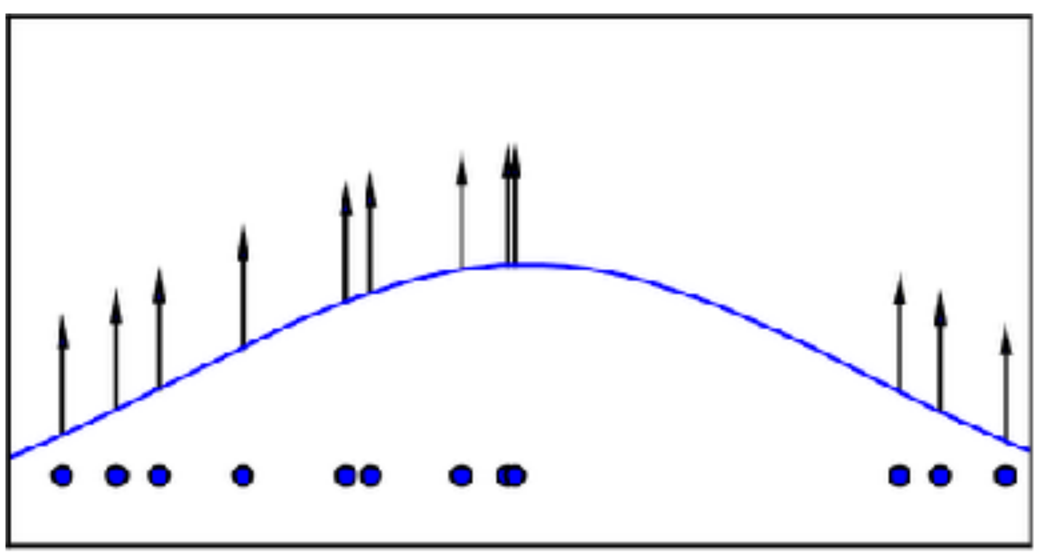
*Rajeswari Sivakumar*
*Deep Learning @UGA*
*2/2/17*

# MOTIVATIONS

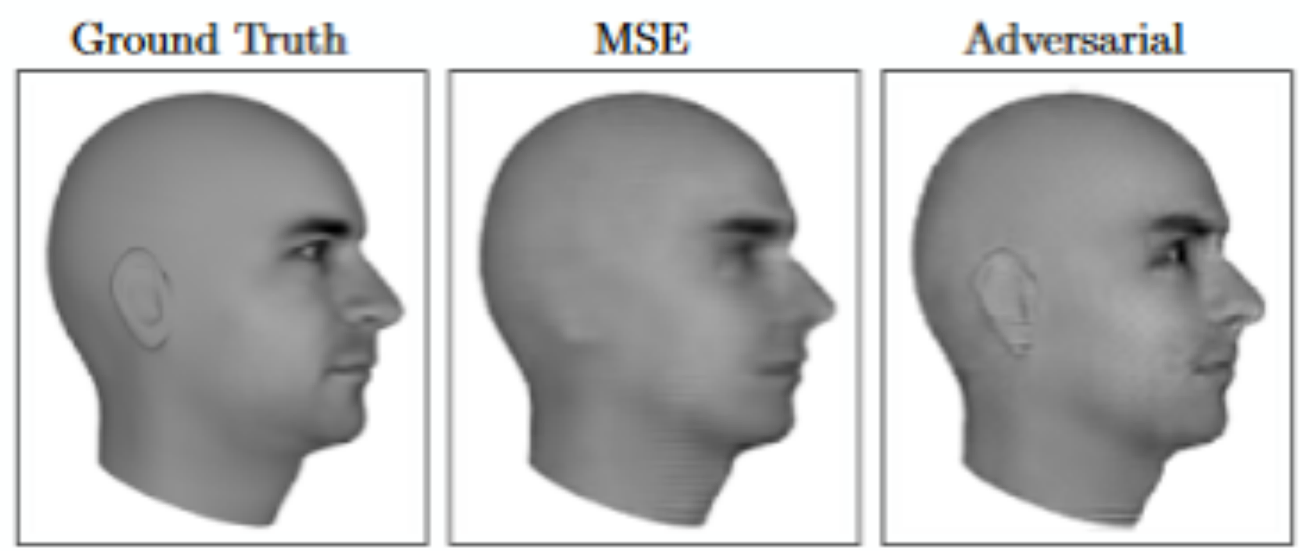
- Many applications.
- Reinforcement Learning
  - Learn different generative models to simulate environment and future actions of an agent
- Improve training in semi-supervised learning
- Multi-modal outputs
- Realistic generation
  - Super-resolution of images
  - Video frame prediction

# GENERATIVE MODELS

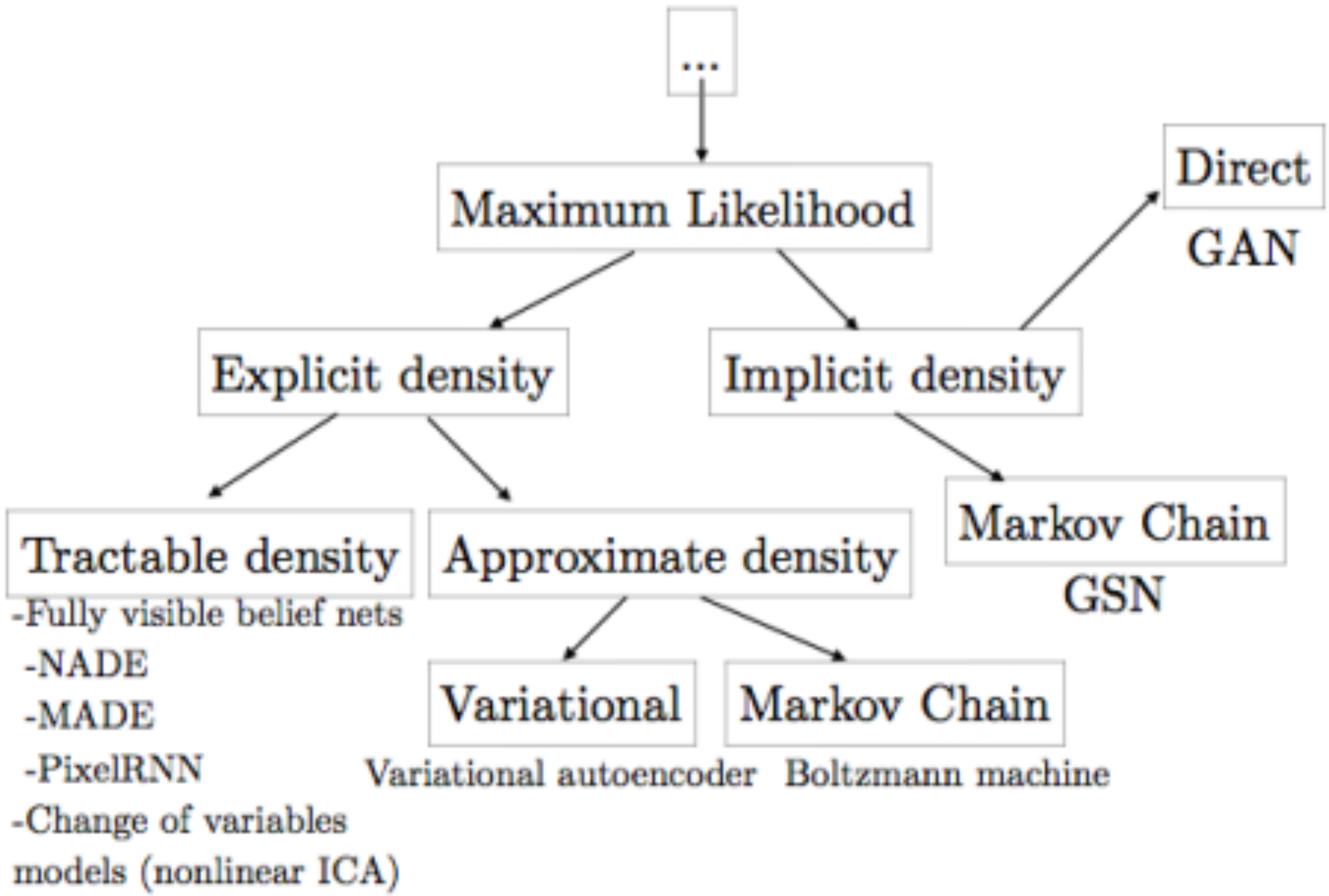
- Traditionally trying to learn distributions
  - i.e. clustering algorithms
- Alternatively can be used to generate samples that resemble real data.



$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x\sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

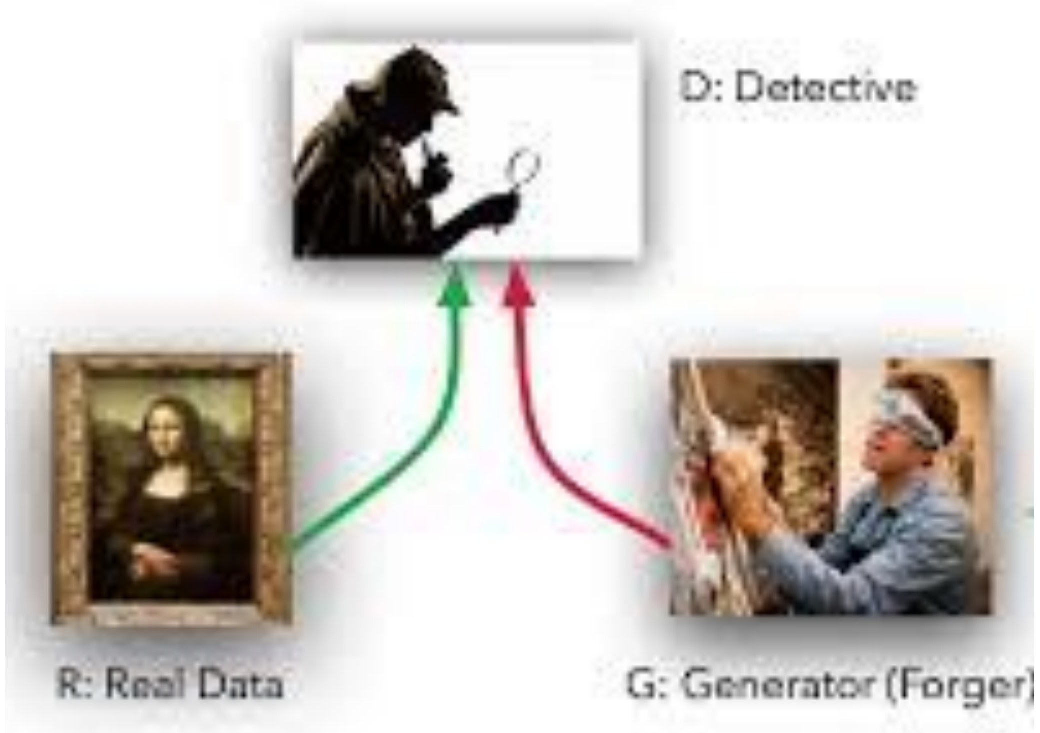


# FINDING MAXIMUM LIKELIHOOD



# WHAT ARE GANS?

- Two part generative network
- Generative
  - Tries to correctly represent distribution of features from data you are training on
- Discriminative
  - Tries to correctly differentiate an example from the generator from an example from the data



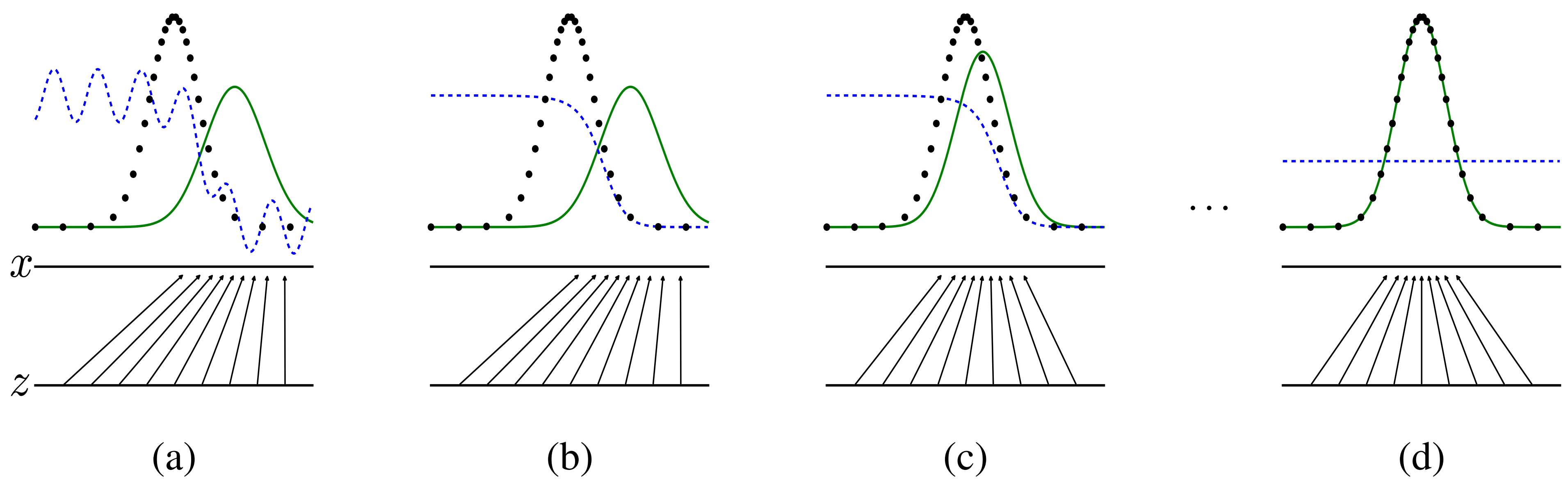


## THE COUNTERFEITER AND THE DETECTIVE

- Think of generator as a counterfeiter and discriminator
- Generator makes fakes to pass off as real
- Discriminator has to tell the two apart
- Both are always trying to one-up each other.

# OPTIMIZING A VALUE FUNCTION

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



# THEORETICAL

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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

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

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

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

**end for**
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

# OPTIMAL SOLUTIONS

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$$\begin{aligned}C(G) =& \max_D V(G, D) \\ =& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ =& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\ =& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]\end{aligned}$$

$$p_g = p_{\text{data}}, D_G^*(\boldsymbol{x}) = \tfrac{1}{2},$$

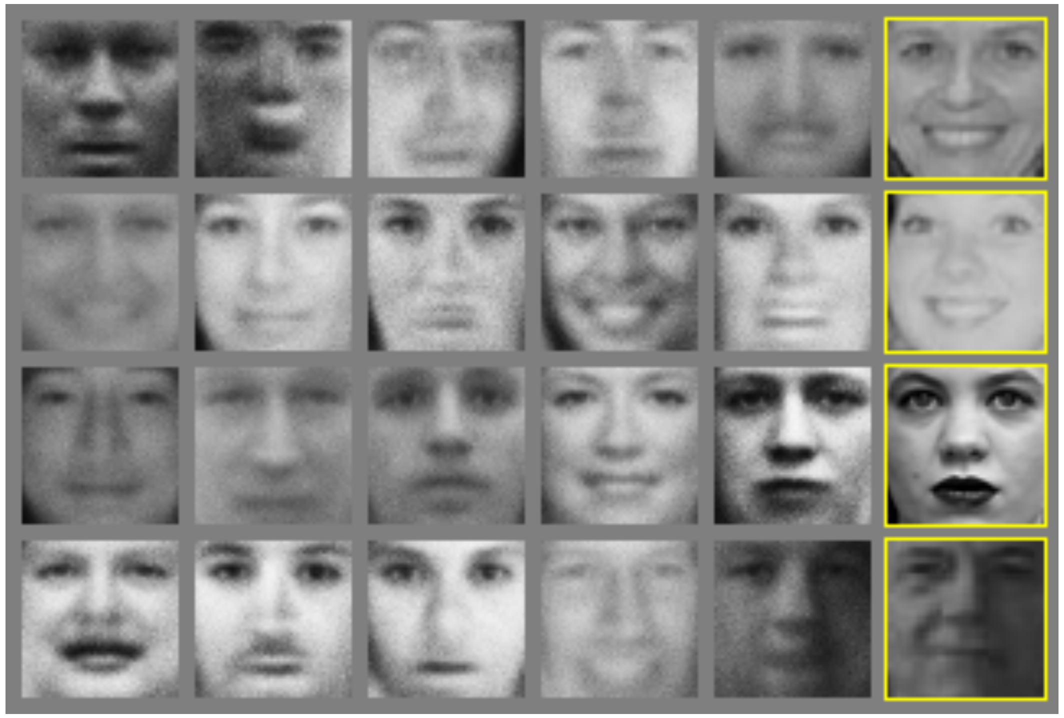
$$C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right)$$

$$C^* = -\log(4)$$

- We can define a optimal discriminator for a given generator
  - Can think of the the solution to the as maximizing the value function
- For optimal generator, the probability distribution matches the data
  - Thus a discriminator should be equally likely to say that a given *x* is data or generated

# EXPERIMENTS

- Trained on MNIST, Toronto Face Database, and CIFAR-10

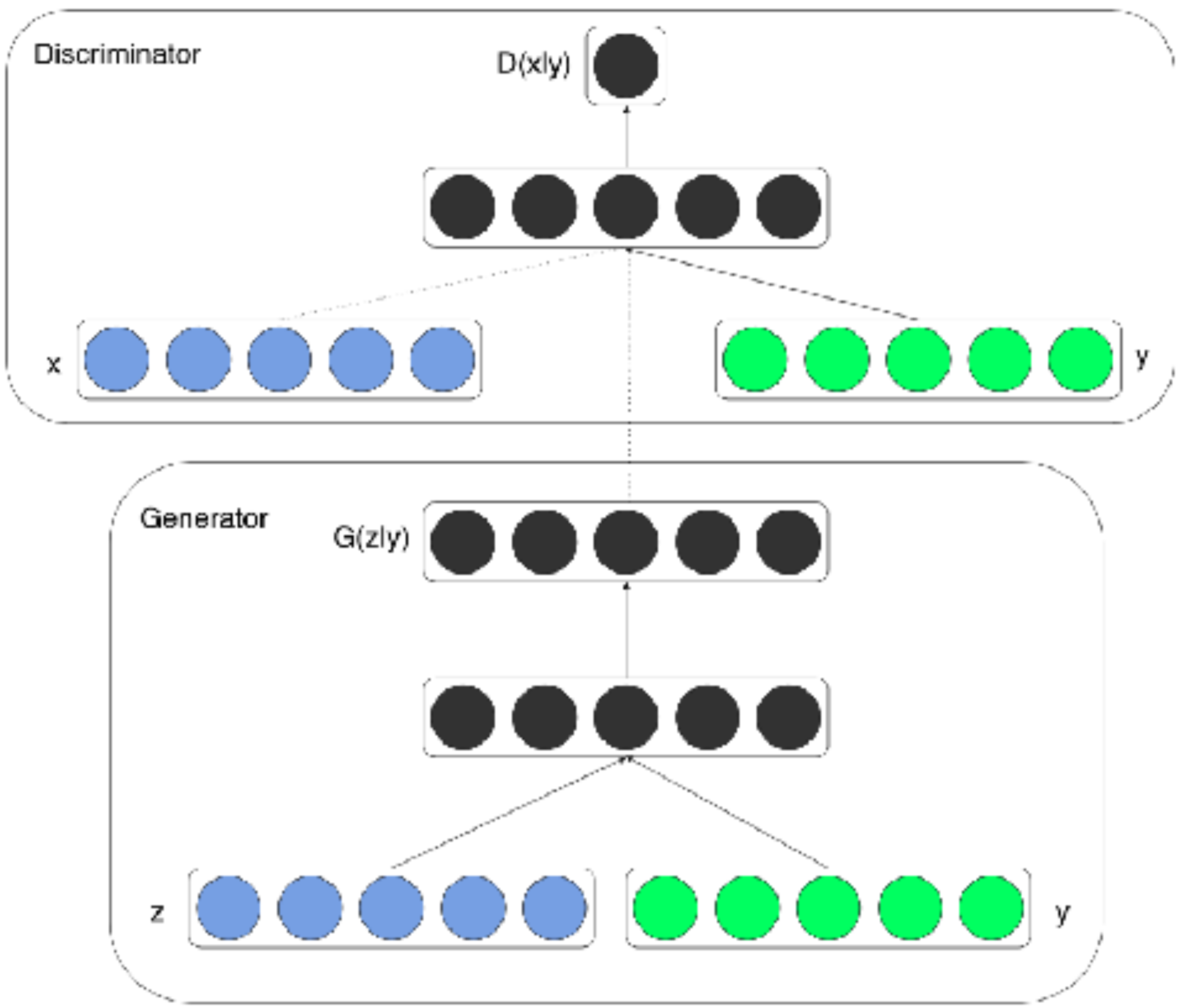


| Model | MNIST | TFD |
|---|---|---|
| DBN [3] | $138 \pm 2$ | $1909 \pm 66$ |
| Stacked CAE [3] | $121 \pm 1.6$ | $\mathbf{2110 \pm 50}$ |
| Deep GSN [6] | $214 \pm 1.1$ | $1890 \pm 29$ |
| Adversarial nets | $\mathbf{225 \pm 2}$ | $\mathbf{2057 \pm 26}$ |

# APPLICATIONS AND IMPROVEMENTS

- Conditional generative model
  - Conditional Generative Adversarial Networks (CGAN)
- Semi-supervised learning
  - Improving training efficiency

# C-GAN




- Mimic GAN, but with added complexity of a given prior event.
- Image tagging
- Avoiding mode collapse

# C-GAN

- Mimic GAN, but with added complexity of a given prior event.
- Image tagging
- Avoiding mode collapse

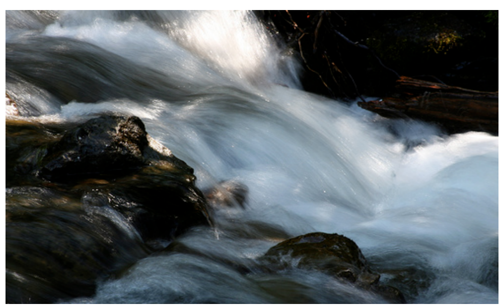
| | User tags + annotations | Generated tags |
|---|---|---|
|  | montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car | taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails |
|  | food, raspberry, delicious, homemade | chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes |
|  | water, river | creek, lake, along, near, river, rocky, treeline, valley, woods, waters |
|  | people, portrait, female, baby, indoor | love, people, posing, girl, young, strangers, pretty, women, happy, life |

Table 2: Samples of generated tags

# C-GAN

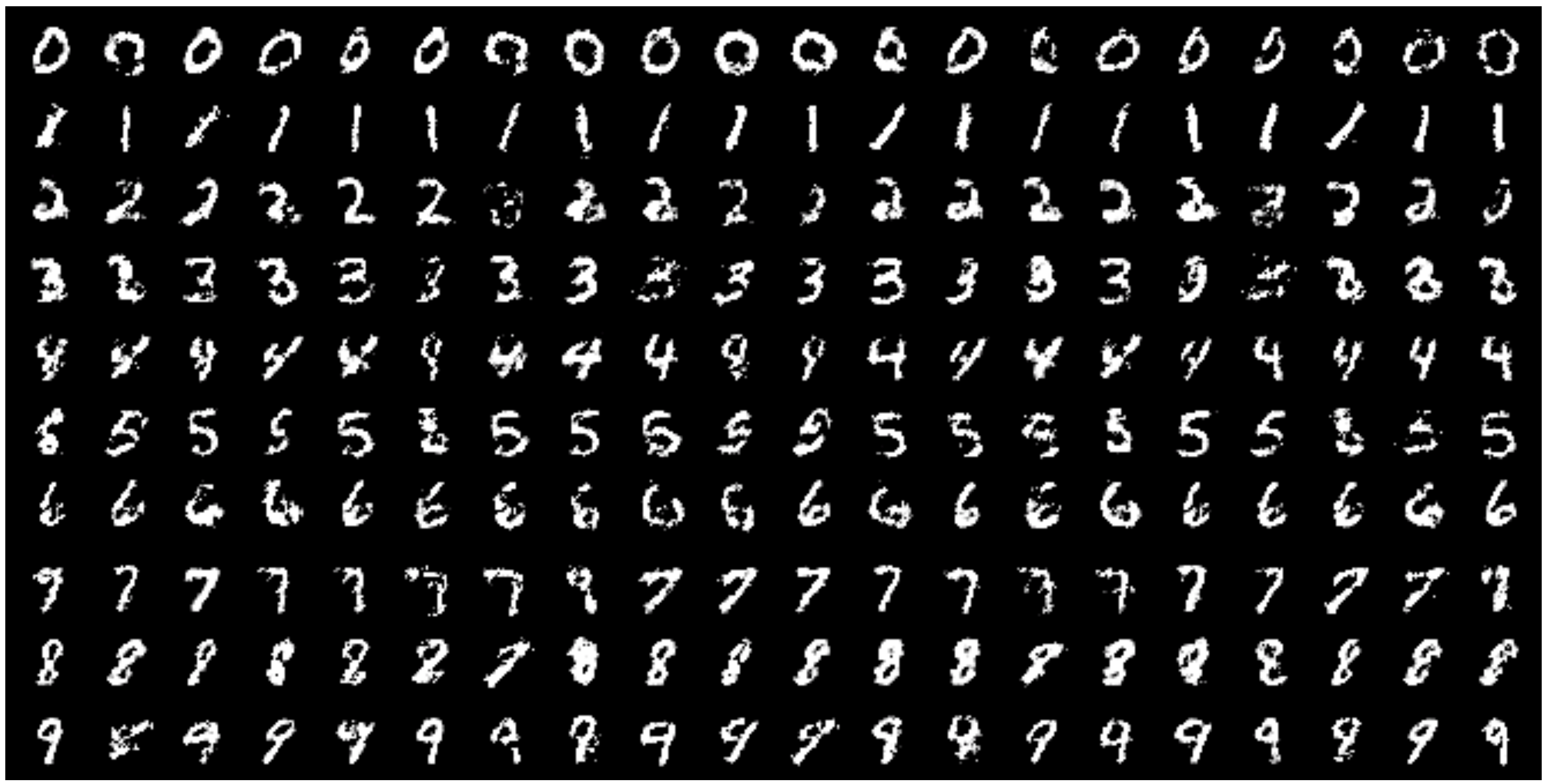


Figure 2: Generated MNIST digits, each row conditioned on one label

- Mimic GAN, but with added complexity of a given prior event.
- Image tagging
- Avoiding mode collapse

*Figure 1.* Output samples from SGAN and GAN after 2 MNIST epochs. SGAN is on the left and GAN is on the right.

*Table 1.* Classifier Accuracy

| EXAMPLES | CNN | SGAN |
|---|---|---|
| 1000 | 0.965 | 0.964 |
| 100 | 0.895 | 0.928 |
| 50 | 0.859 | 0.883 |
| 25 | 0.750 | 0.802 |

# SEMI-SUPERVISED GAN

- Train generative network and classifier at the same time
- By doing so:
  - Cut down on training time and improve accuracy
- Useful when there isn't a lot of data for training.

# QUESTIONS?