Training Generative Adversarial Networks with Binary Neurons by End-to-end Backpropagation

Hao-Wen Dong, Yi-Hsuan Yang

2018.10.17 at Music and AI Lab

OUTLINES

- Backgrounds
  - Generative Adversarial Networks
  - Binary Neurons
  - Straight-through Estimators
- BinaryGAN
- Experiments & Results
- Discussions & Conclusion

Backgrounds

Generative Adversarial Networks

- **Goal**—learn a **mapping** from the prior distribution to the data distribution [2]

prior distribution $\longrightarrow$ data distribution

Generative Adversarial Networks

- **Goal**—learn a **mapping** from the prior distribution to the data distribution [2]

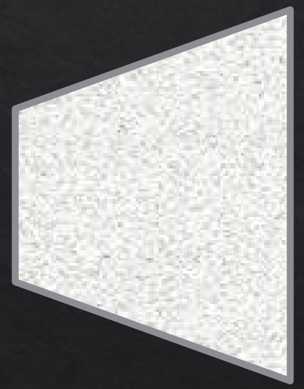
can be intractable

prior distribution → data distribution

Generative Adversarial Networks

- Use a **deep neural network** to learn an **implicit** mapping

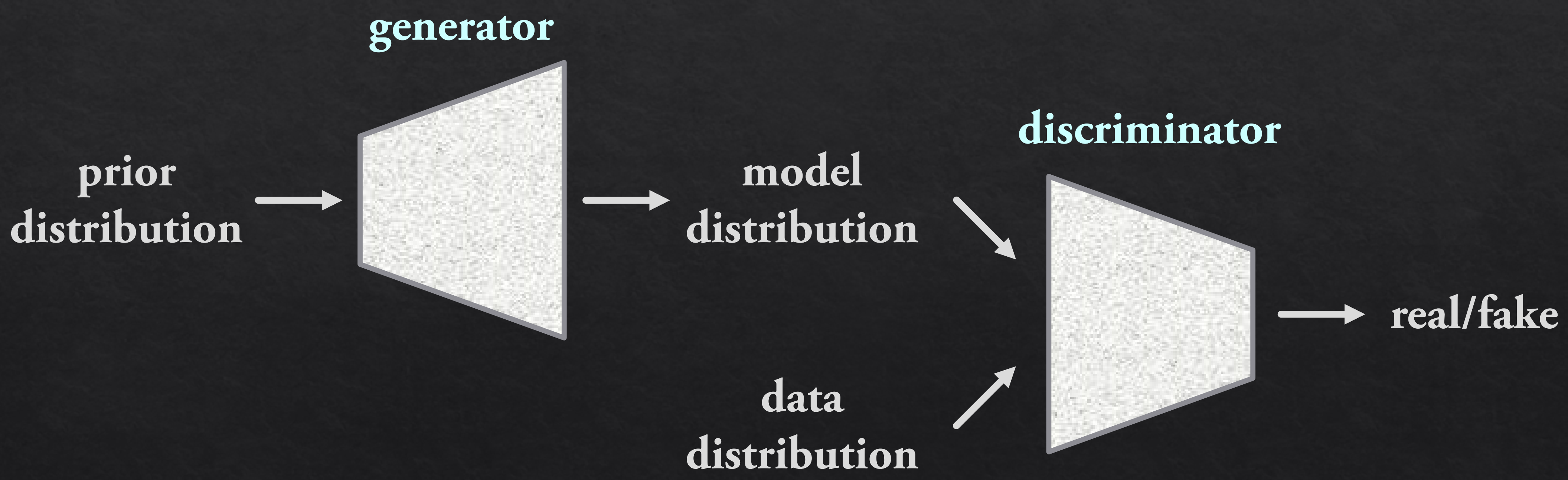
generator

prior distribution →  → model distribution

data distribution

Generative Adversarial Networks

- Use another **deep neural network** to provide **guidance/critics**



Binary Neurons

- **Definition:** neurons that output **binary-valued** predictions

BINARY NEURONS

- **Definition:** neurons that output **binary-valued** predictions
- **Deterministic binary neurons (DBNs):**

$$DBN(x) \equiv \begin{cases} 1, & if\ \sigma(x) < 0.5 \\ 0, & otherwise \end{cases}$$ (hard thresholding)

Binary Neurons

- **Definition:** neurons that output **binary-valued** predictions
- Deterministic binary neurons (**DBNs**):

$$DBN(x) \equiv \begin{cases} 1, & if\ \sigma(x) < 0.5 \\ 0, & otherwise \end{cases} \quad \text{(hard thresholding)}$$

- **Stochastic binary neurons** (**SBNs**):

$$SBN(x) \equiv \begin{cases} 1, & if\ z < \sigma(x) \\ 0, & otherwise \end{cases}, \quad z \sim U[0,1] \quad \text{(Bernoulli sampling)}$$

Backpropagating through Binary Neurons

- Backpropagating through binary neurons is intractable
  - For DBNs, it involves the nondifferentiable threshold function
  - For SBNs, it requires the computation of expected gradients on all possible combinations (*exponential to the number of binary neurons*) of values taken by the binary neurons

Backpropagating through Binary Neurons

- Backpropagating through binary neurons is intractable
  - For DBNs, it involves the nondifferentiable threshold function
  - For SBNs, it requires the computation of expected gradients on all possible combinations (*exponential to the number of binary neurons*) of values taken by the binary neurons
- We can introduce gradient estimators for the binary neurons
  - Examples include Straight-through [3,4] (*the one adopted in this work*), REINFORCE [5], REBAR [6], RELAX [7] estimators

Straight-through Estimators

- **Straight-through estimators:** [3,4]
  - Forward pass—use hard thresholding (DBNs) or Bernoulli sampling (SBNs)
  - Backward pass—pretend it as an identity function

Straight-through Estimators

- **Straight-through estimators:** [3,4]
  - Forward pass—use hard thresholding (DBNs) or Bernoulli sampling (SBNs)
  - Backward pass—pretend it as an identity function
- **Sigmoid-adjusted straight-through estimators**
  - Use the derivative of the sigmoid function in the backward pass
  - Found to achieve better performance in a classification task presented in [4]
  - Adopted in this work

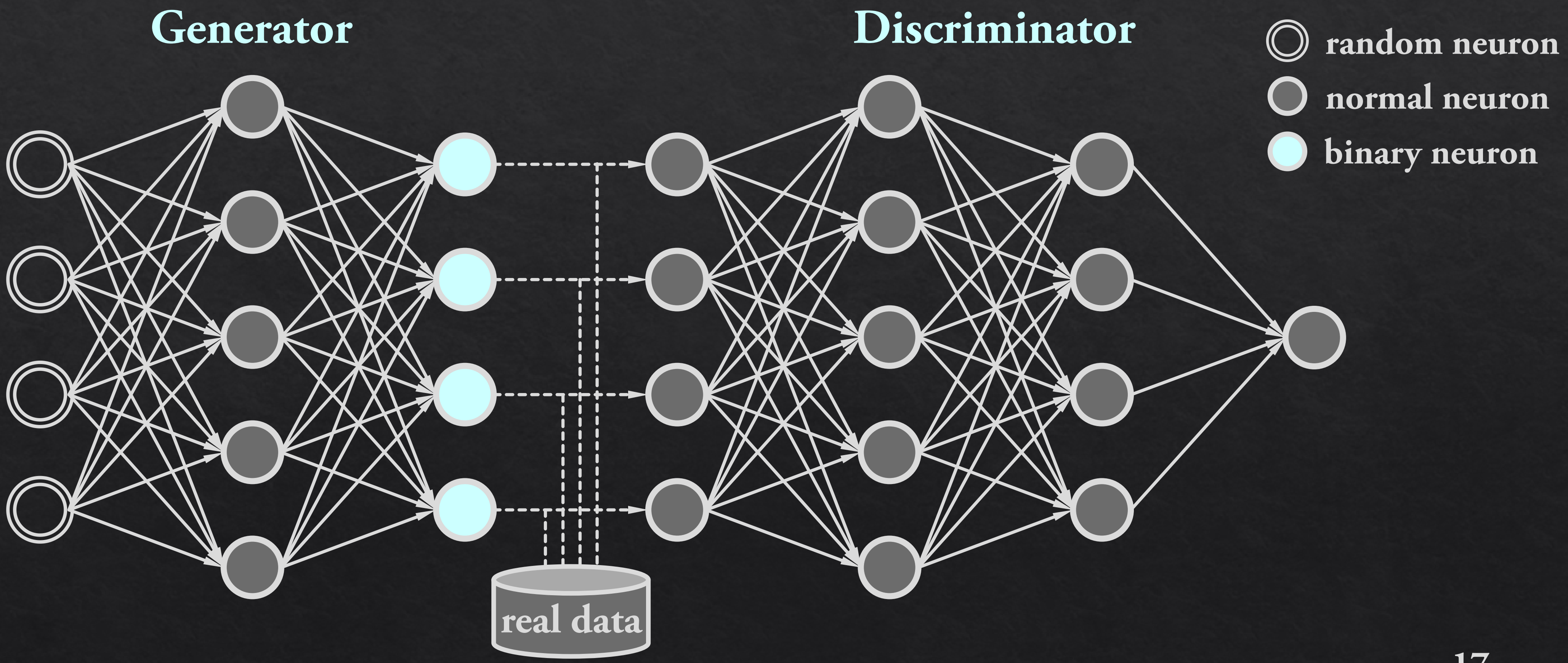
BinaryGAN

BinaryGAN

- Use **binary neurons** at the **output layer** of **the generator**
- Use **sigmoid-adjusted straight-through estimators** to provide the gradients for the binary neurons
- Train the whole network by **end-to-end backpropagation**

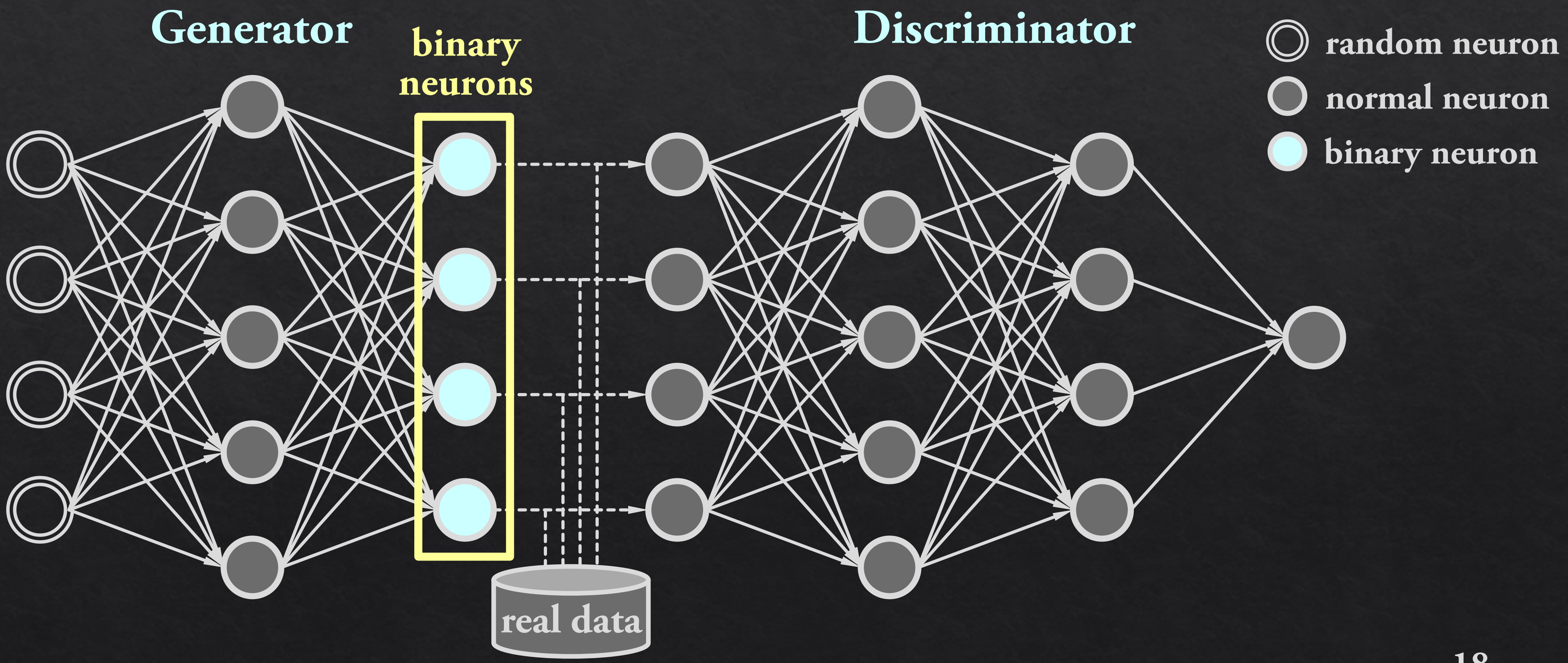
BinaryGAN

(implemented by multilayer perceptrons)



BinaryGAN

(implemented by multilayer perceptrons)

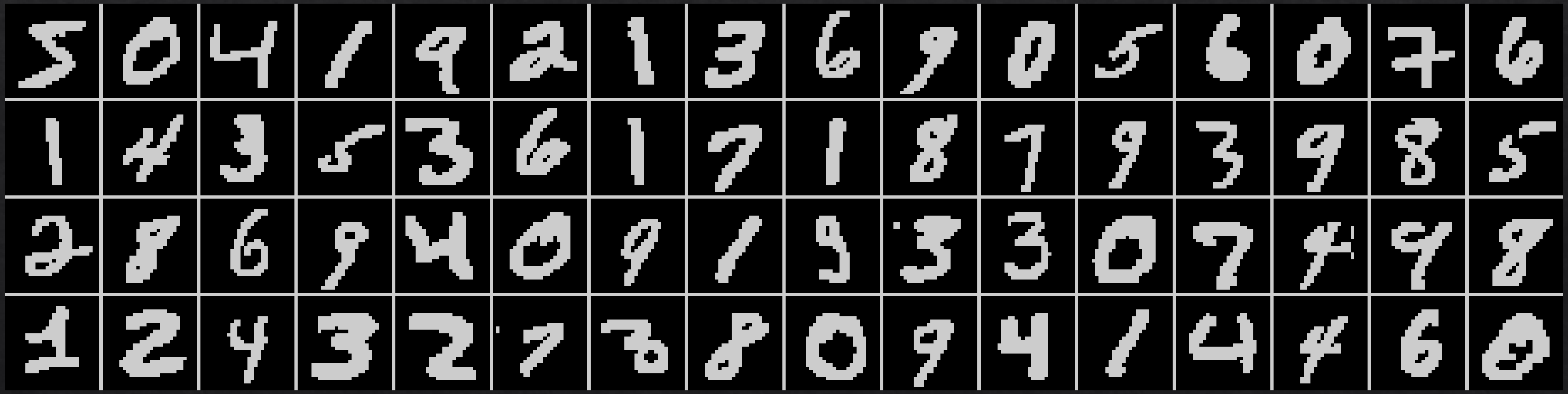


Experiments & Results

Training Data

- **Binarized MNIST** handwritten digit database [8]
  - Digits with **nonzero intensities** → 1
  - Digits with **zero intensities** → 0

Sample binarized MNIST digits



Implementation Details

- Batch size is **64**
- Use **WGAN-GP** [9] objectives
- Use **Adam** optimizer [10]
- Apply **batch normalization** [11] **to the generator** (but not to the discriminator)
- Binary neurons are implemented with the code kindly provided in a blog post on the R2RT blog [12]

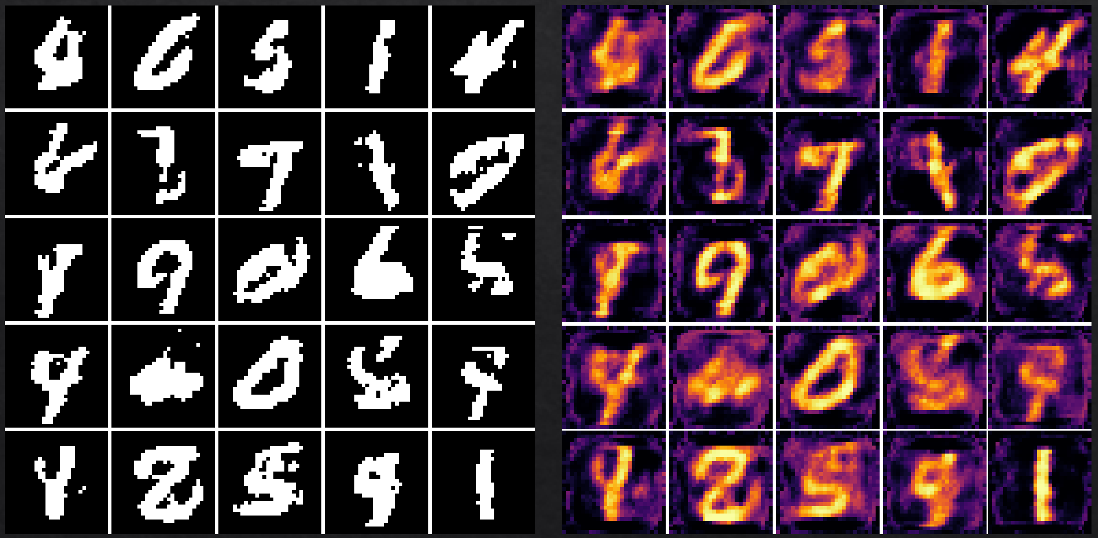
Implementation Details

- Apply slope annealing trick [13]
  - Gradually increase the slopes of the sigmoid functions used in the sigmoid-adjusted straight-through estimators as the training proceeds
  - We multiply the slopes by 1.1 after each epoch
  - The slopes start from 1.0 and reach 6.1 at the end of 20 epochs

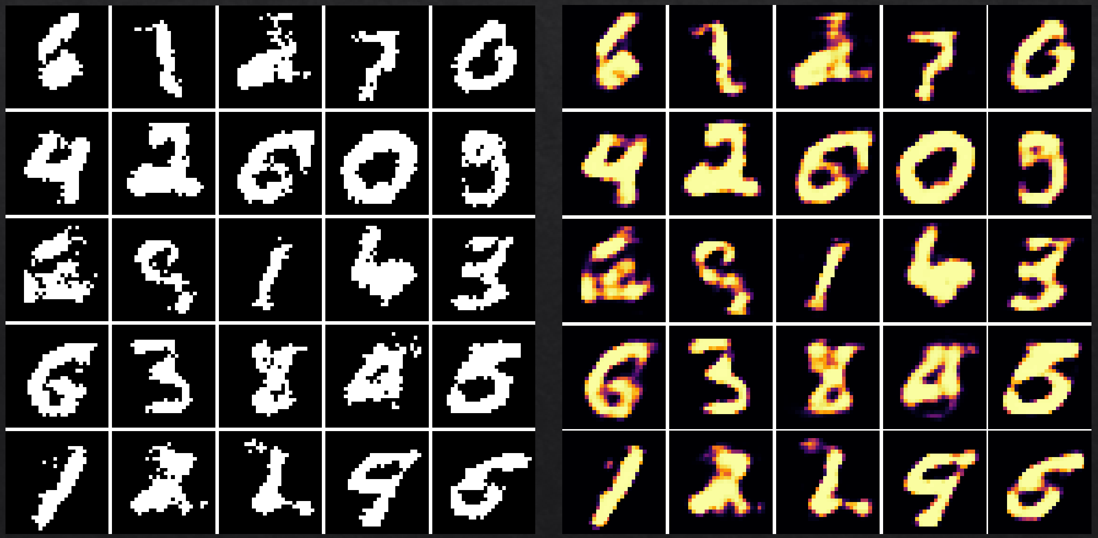
EXPERIMENT I—DBNS VS SBNS

- DBNs and SBNs can achieve **similar qualities**
- They show **distinct characteristics** on the **preactivated outputs**

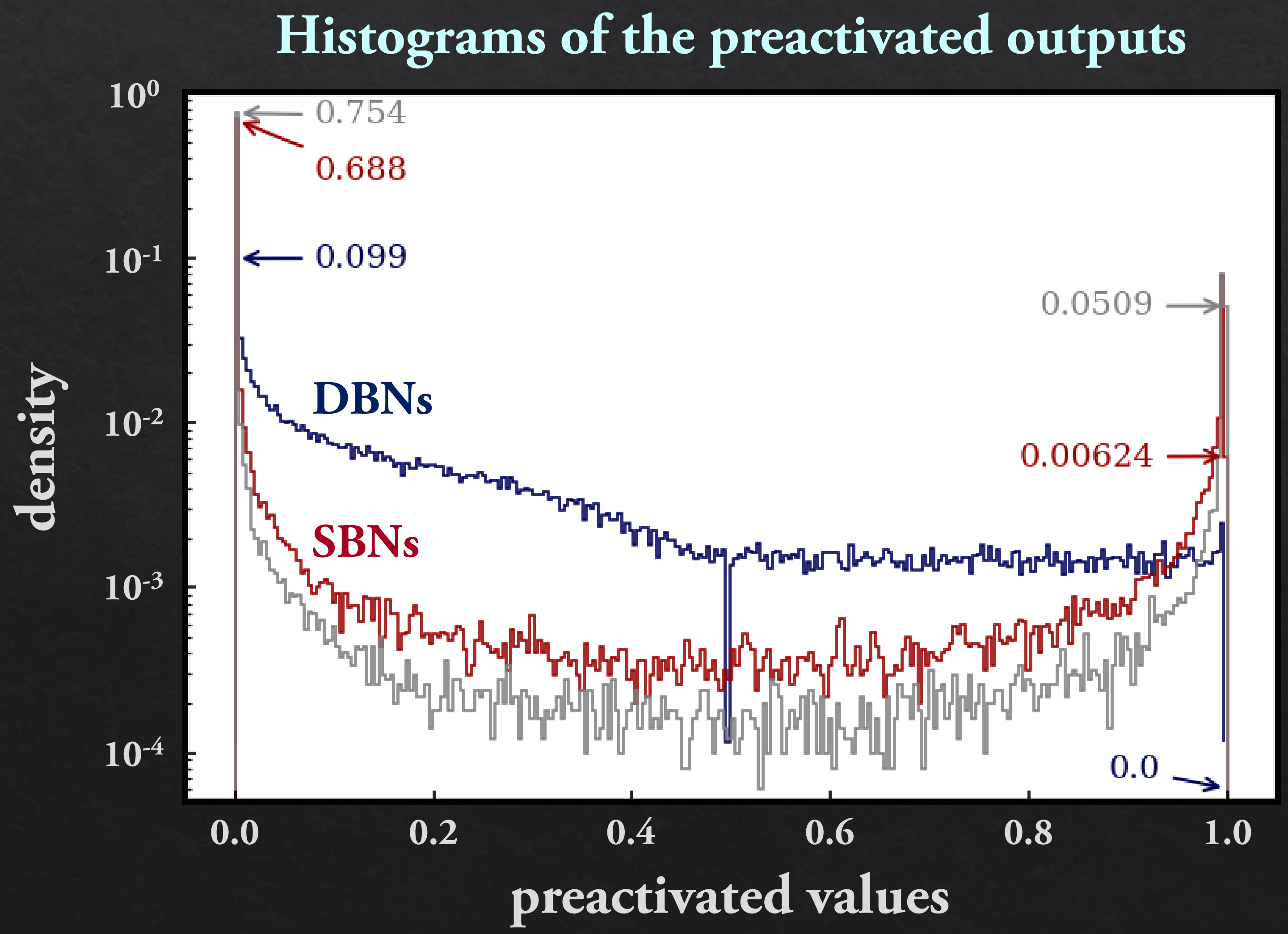
DBNs



SBNs



Experiment I—DBNs vs SBNs



DBNs

– more values in the middle

– a notch at 0.5 (the threshold)

SBNs

– more values close to 0 and 1

EXPERIMENT II—REAL-VALUED MODEL

- Use **no binary neurons**
- Train the discriminator by the **real-valued outputs** of the generator

raw predictions

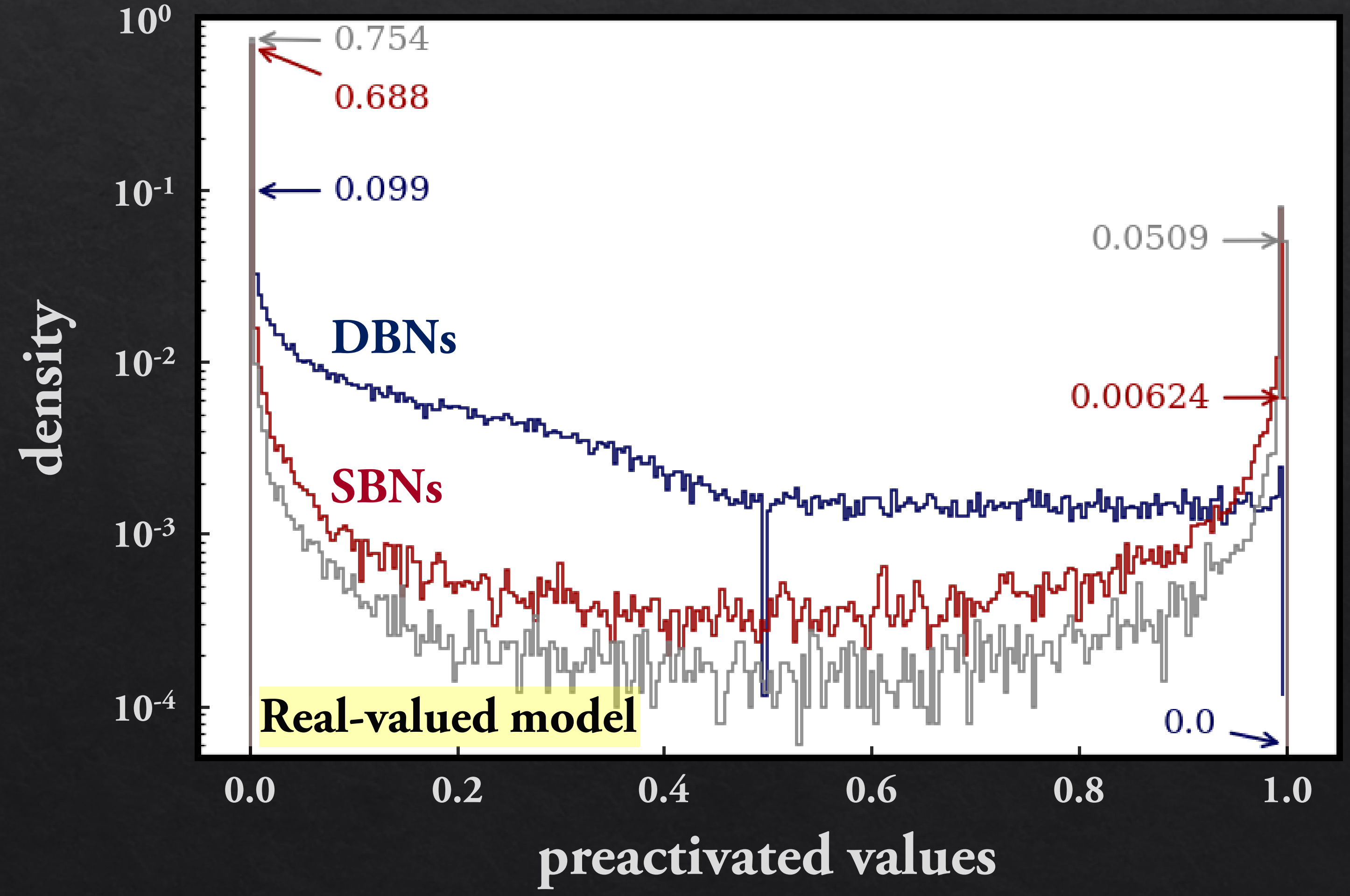
binarized results

hard thresholding

Bernoulli sampling

EXPERIMENT II—REAL-VALUED MODEL

Histograms of the preactivated outputs



DBNs

– more values in the middle

– a notch at 0.5 (the threshold)

SBNs

– more values close to 0 and 1

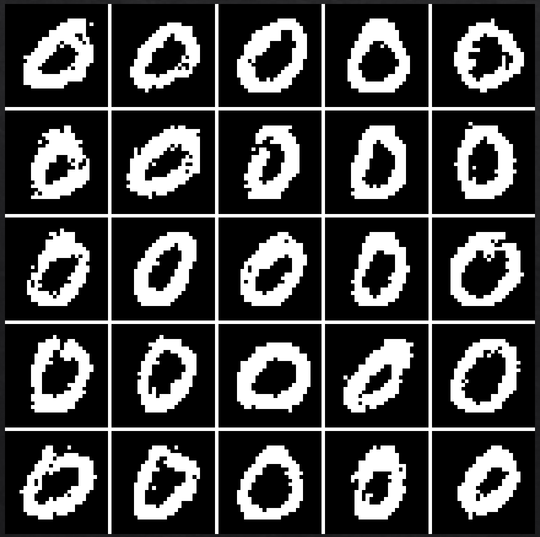
Real-valued model

– even more U-shaped

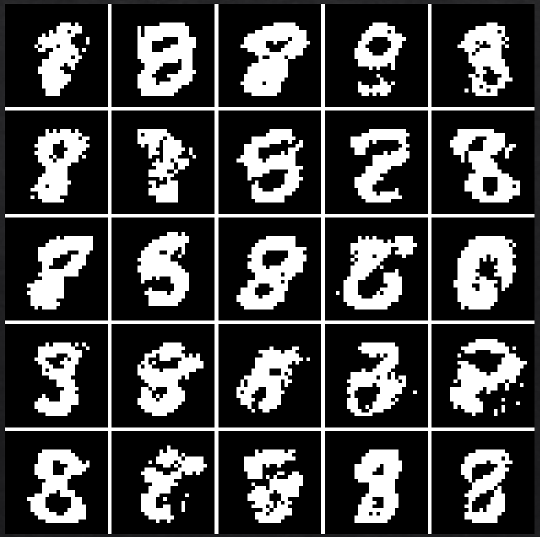
EXPERIMENT III—GAN OBJECTIVES

- **WGAN** [14] model can **achieve similar qualities** to the WGAN-GP
- **GAN** [2] model suffers from **mode collapse** issue

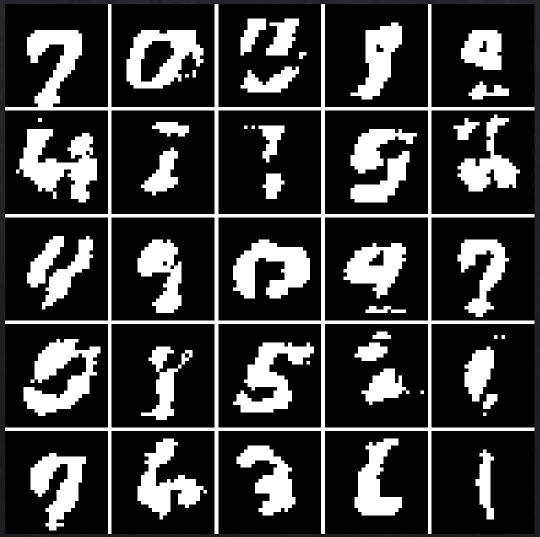
GAN + DBNs



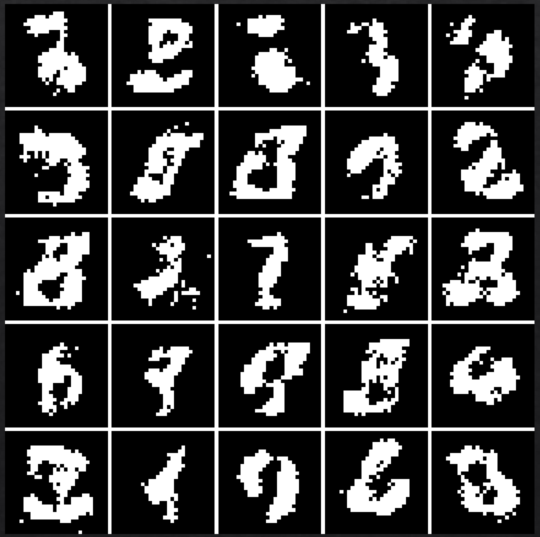
GAN + SBNs



WGAN + DBNs



WGAN + SBNs

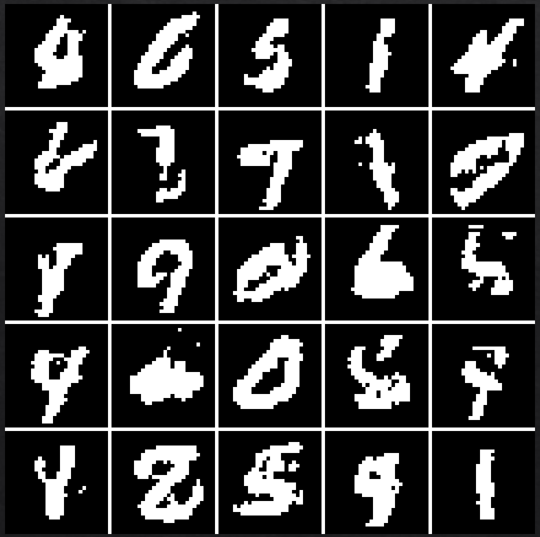


Experiment IV—MLPs vs CNNs

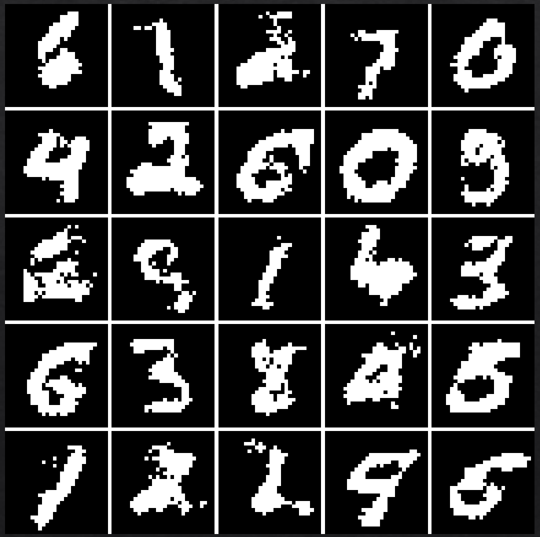
- **CNN model** produces **less artifacts** even with **a small number of trainable parameters** (MLP—0.53M; CNN—1.4M)

MLP model

DBNs

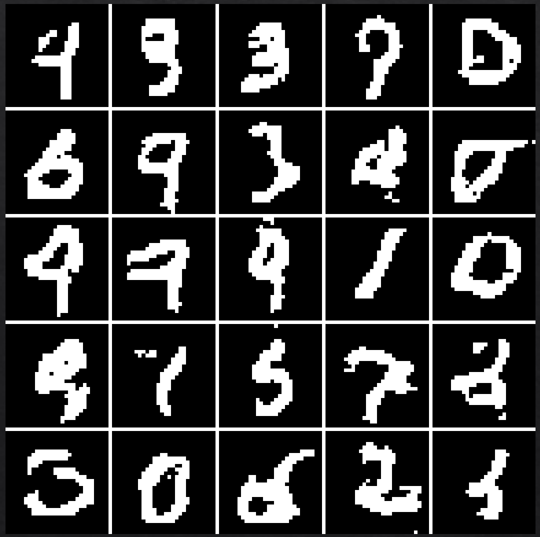


SBNs



CNN model

DBNs



SBNs

Discussions & Conclusion

Discussions

- **Why is binary neurons important?**
  - Open the possibility of **conditional computation graph** [4,13]
  - Move toward a stronger AI that can make **reliable decisions**

Discussions

- **Why is binary neurons important?**
  - Open the possibility of **conditional computation graph** [4,13]
  - Move toward a stronger AI that can make **reliable decisions**
- **Other approaches to model discrete distributions with GANs**
  - Replace the target discrete outputs with **continuous relaxations**
  - View the generator as agent in reinforcement learning (RL) and introduce **RL-based training strategies**

Future Work

- Examine the use of gradient estimators for training a GAN that has a conditional computation graph

Conclusion

- A new GAN model that
  - can generate binary-valued predictions without further post-processing
  - can be trained by end-to-end backpropagation
- Experimentally compare
  - deterministic and stochastic binary neurons
  - the proposed model and the real-valued model
  - GAN, WGAN, WGAN-GP objectives
  - MLPs and CNNs

References

[1] Hao-Wen Dong and Yi-Hsuan Yang. Training Generative Adversarial Networks with Binary Neurons by End-to-end Backpropagation. *arXiv Preprint arXiv:1810.04714*, 2018

[2] Ian J. Goodfellow et al. Generative adversarial nets. In *Proc. NIPS*, 2014.

[3] Geoffrey Hinton. Neural networks for machine learning—Using noise as a regularizer (lecture 9c), 2012. Coursera, video lectures. [Online] https://www.coursera.org/lecture/neural-networks/using-noise-as-a-regularizer-7-min-wbw7b.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[6] George Tucker, Andriy Mnih, Chris J Maddison, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Proc. NIPS*, 2017.

[7] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *Proc. ICLR*. 2018.

References

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *Proc. NIPS*, 2017.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[12] Binary stochastic neurons in tensorflow, 2016. Blog post on the R2RT blog. [Online] https://r2rt.com/binary-stochastic-neurons-in-tensorflow.

[13] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *Proc. ICLR*, 2017.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. ICML*, 2017.

Thank you for your attention