Information theory and Deep learning: An Emerging Interface

Presenting Team

Sreeram Kannan

Hyeji Kim

Sewoong Oh

University of Washington, Seattle

University of Illinois, Urbana Champaign

Special Thanks:

Pramod Viswanath (UIUC)

Success of Deep Learning

Speech

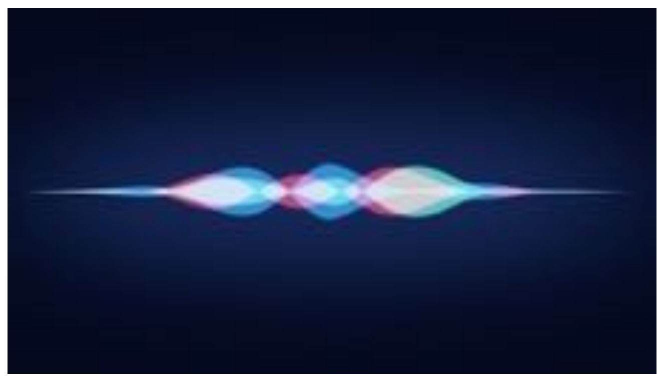
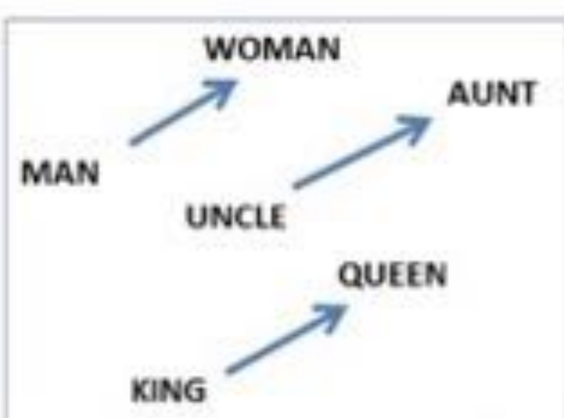
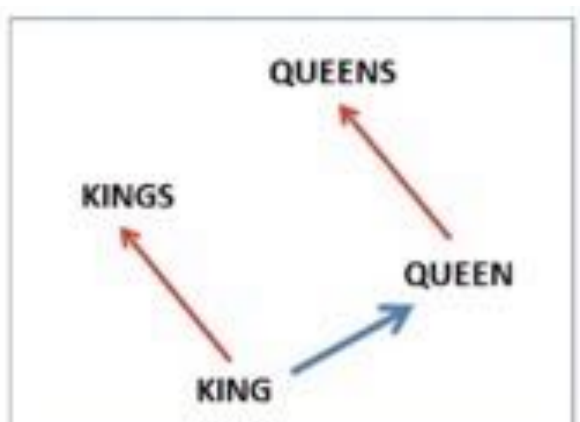


Image recognition

"construction worker in orange safety vest is working on road."

NLP





Video

https://www.youtube.com/watch?v=9Yq67CjDqvw

Why does Deep Learning work?

Model deficit

- Hard to model image, speech, language, video..

alphaGo => No model deficit

Algorithm deficit

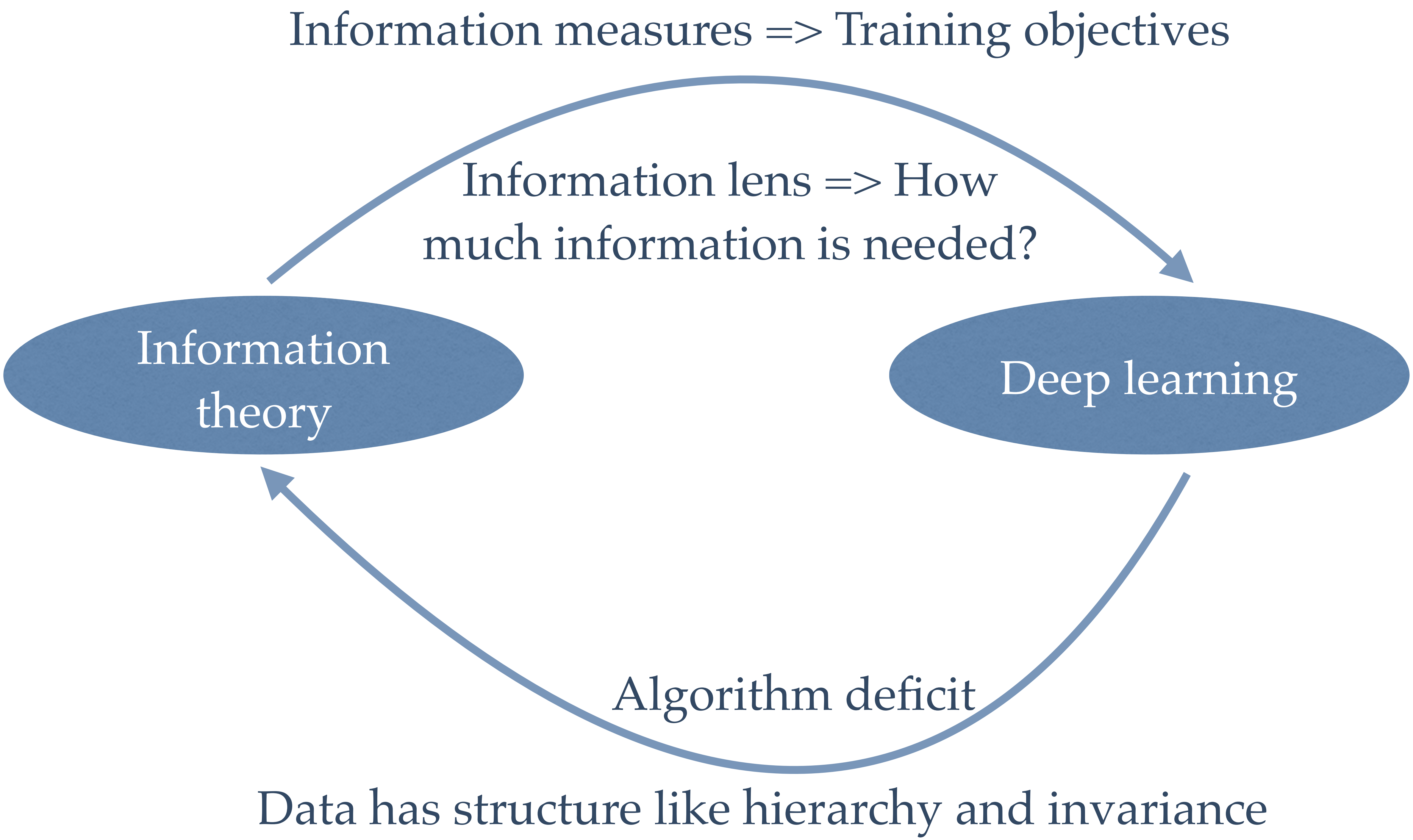
- Hard to find optimal algorithms for known model..

Example: Nanopore sequencing

Nearly a markov model

- Yet deep learning does “better”. Why?

Information theory and Deep learning



Organization: This Tutorial

Part-1: Deep learning for information theory

1a. Deep learning for communication

1b. Deep learning for statistical inference

Part-2: Information theory for deep learning

2a. Theory for GAN

2b. Learning Gated Neural Networks

Background on Neural Network Training

Sewoong Oh

University of Illinois at Urbana-Champaign

Classification

- Problem statement

Given labelled examples $\{(X_i, Y_i)\}_{i=1}^n$, find a classifier f that minimizes the loss $\mathcal{L}$ of our choice

$$\min_f \ \mathbb{E}_{X,Y} \left[\, \mathcal{L}(f(X), Y) \, \right]$$

- As we access the joint distribution $P_{X,Y}$ through samples, we minimize the sample mean instead,

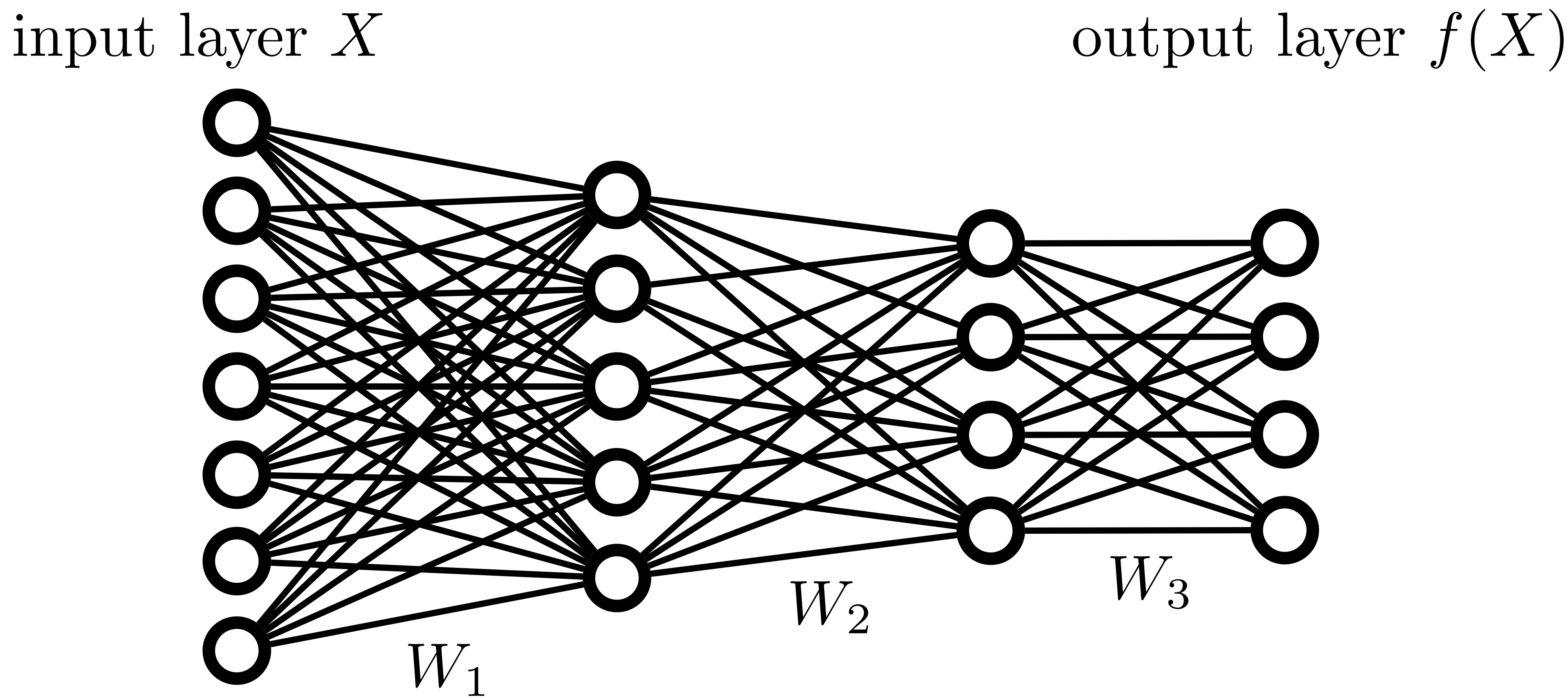
$$\min_f \ \frac{1}{n} \sum_{i=1}^n \ \mathcal{L}(f(X_i), Y_i)$$

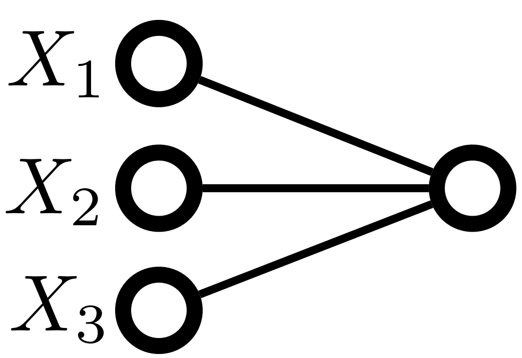
- To avoid overfitting to the training samples, we search over a restricted class of functions

$$\min_{f \in \mathcal{F}} \ \frac{1}{n} \sum_{i=1}^n \ \mathcal{L}(f(X_i), Y_i)$$

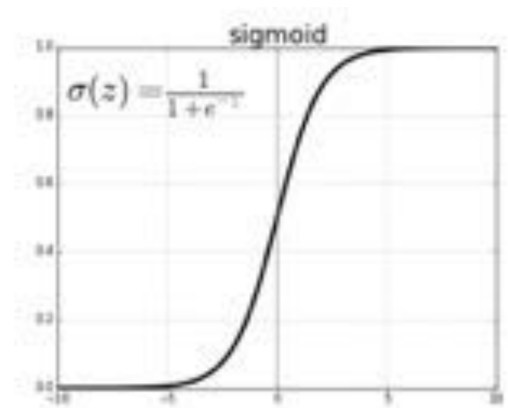
- Neural networks: a parametric family with a graceful tradeoff between representation and generalization

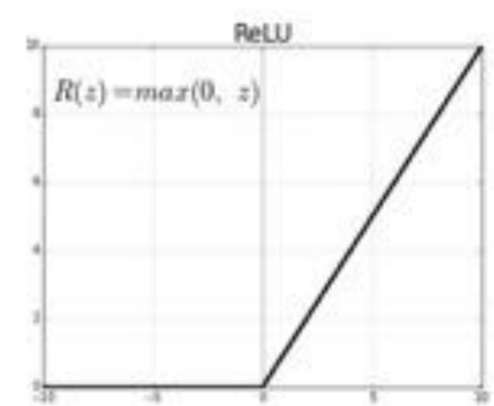
Neural Network of depth *d* and weights ($W_1,\ldots,W_d$)





$$\sigma(W_{11}X_1 + W_{13}X_3 + W_{13}X_3)$$





$$f(X) \;=\; \sigma\Big(\, W_d \cdots \sigma\,\big(\, W_2\,\sigma(W_1 X)\,\big)\;\cdots\Big)$$

Gradient computation is simple

- Choose the loss function (e.g. for binary classification)
  - L_2 loss

$$\min_{W_1,\ldots,W_d} \quad \frac{1}{n}\sum_{i=1}^{n}(Y_i - f(X_i))^2$$

  - Cross entropy loss

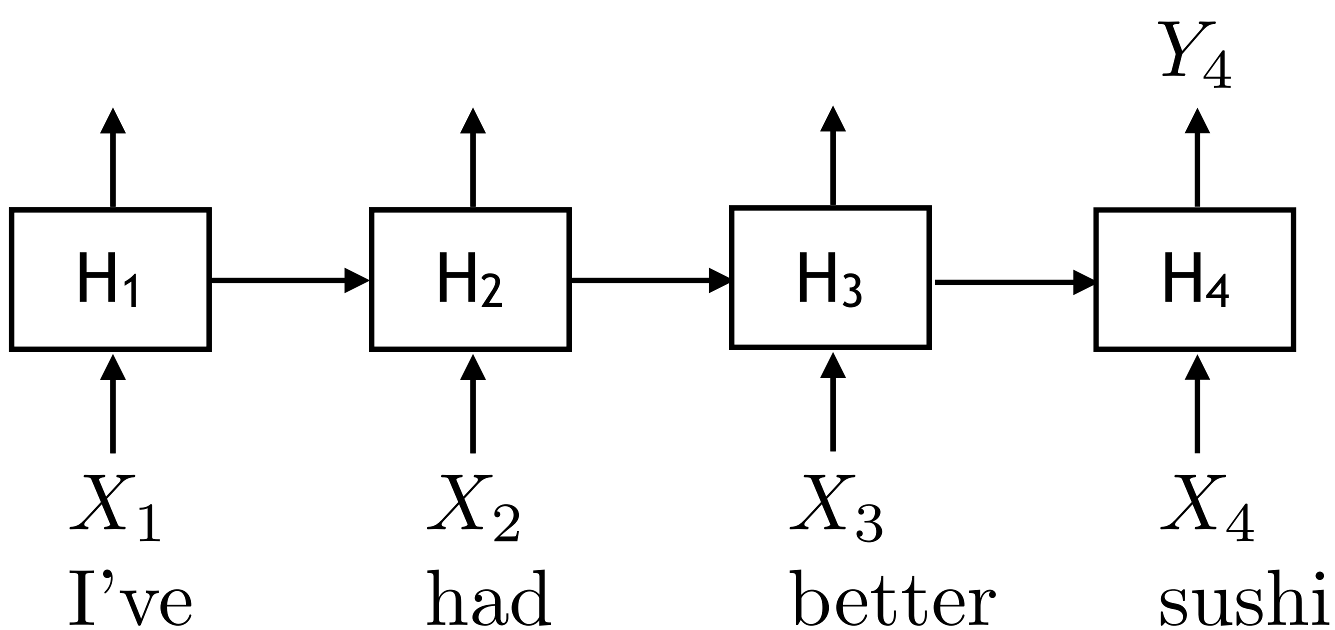
$$\min_{W_1,\ldots,W_d} \quad \frac{1}{n}\sum_{i=1}^{n} -\{Y_i \log(f(X_i)) + (1-Y_i)\log(1-f(X_i))\}$$

- (variants of) gradient descent are used
  - Efficient gradient computation via backpropagation

$$f(X) \quad = \quad \sigma\Big(\, W_d \cdots \sigma\,(\, W_2\,\sigma(W_1 X)\,)\ \cdots\Big)$$

Sequential data / time series (e.g. translation)

- Feed-forward NN fails for sequential data that has
  - causal structures and
  - variable lengths
- Recurrent neural networks (RNN) have been proposed
  - captures the causal structure via memory



$$H_t = \tanh\left(W\,X_t + U\,H_{t-1}\right)$$
$$Y_t = V\,H_t$$

Autoencoder for unsupervised learning

- (informal) Problem statement

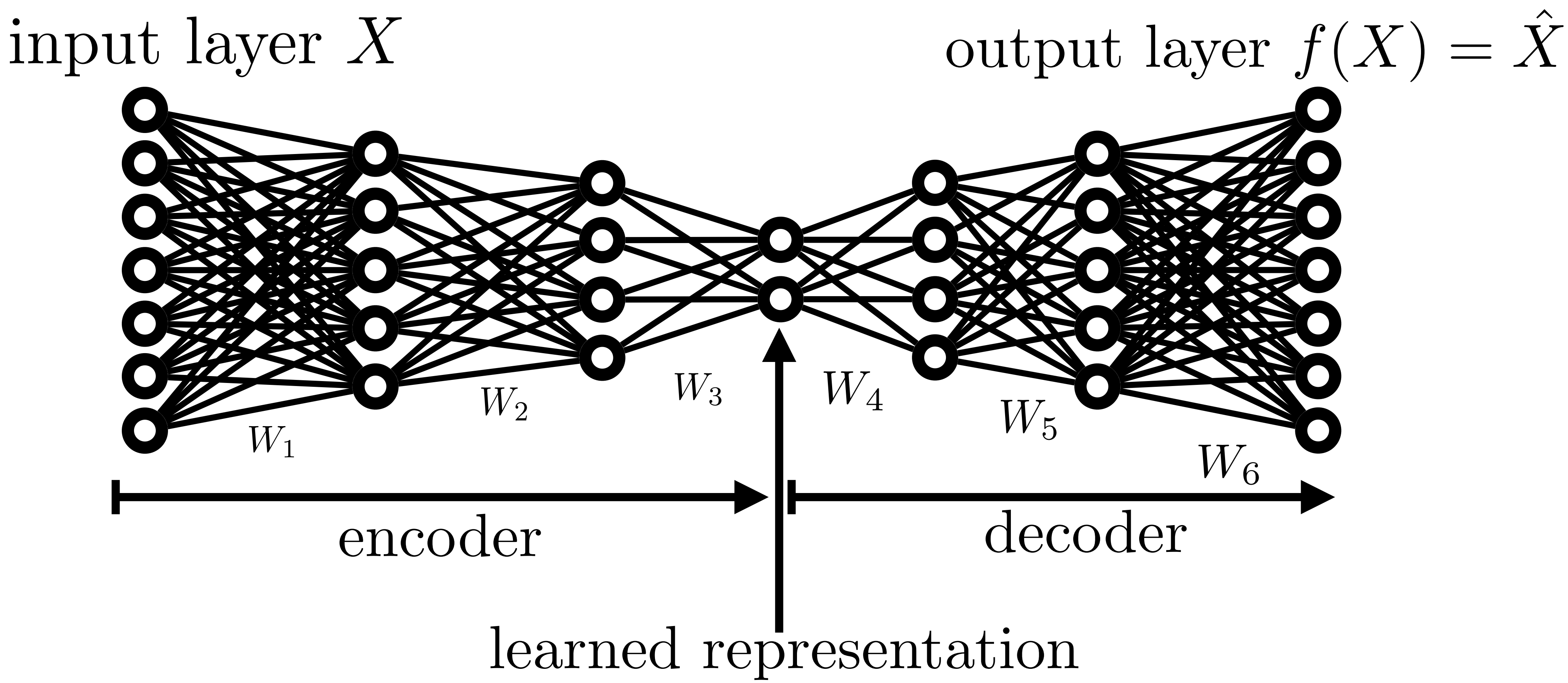
  Given unlabelled training data $\{X_i\}_{i=1}^n$,
  learn a useful representation $f(X_i)$

- What is useful?
  - Dimensionality reduction (as in visualization or efficient processing)
  - Compression (as in smaller file size)
  - Representation learning for downstream tasks (as in word2vec)

- Premise of autoencoder:
  - a good representation should recover X

Autoencoder

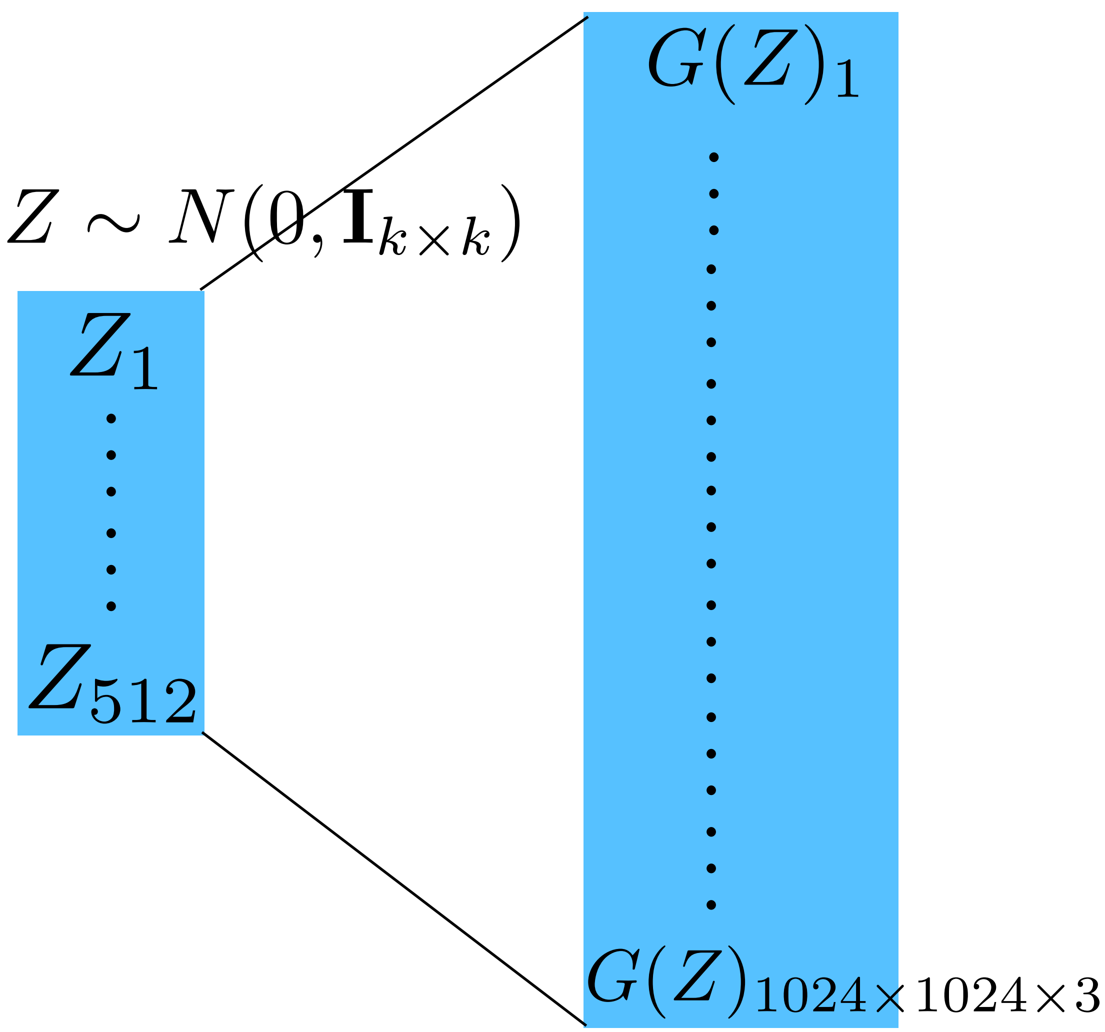
- An encoder and a decoder via neural networks



- minimize loss in recovering the original example

$$\min_{W_1,\ldots,W_d} \frac{1}{n} \sum_{i=1}^{n} \|X_i - f(X_i)\|^2$$

Neural network generative models



$$G(Z) \in \mathbb{R}^{1024\times1024\times3}$$

Part 1A. Application of deep learning to communications

Hyeji Kim

University of Illinois at Urbana-Champaign

Organization: This Tutorial

Part-1: Deep learning for information theory

1a. Deep learning for communication

1b. Deep learning for statistical inference

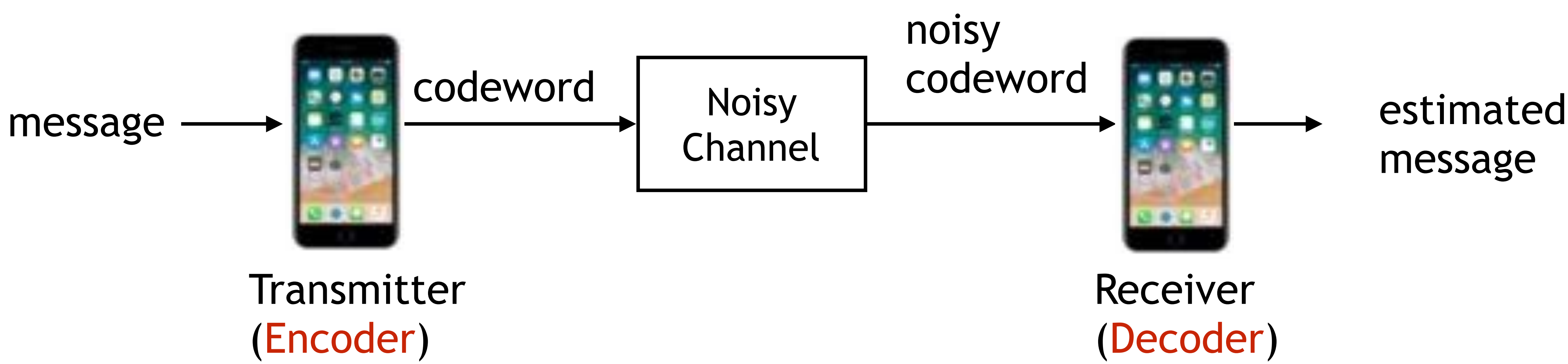
Part-2: Information theory for deep learning

2a. Theory for GAN

2b. Learning Gated Neural Networks

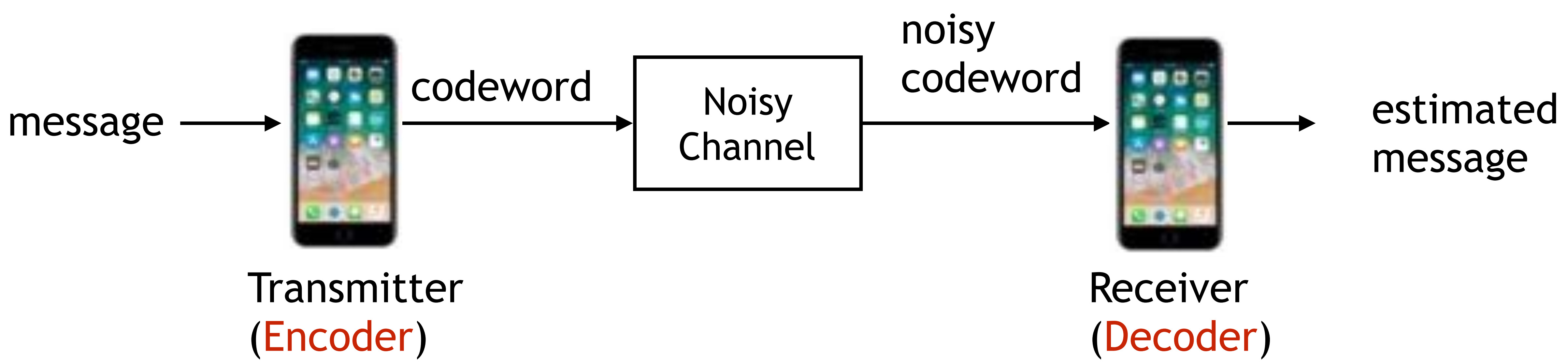
Communications

- Models are often well defined => No model deficit



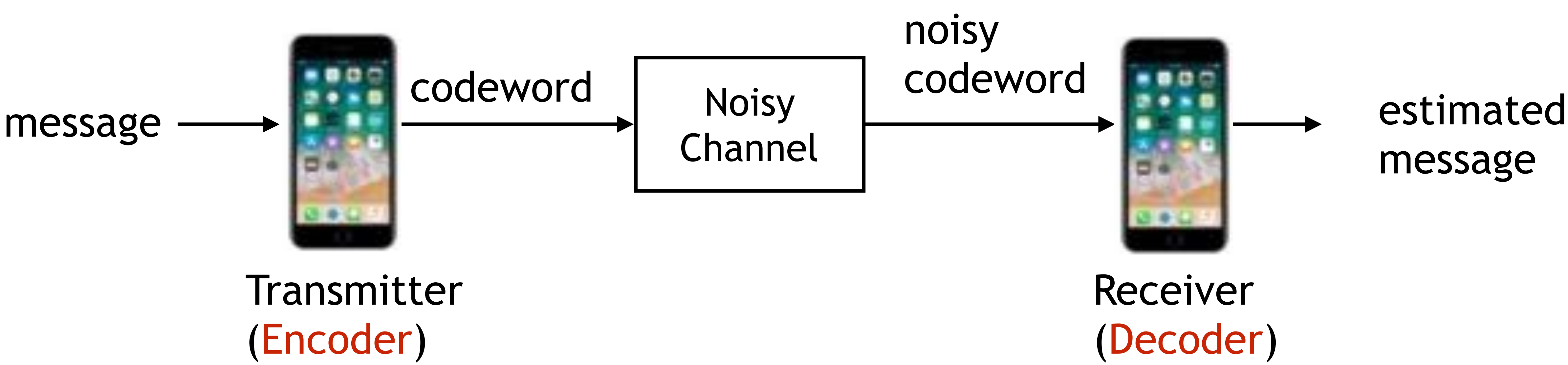
Communications

- Models are often well defined => No model deficit
- Designing a robust encoder/decoder is critical



Communications

- Models are often well defined => No model deficit
- Designing a robust encoder/decoder is critical
- Challenge: space of algorithms very large



Channel coding

- Central problems in

Channel coding

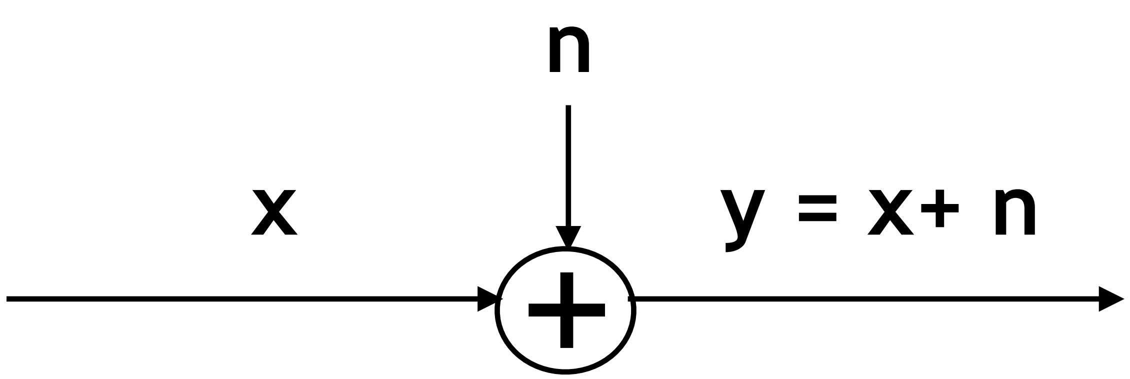
- Central problems in
- Sporadic progress

Figure by Kai Niu

Channel coding

- Classical :

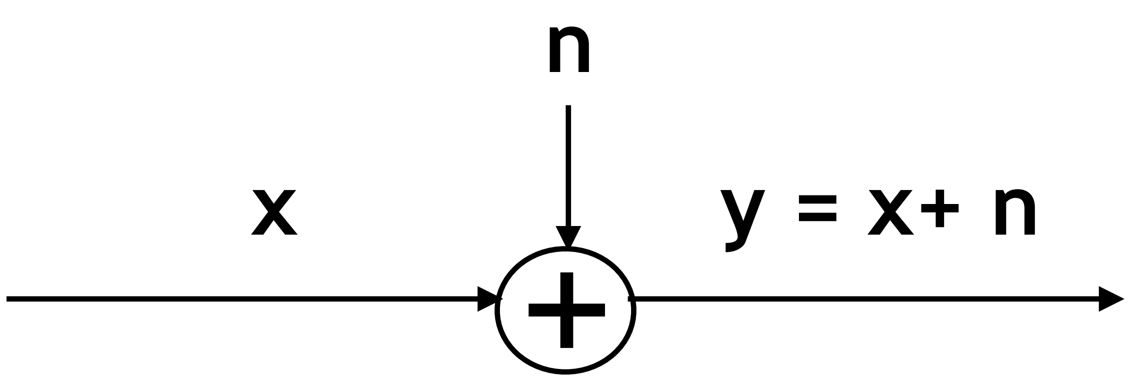
  Additive White Gaussian Noise (AWGN) channels



Channel coding

- Classical :

  Additive White Gaussian Noise (AWGN) channels



- Good codes under AWGN
  - e.g. turbo, LDPC, polar codes

Open problems: type I

- Channel coding (encoder and decoder)
  - Network settings

Open problems: type I

- Channel coding (encoder and decoder)
    - Network settings

    - Channels with feedback

Open problems: type I

- Channel coding (encoder and decoder)
  - Network settings

  - Channels with feedback

  - Deletion/insertion channels

Open problems: type II

- Channel decoding
  - Encoder is fixed (e.g. standardization)

Open problems: type II

- Channel decoding
  - Encoder is fixed (e.g. standardization)
  - Practical channels are not always AWGN
  - Adaptive and robust decoder to non-AWGN channels?

Open problems: type II

- Channel decoding
  - Encoder is fixed (e.g. standardization)
  - Practical channels are not always AWGN
  - Adaptive and robust decoder to non-AWGN channels?
  - Reliable decoder for complicated channels

Central goal

Automate the search for codes and decoders via deep learning

Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

Open problem 1

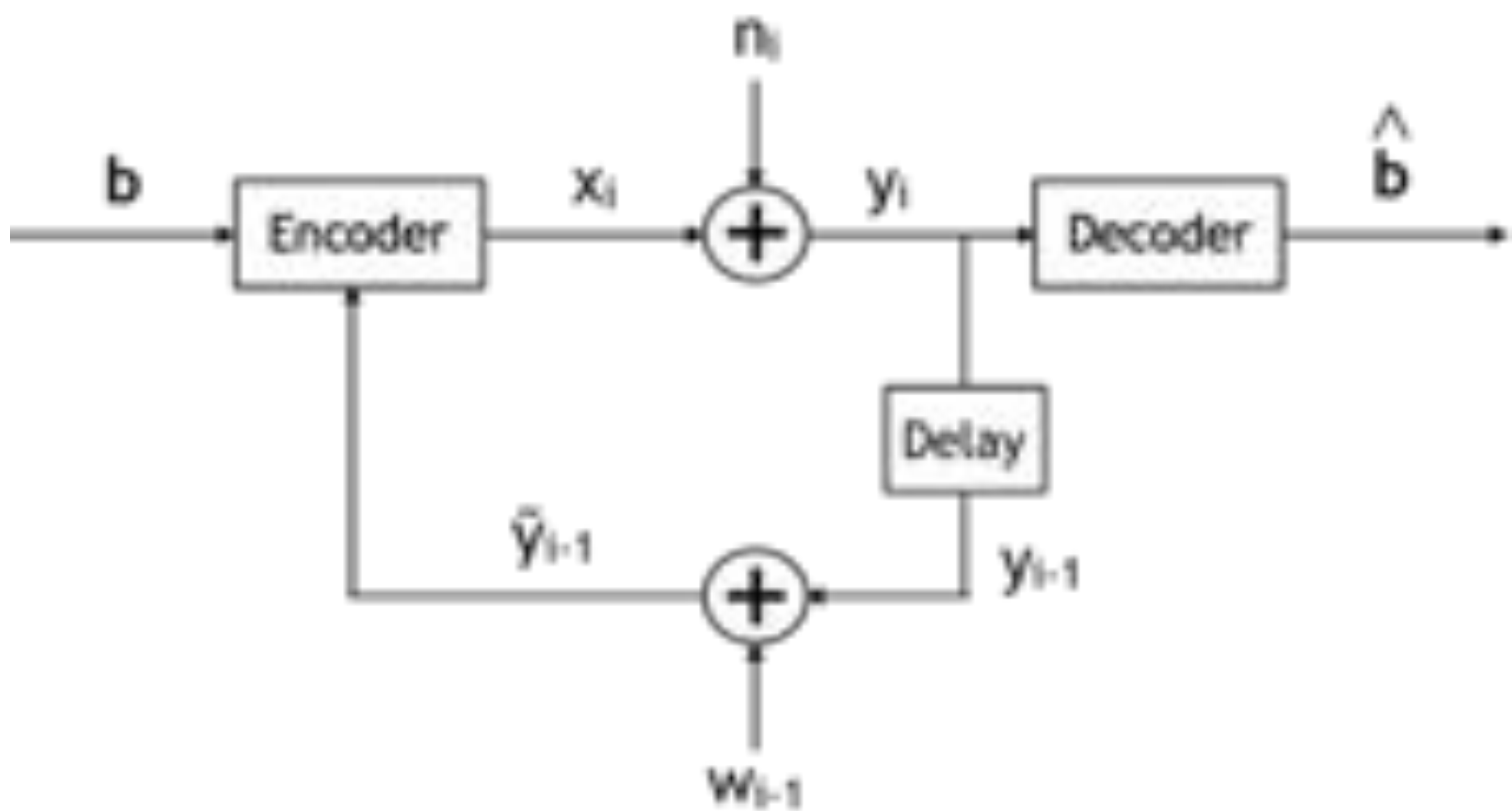
Learning a code

for channels with feedback

H. Kim, Y. Jiang, S. Kannan, S. Oh, P. Viswanath, "***Discovering feedback codes via deep learning***", 2018

AWGN channels with feedback

- AWGN channel from transmitter to receiver
- Output fed back to the transmitter



Literature

- Noiseless feedback
  - Improved reliability
    - BLER decays doubly exponentially in block length

Literature

- Noiseless feedback
  - Improved reliability
    - BLER decays doubly exponentially in block length
  - Coding schemes
    - Schalkwijk-Kailath, ’66
    - Posterior matching

Literature

- Noisy feedback
  - Existing schemes sensitive to noise

Literature

- Noisy feedback
  - Existing schemes sensitive to noise
  - Negative results
  - Linear codes very bad (Kim-Lapidoth-Weissman, ’07)

Literature

- Noisy feedback
  - Existing schemes sensitive to noise
  - Negative results
  - Linear codes very bad (Kim-Lapidoth-Weissman, ’07)

- Widely open

Focus of our work

- AWGN channels with noisy feedback

Focus of our work

- AWGN channels with noisy feedback
- **Challenge**:

  How to combine noisy feedback and message causally?

Focus of our work

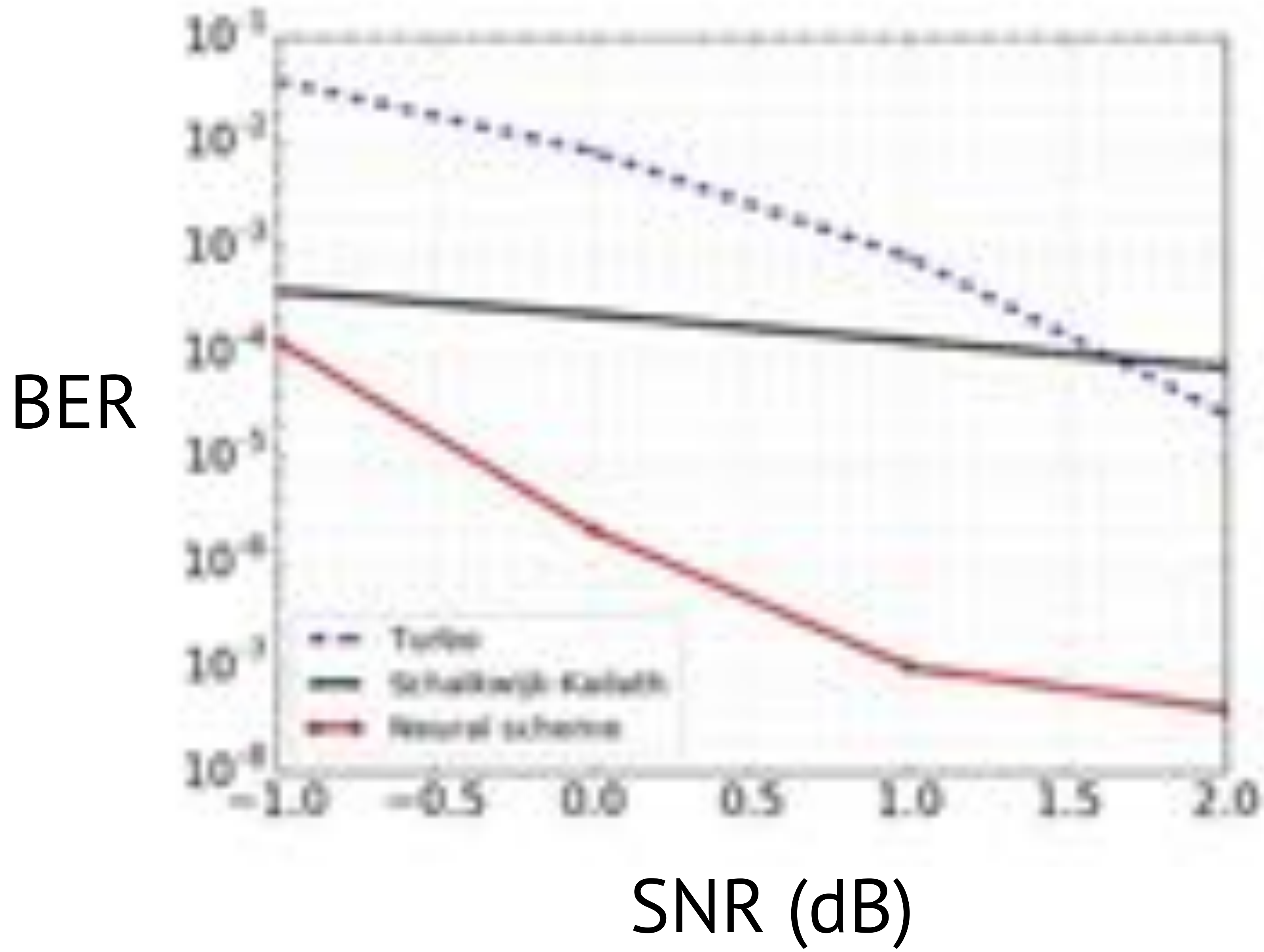
- AWGN channels with noisy feedback
- **Challenge**:

  How to combine noisy feedback and message causally?

- Model encoder and decoder as neural networks and train

Main results

- 100x better reliability under feedback with machine precision

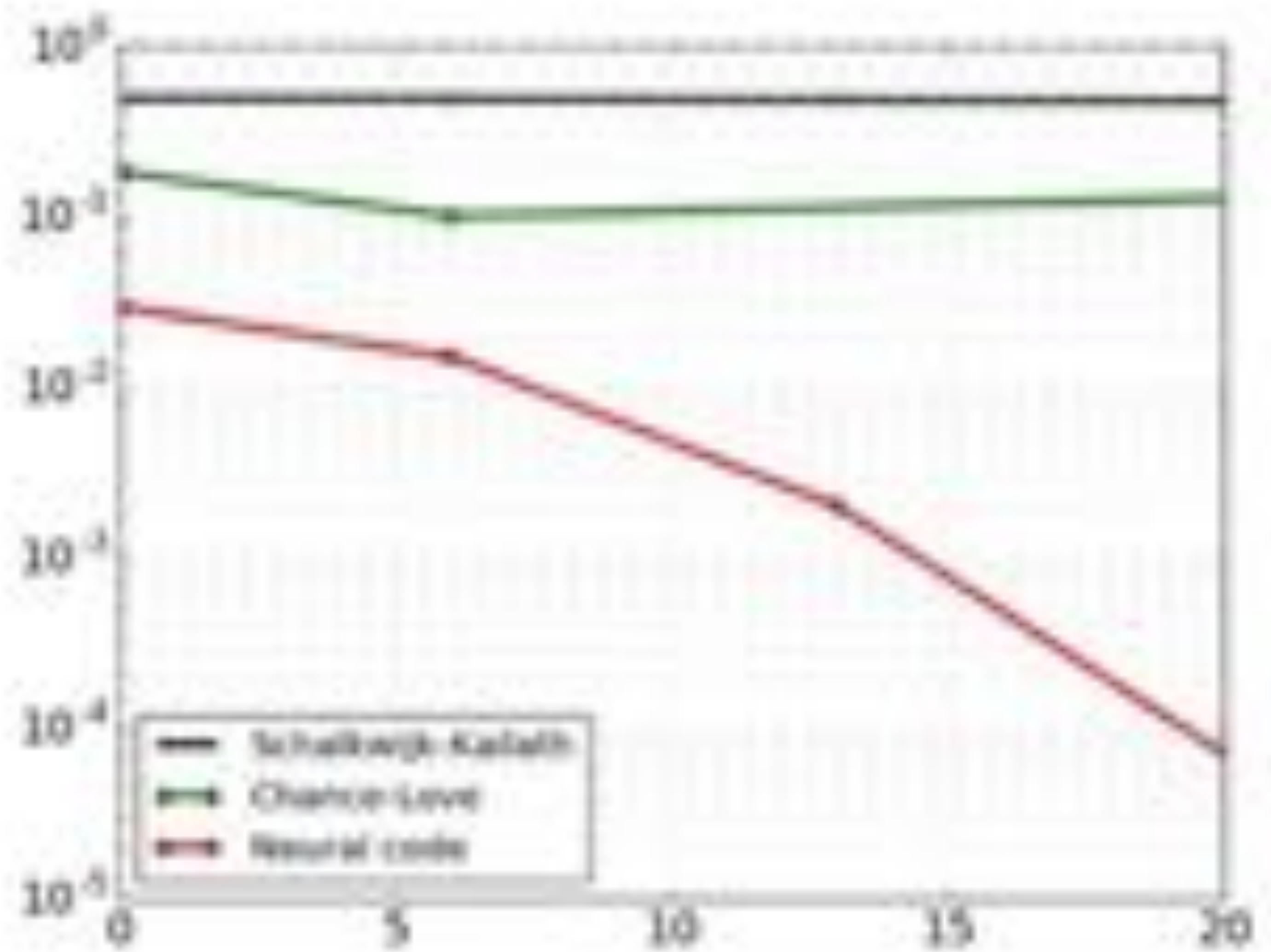


(Rate 1/3, 50 bits)

Main results

- Robust to noise in the feedback

BER



Feedback SNR (dB)

(Rate 1/3, 50 bits, SNR = 0dB)

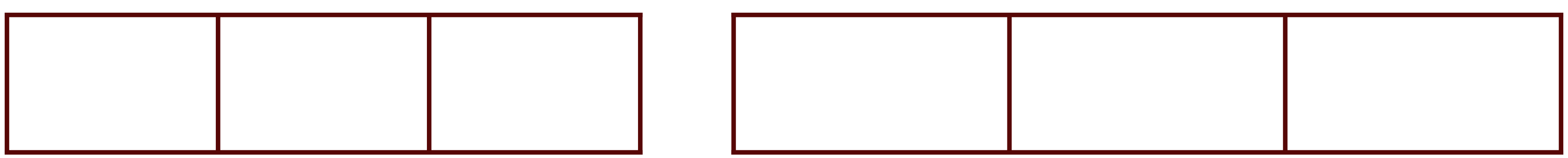
Neural feedback code

Key: Architectural innovations, ideas from communications

Neural encoder

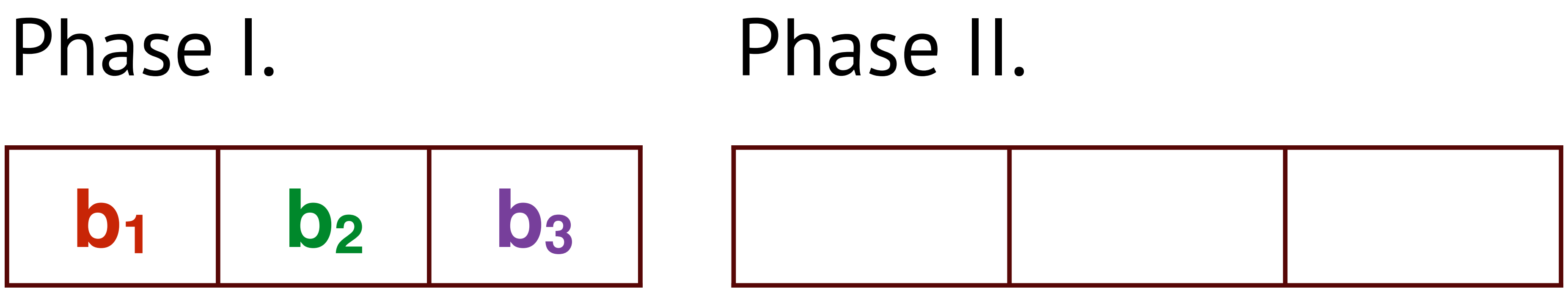
- Two-phase scheme
  - e.g. maps information bits b_1, b_2, b_3 to a length-6 code

Phase I. Phase II.



Neural encoder

- Two-phase scheme
  - e.g. maps information bits b_1, b_2, b_3 to a length-6 code



Neural encoder

- Two-phase scheme
  - e.g. maps information bits b_1, b_2, b_3 to a length-6 code

Phase I.

b_1	b_2	b_3

Phase II.

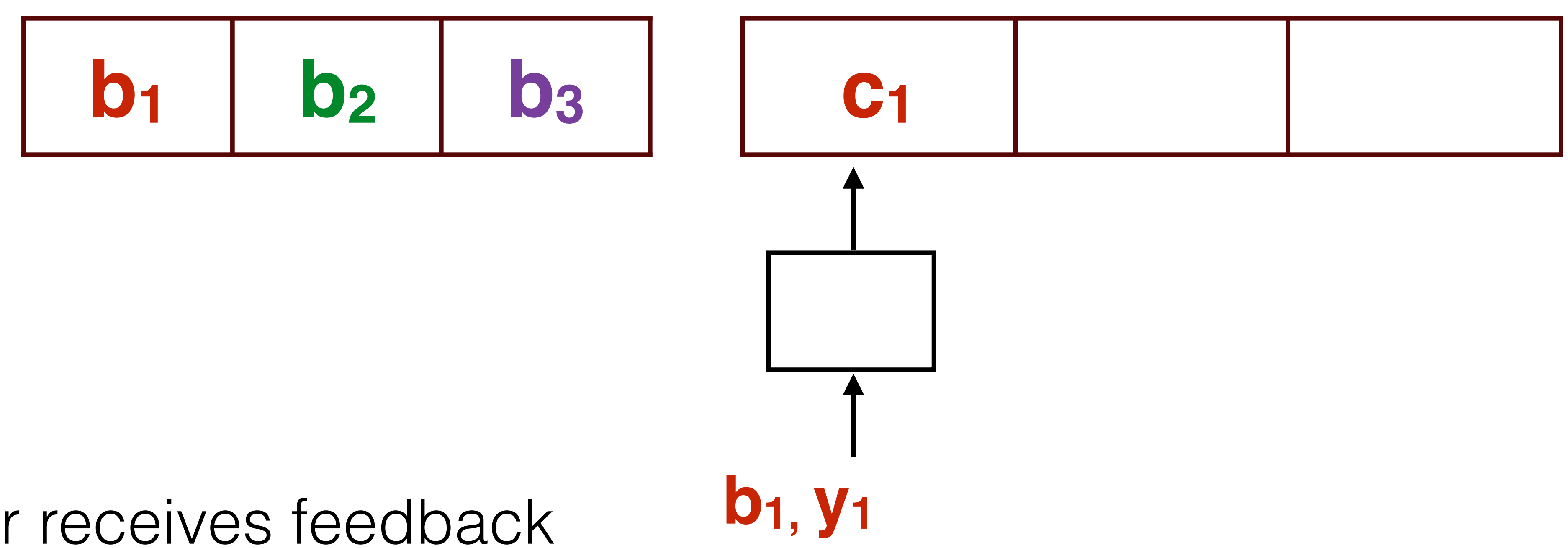
Encoder receives feedback

y_1 y_2 y_3

Phase II: use feedback to generate parity bits

- Parity for b_1

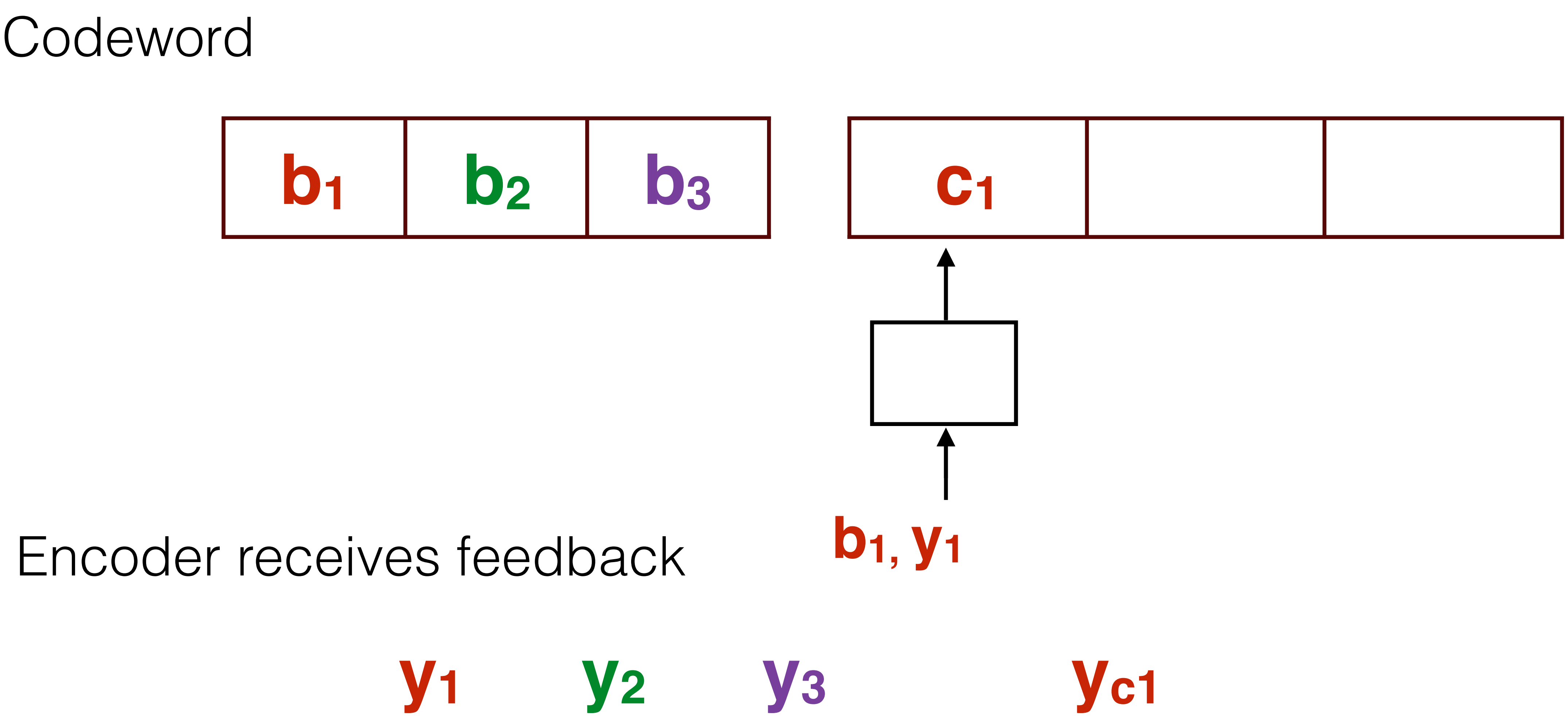
Codeword



Encoder receives feedback

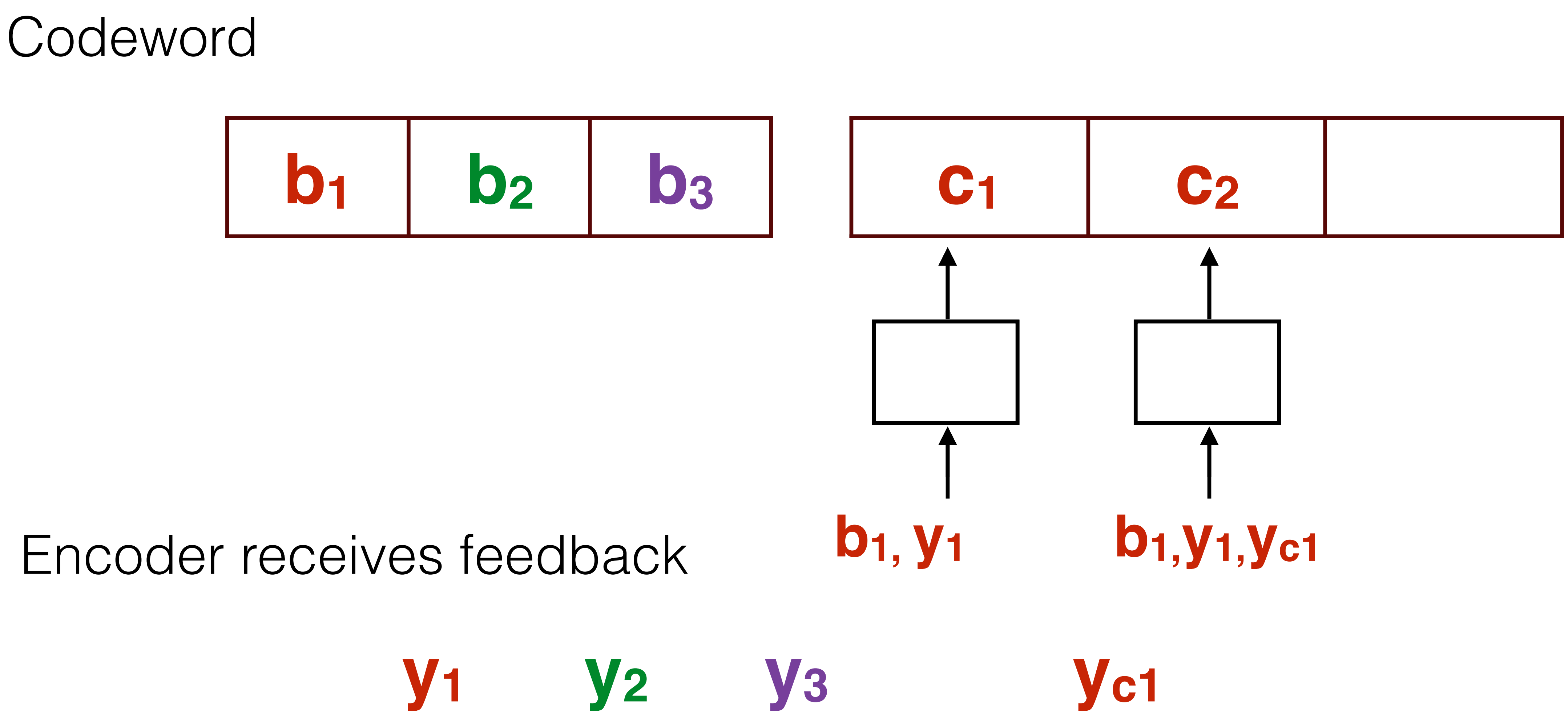
y_1 y_2 y_3

Phase II: use feedback to generate parity bits



Phase II: use feedback to generate parity bits

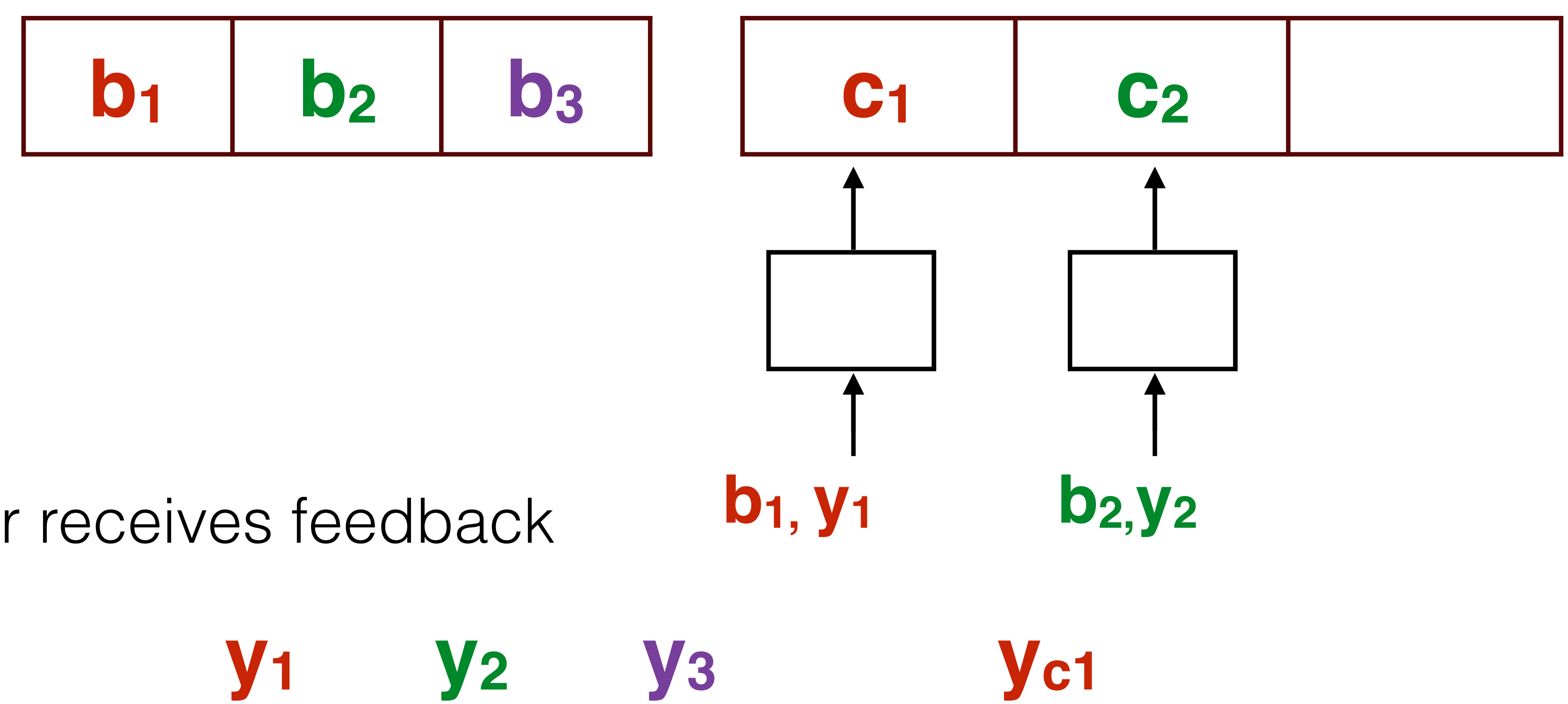
- Another parity for b_1?



Phase II: use feedback to generate parity bits

- Parity for b_2?

Codeword

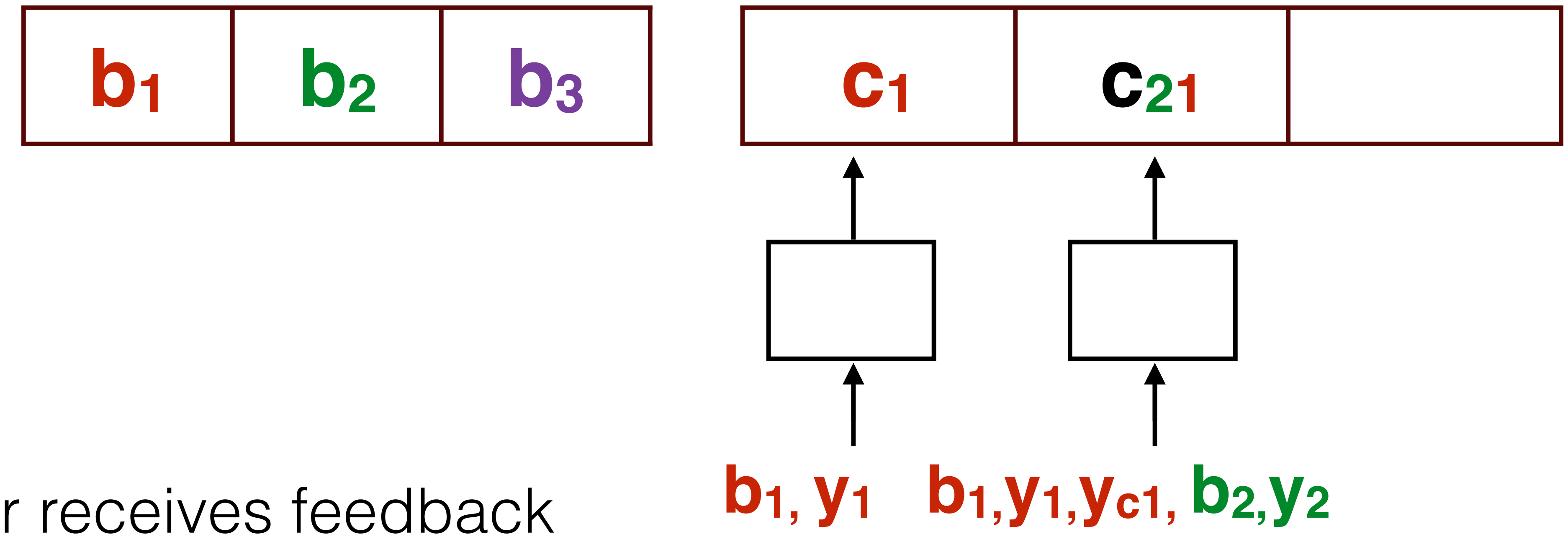


Encoder receives feedback

Phase II: use feedback to generate parity bits

- Parity for b_2 and b_1

Codeword

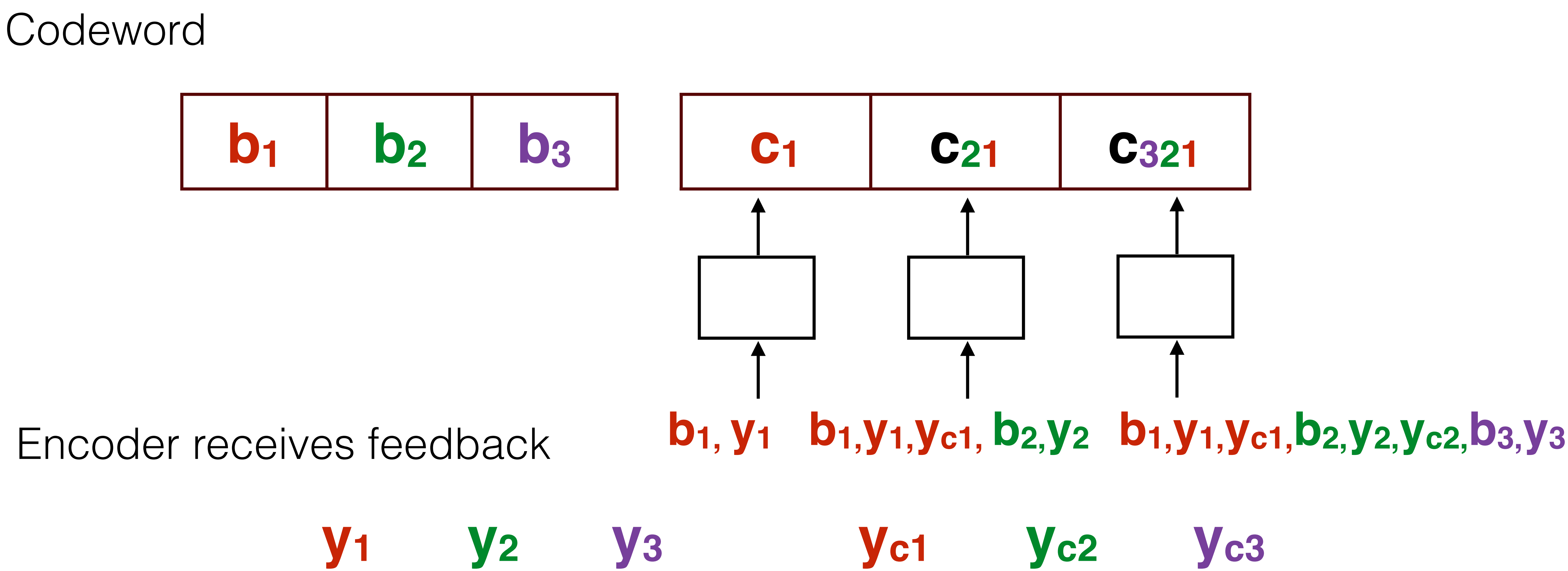


Encoder receives feedback

y_1 y_2 y_3 y_{c1}

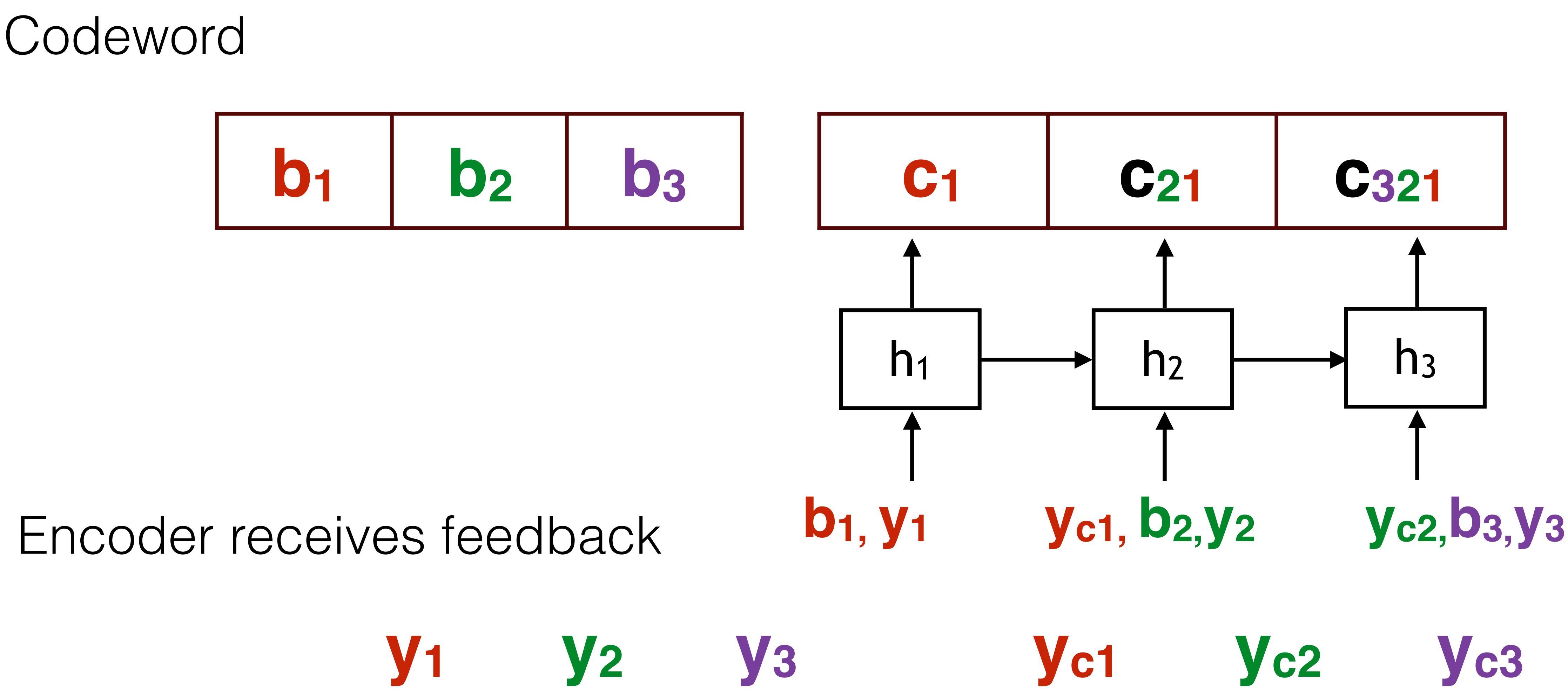
Phase II: use feedback to generate parity bits

- Parity for b_3, b_2 and b_1



Recurrent Neural Network for parity generation

- Sequential mapping with memory



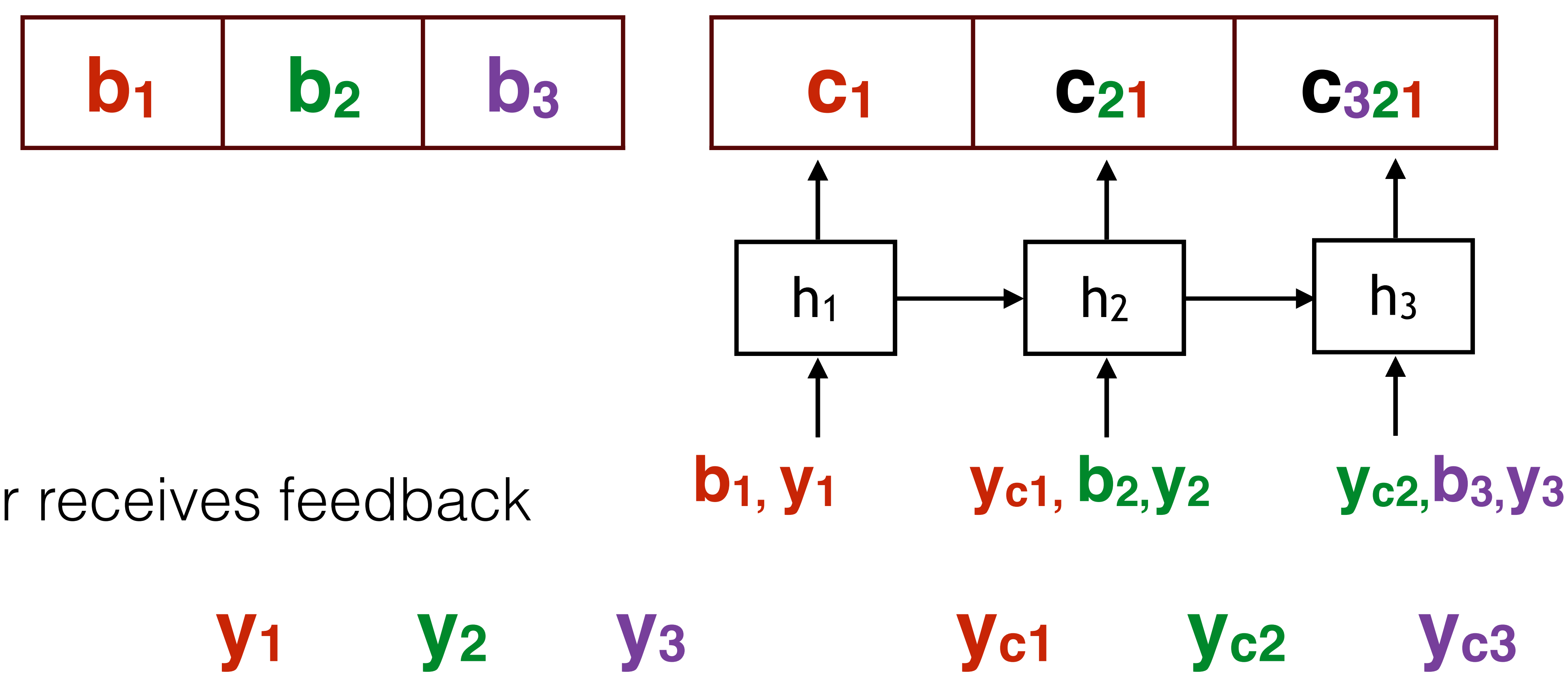
Recurrent Neural Network for parity generation

- Sequential mapping with memory

$$h_i = f(h_{i-1}, \mathrm{Input}_i)$$
$$\mathrm{Output}_i = g(h_i)$$

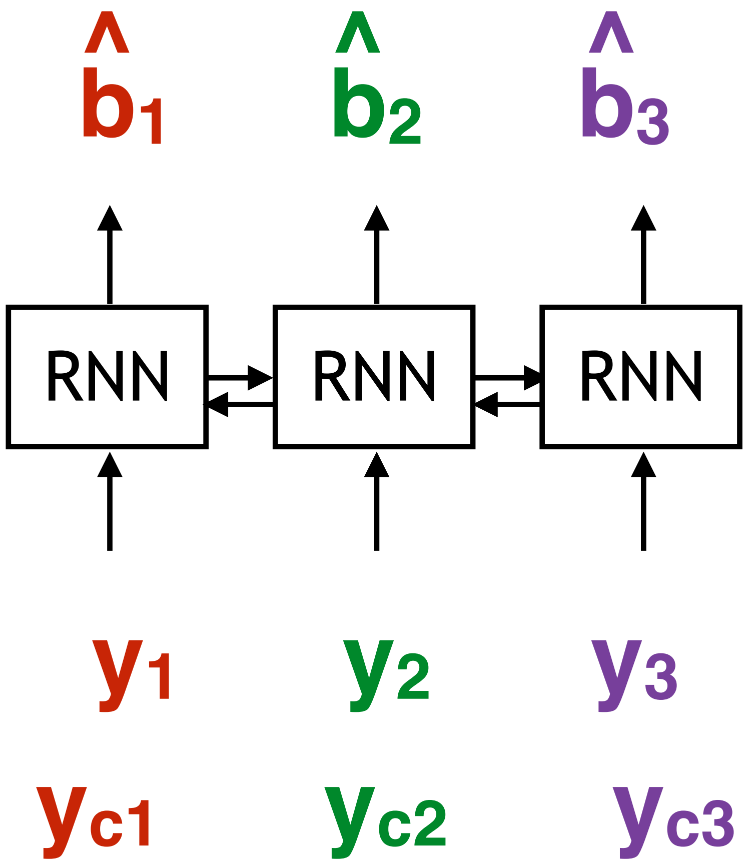
Codeword

Encoder receives feedback



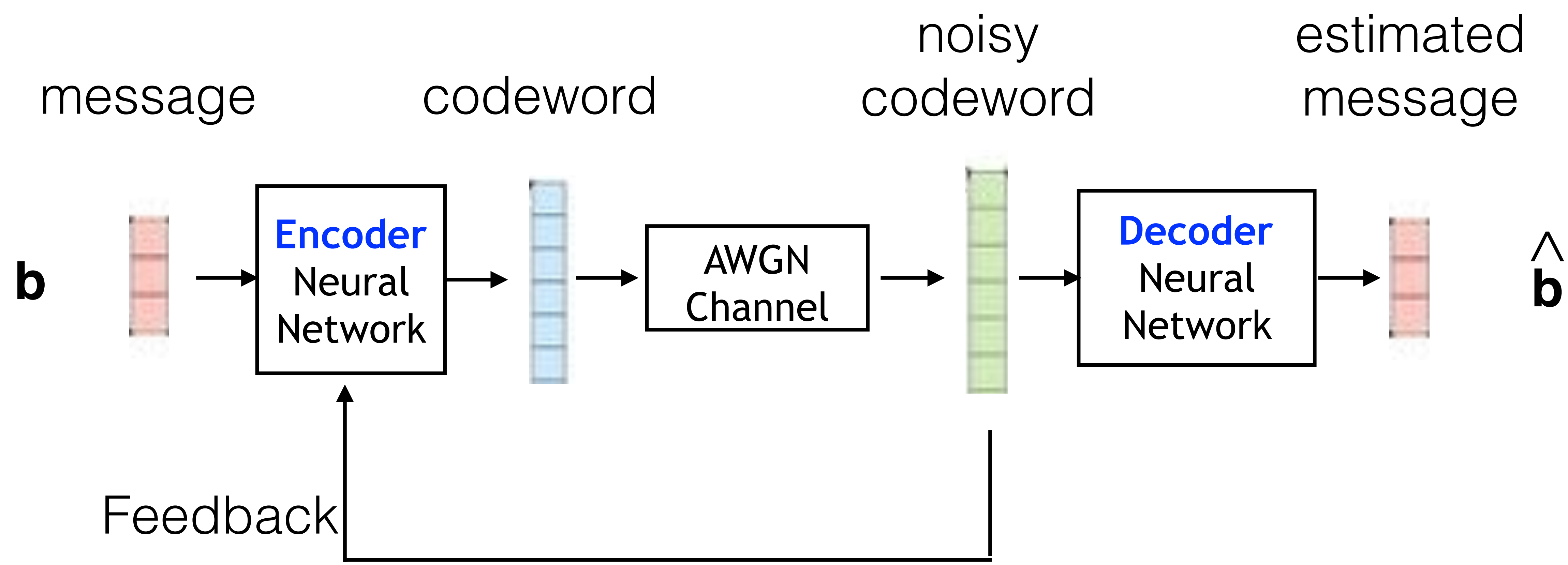
Neural decoder

- Maps $(y_1, y_2, y_3, y_{c1}, y_{c2}, y_{c3})$ to $\hat{b}_1, \hat{b}_2, \hat{b}_3$ via bi-direct. RNN



Training

- Learn the encoder and decoder jointly



Training

- Auto-encoder training : (input,output) = (**b**,**b**)

$$\mathbf{b} = (b_1, b_2, \cdots, b_K)$$

- Loss : binary cross entropy

$$\mathcal{L}(\mathbf{b}, \hat{\mathbf{b}}) = -\mathbf{b}\log\hat{\mathbf{b}} - (1-\mathbf{b})\log(1-\hat{\mathbf{b}})$$

Training

- Auto-encoder training : (input,output) = (**b**,**b**)

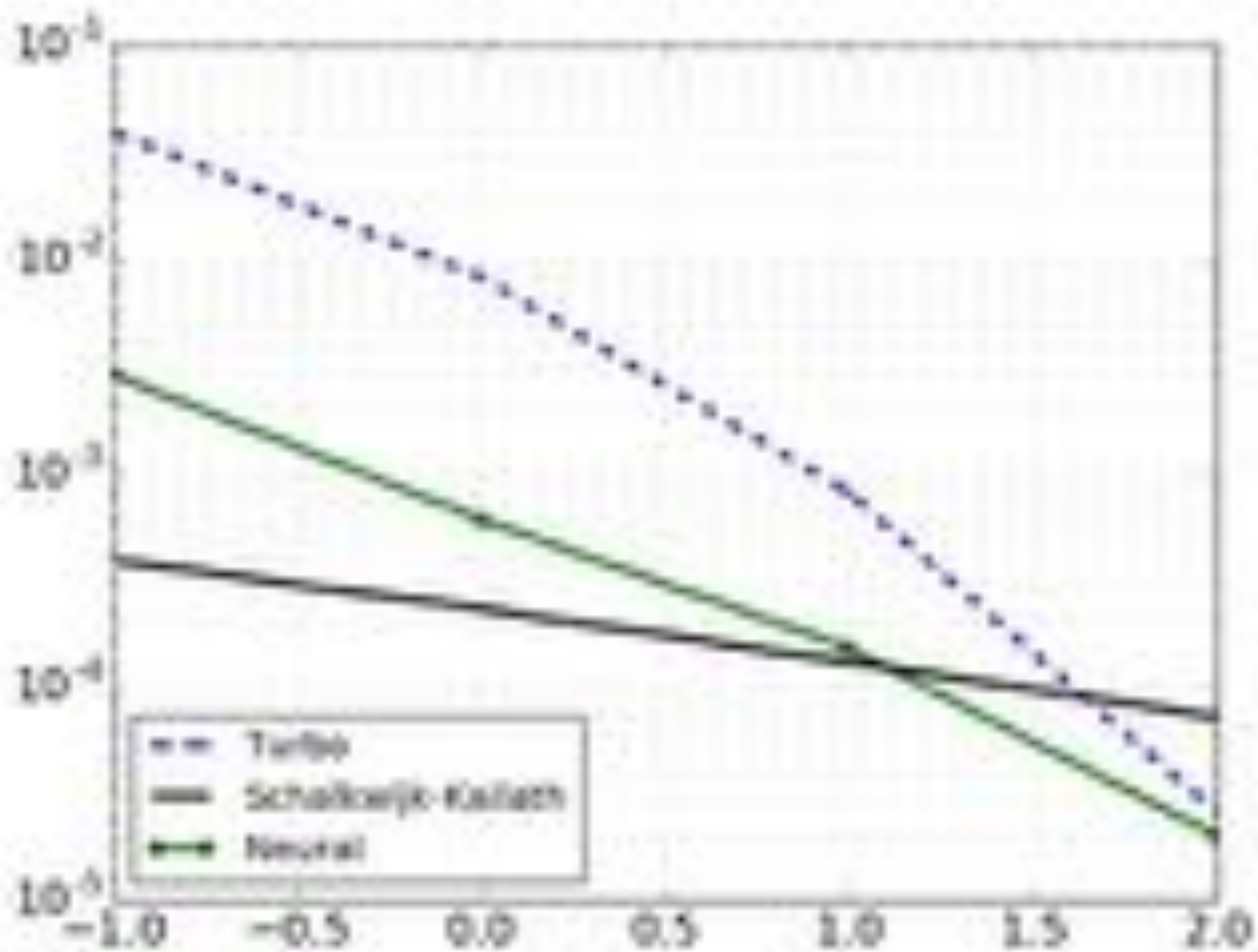
$$\mathbf{b} = (b_1, b_2, \cdots, b_K)$$

- Loss : binary cross entropy

$$\mathcal{L}(\mathbf{b}, \hat{\mathbf{b}}) = -\mathbf{b} \log \hat{\mathbf{b}} - (1 - \mathbf{b}) \log(1 - \hat{\mathbf{b}})$$

- Length of training examples :
  - Block length K has to be long enough (100)

Intermediate results

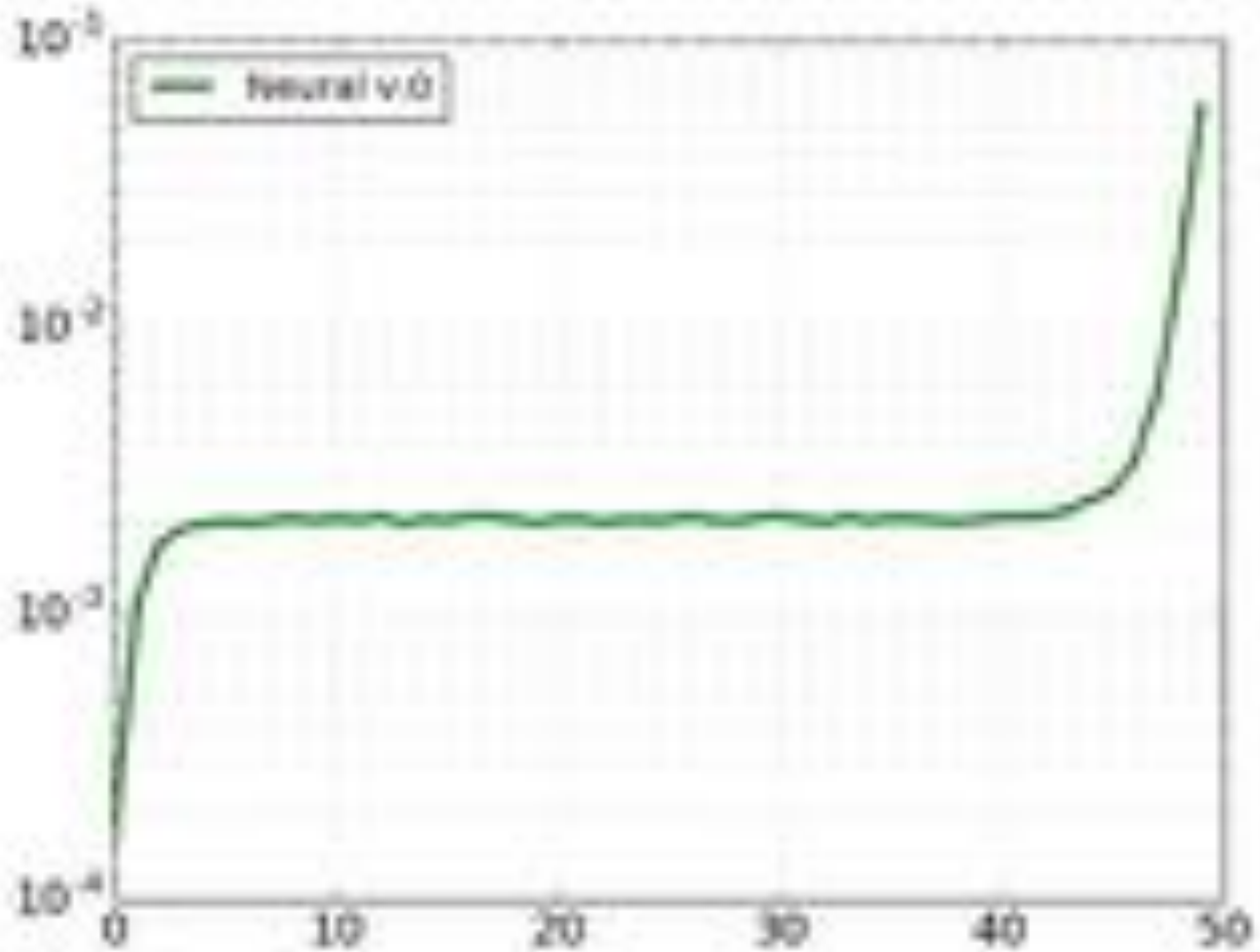


BER

SNR (dB)

High error in the last bits

BER



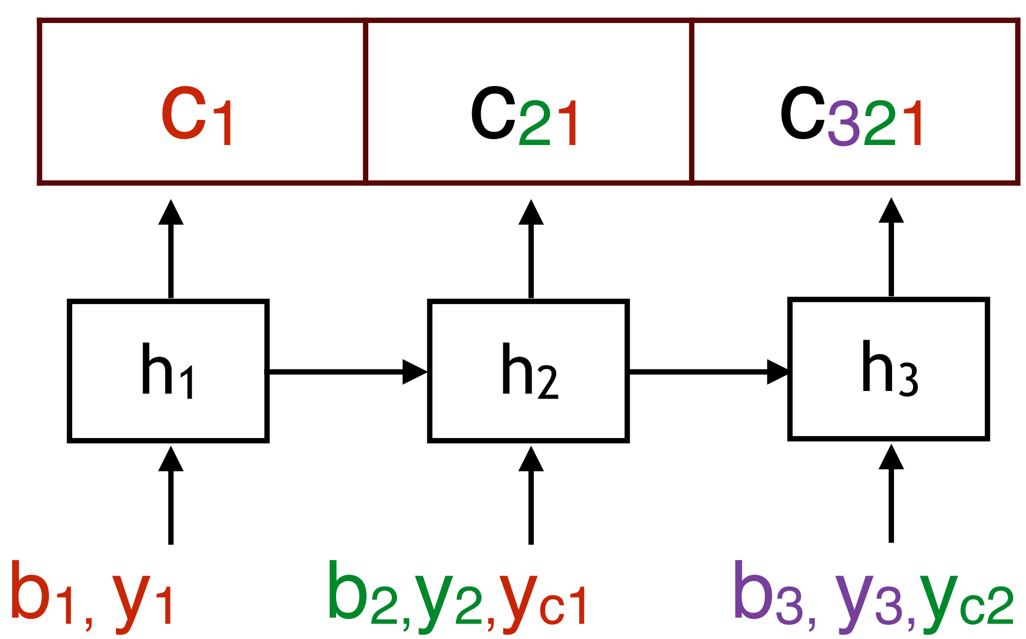
Position

High error in the last bits

Phase I.

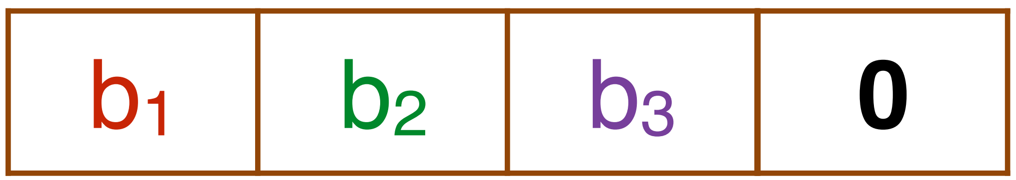
b_1	b_2	b_3

Phase II.

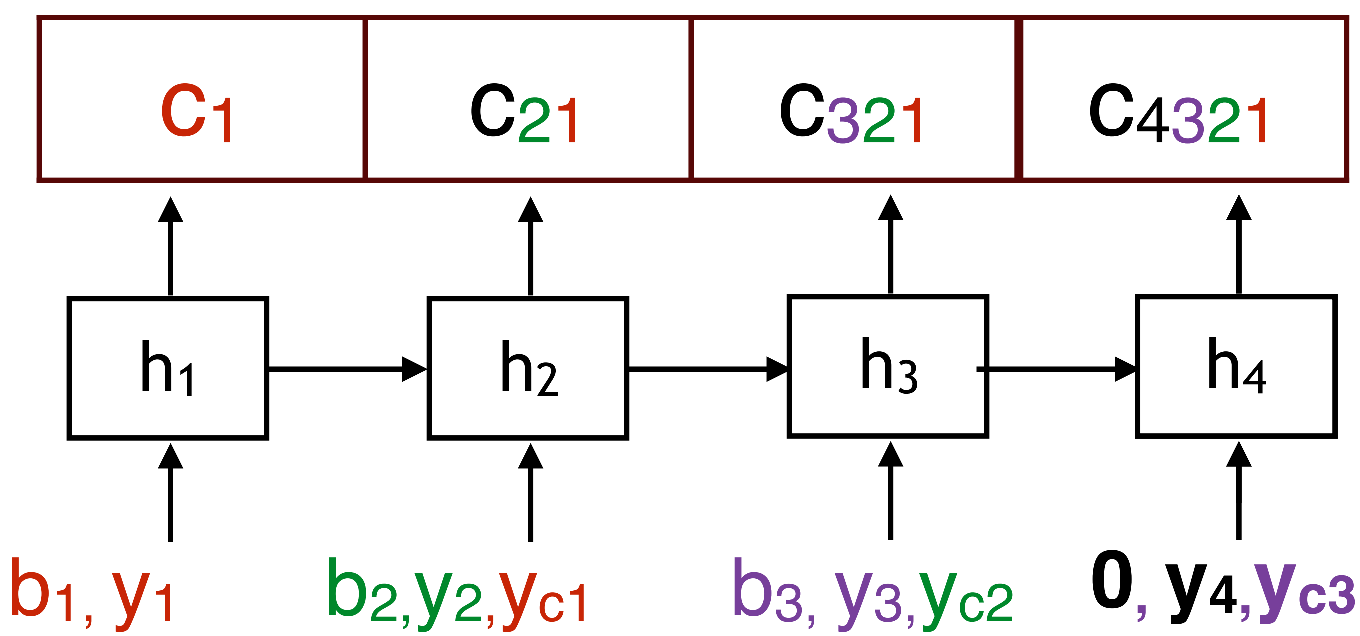


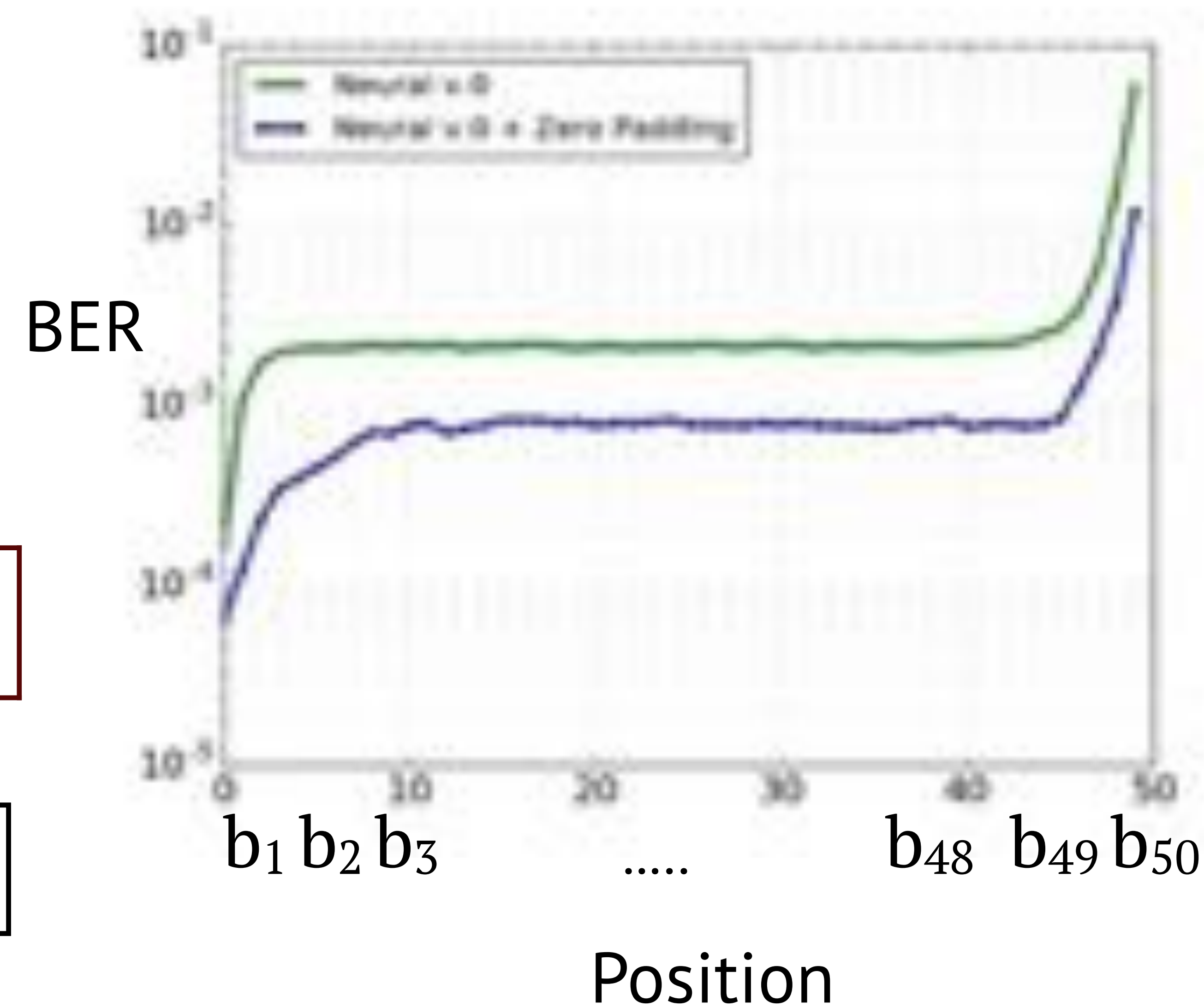
Idea 1. Zero padding

Phase I.



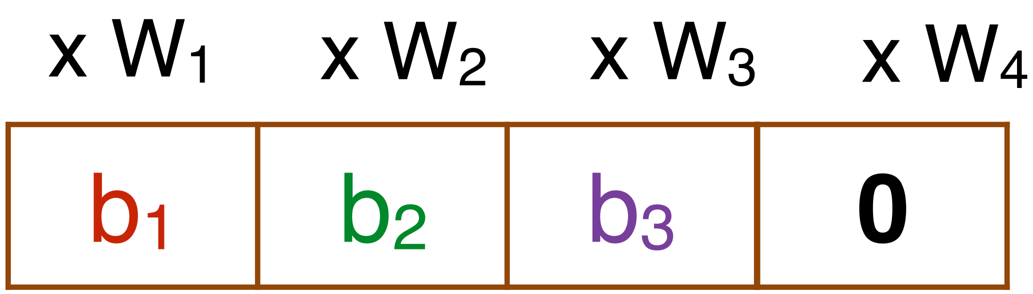
Phase II.



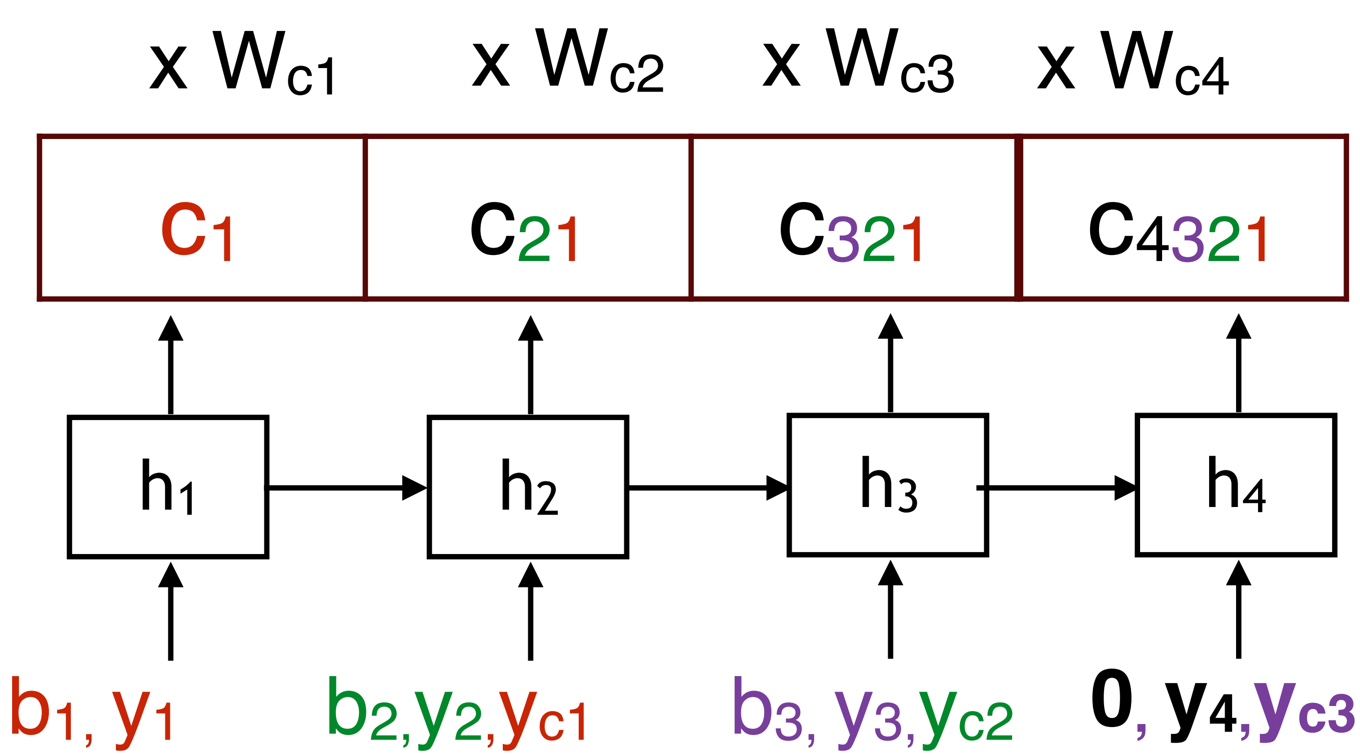


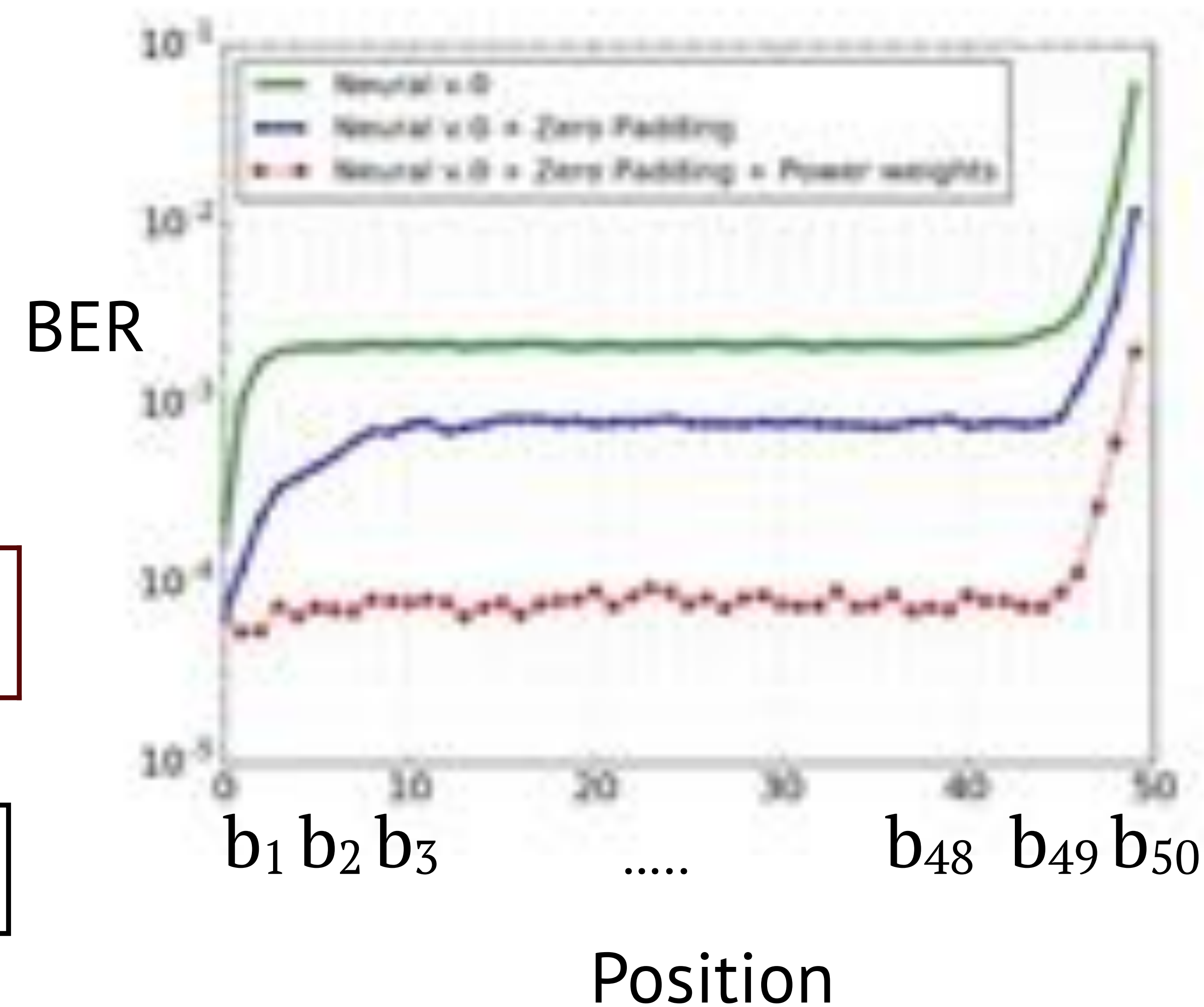
Idea 2. Power allocation

Phase I.



Phase II.

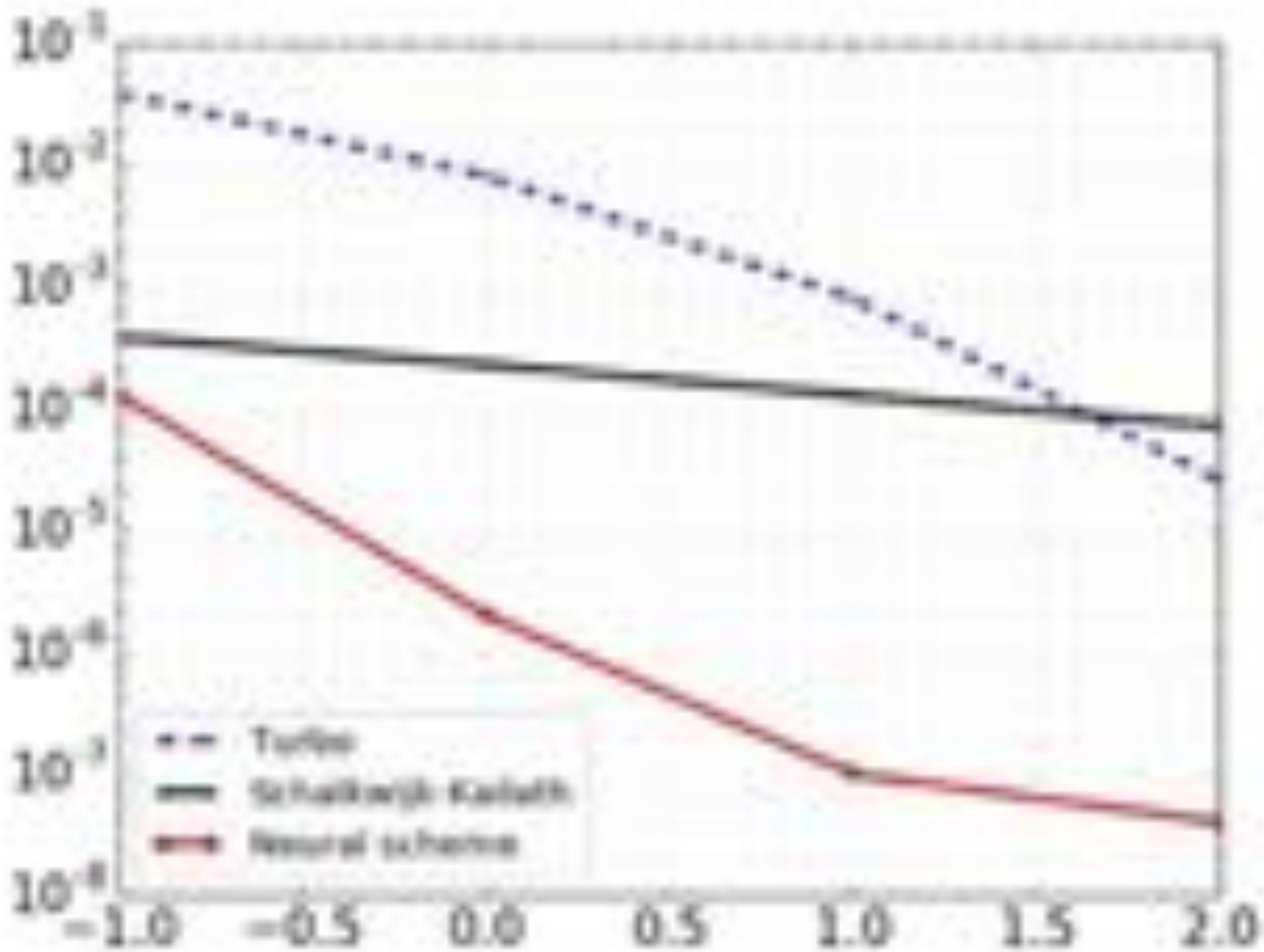




Results

- 100x better reliability under feedback w. machine precision

BER



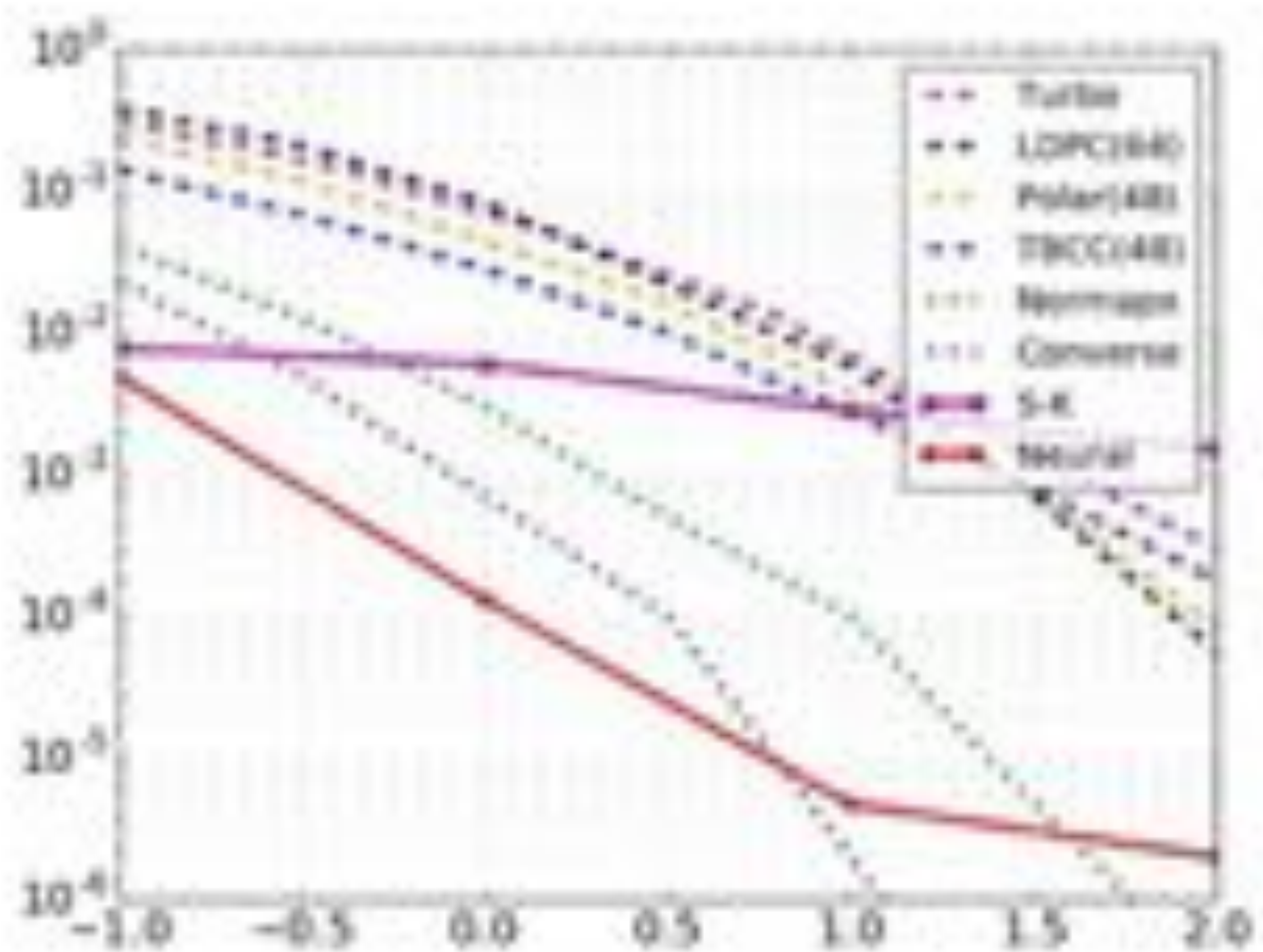
SNR (dB)

(Rate 1/3, 50 bits)

Results

- 100x better reliability under feedback w. machine precision

BLER



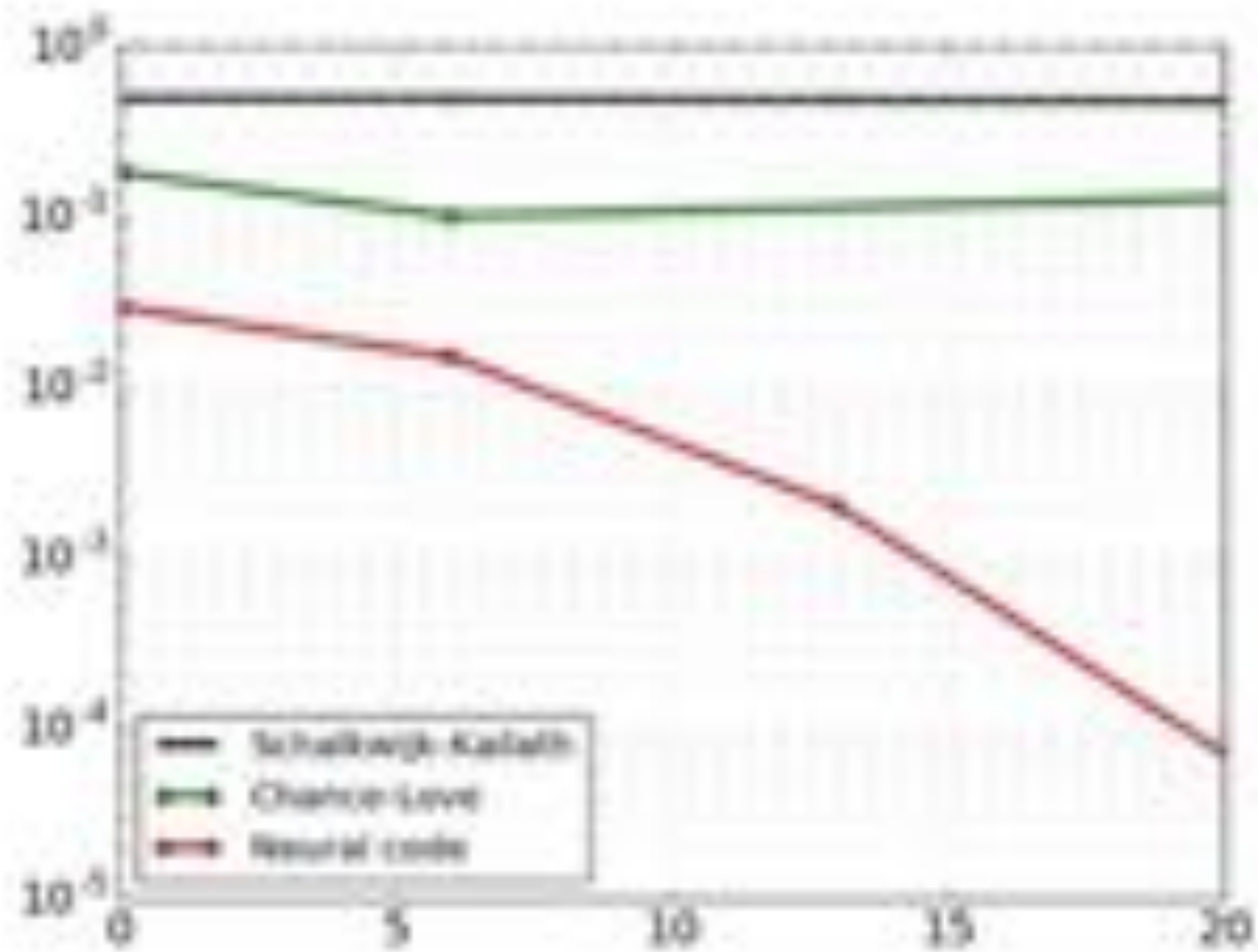
SNR (dB)

(Rate 1/3, 50 bits)

Results

- Robust to noise in the feedback

BER



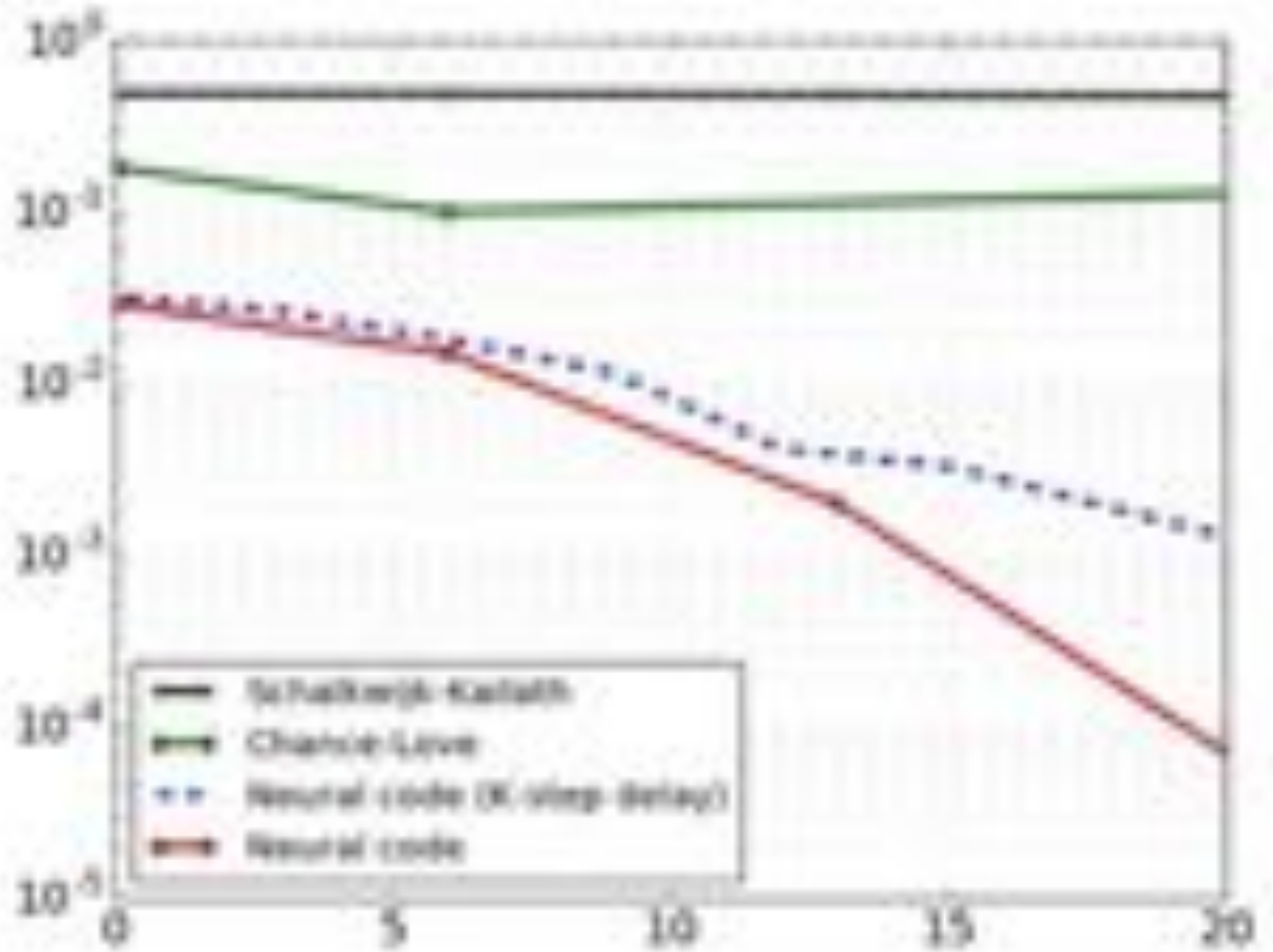
Feedback SNR (dB)

(Rate 1/3, 50 bits, 0dB)

Results

- Delayed feedback

BER



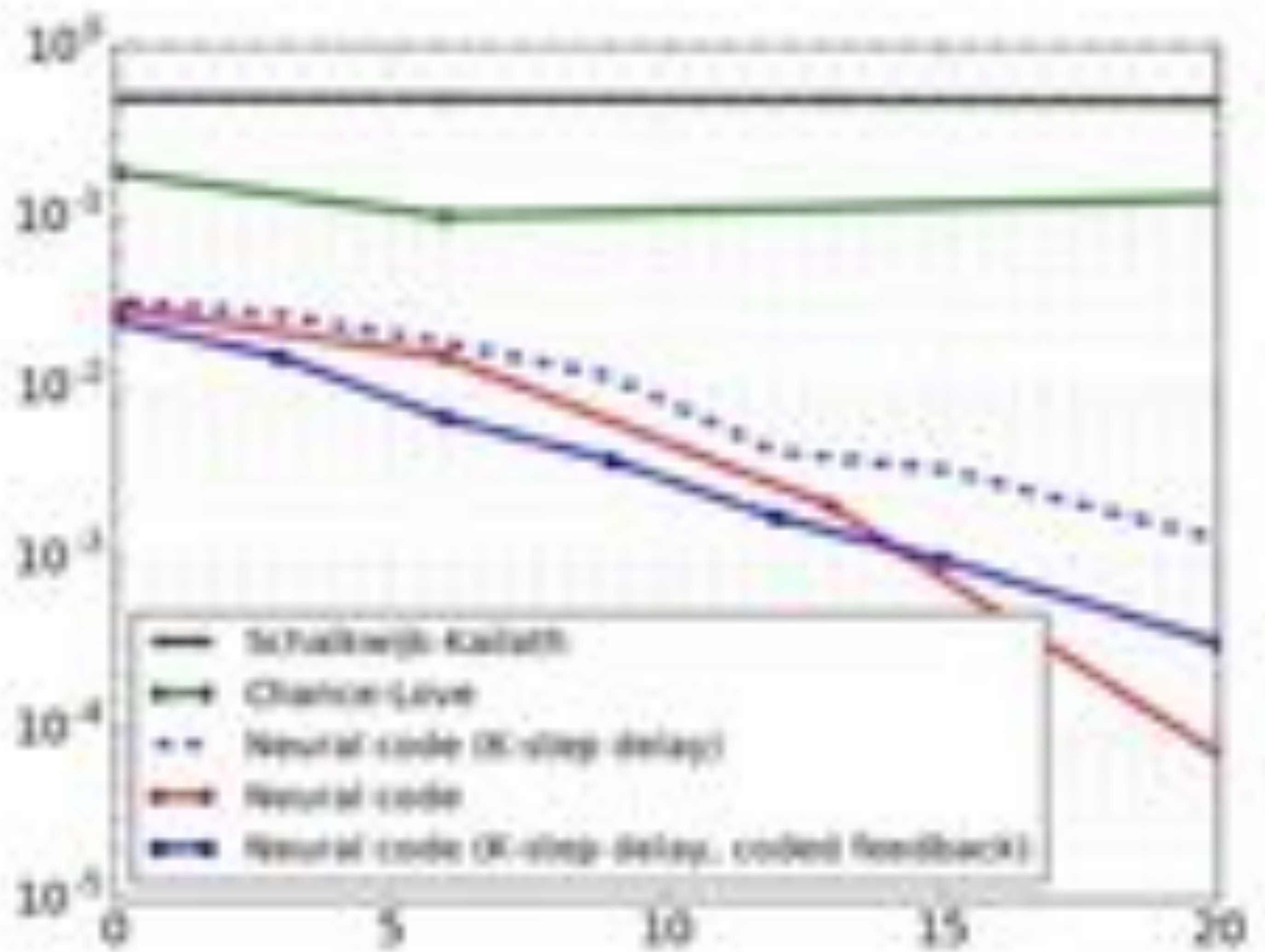
Feedback SNR (dB)

(Rate 1/3, 50 bits, 0dB)

Results

- Delayed and coded feedback

BER

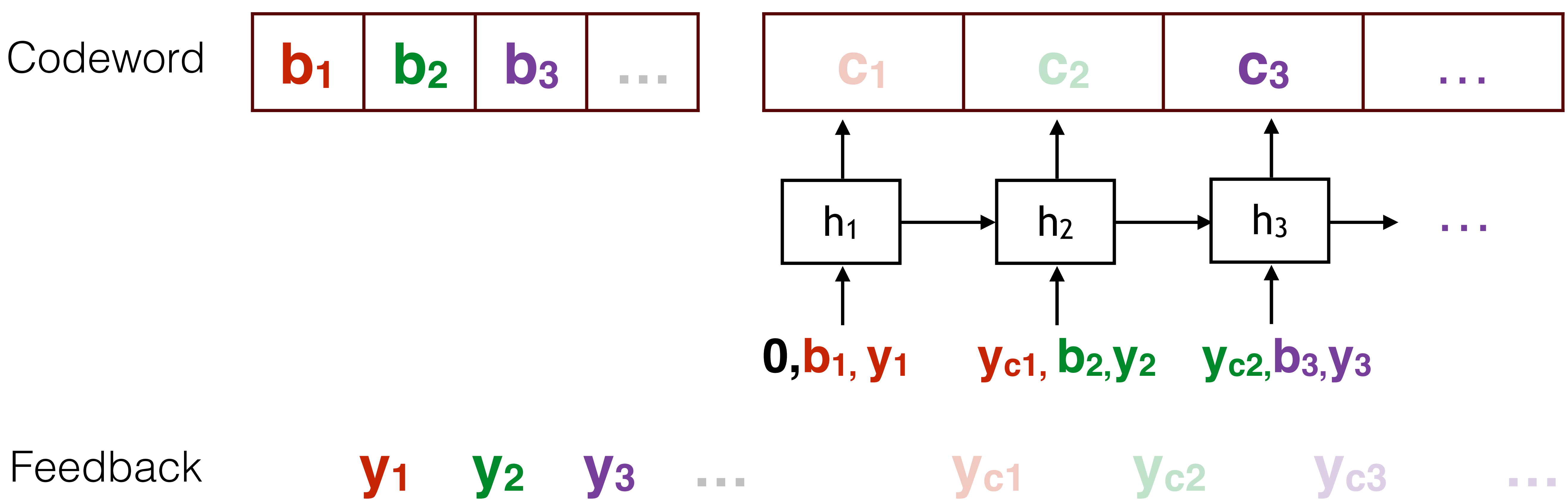


Feedback SNR (dB)

(Rate 1/3, 50 bits, 0dB)

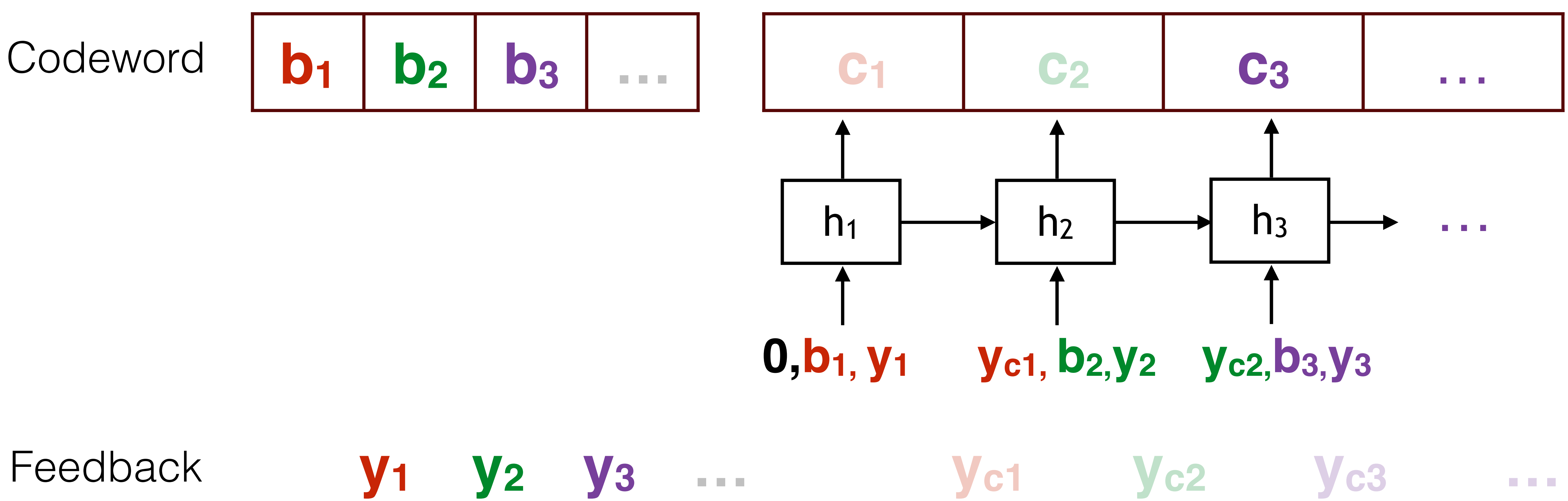
Interpretation of neural codes

- How does parity c_3 depend on b_3, y_3, b_2, y_2, y_{c2}, b_1, y_1, y_{c1}



Interpretation of neural codes

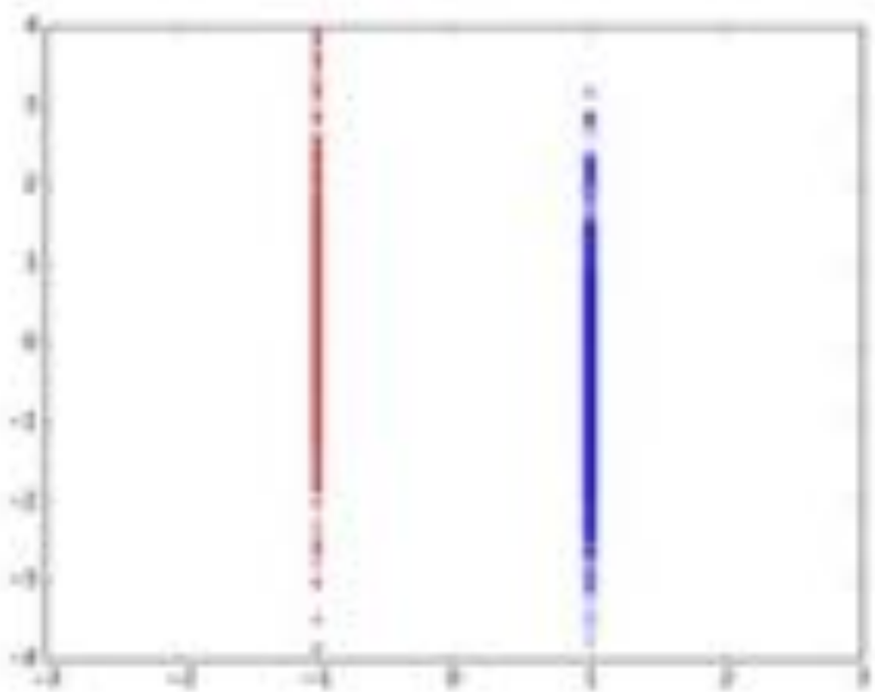
- How does parity c_3 depend on b_3, y_3, b_2, y_2, y_{c2}, b_1, y_1, y_{c1}
- For a rate 1/3 code, $c_k = (c_{k,1}, c_{k,2})$



Interpretation of neural codes

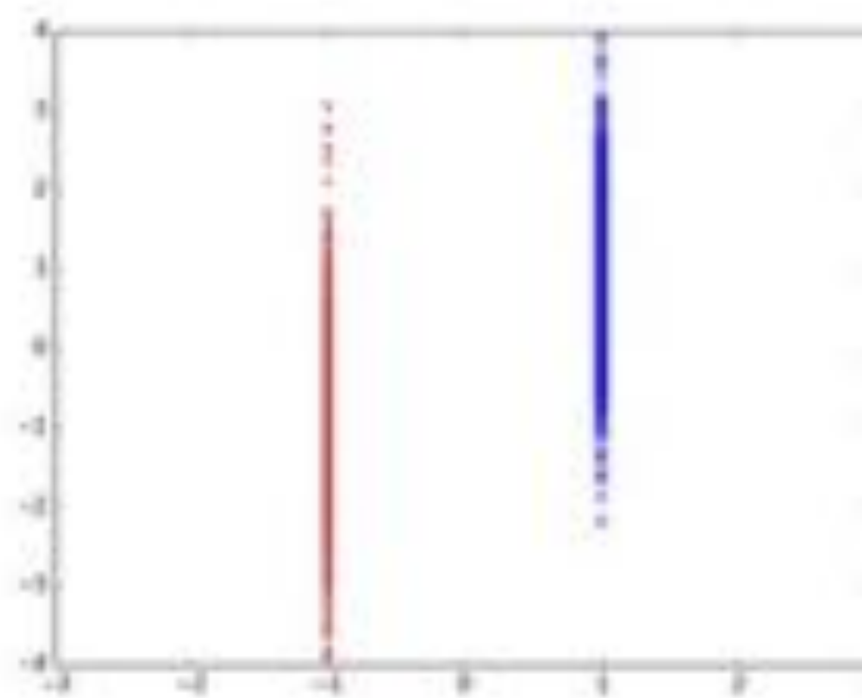
- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on b_k?

$c_{k,1}$



b_k

$c_{k,2}$

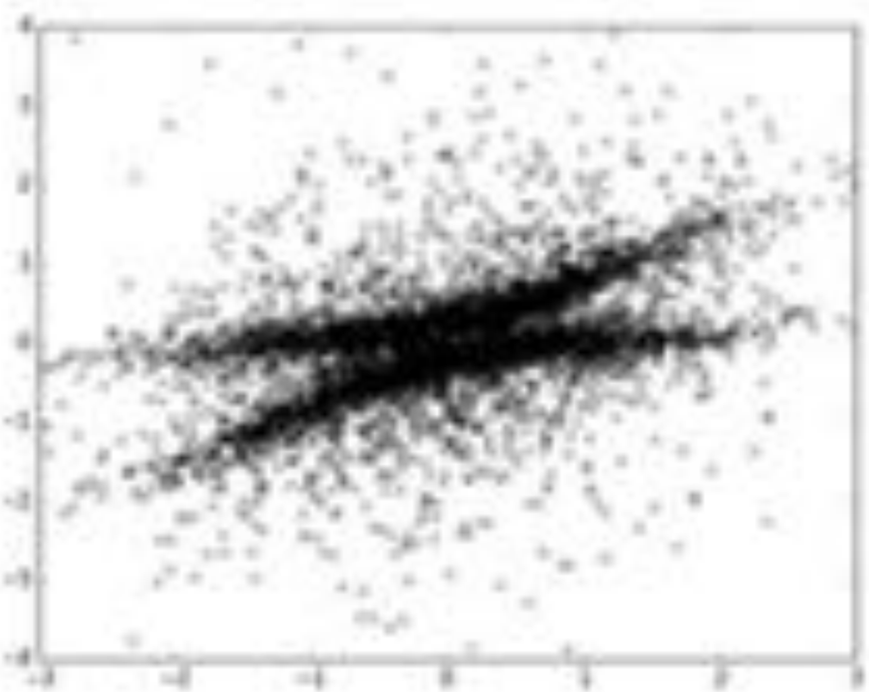


b_k

Interpretation of neural codes

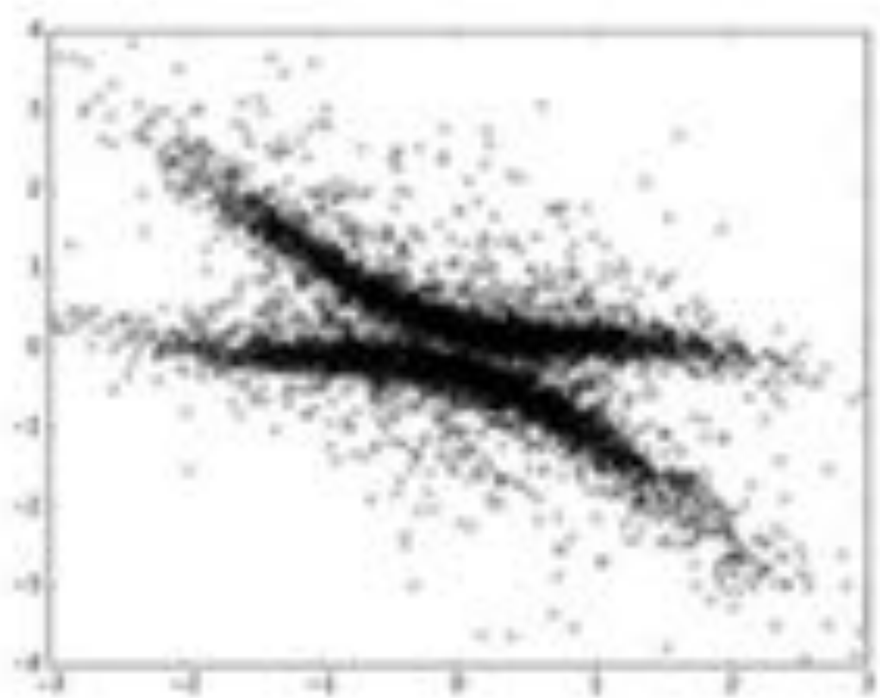
- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on $y_k - b_k$?

$c_{k,1}$



$y_k - b_k$

$c_{k,2}$

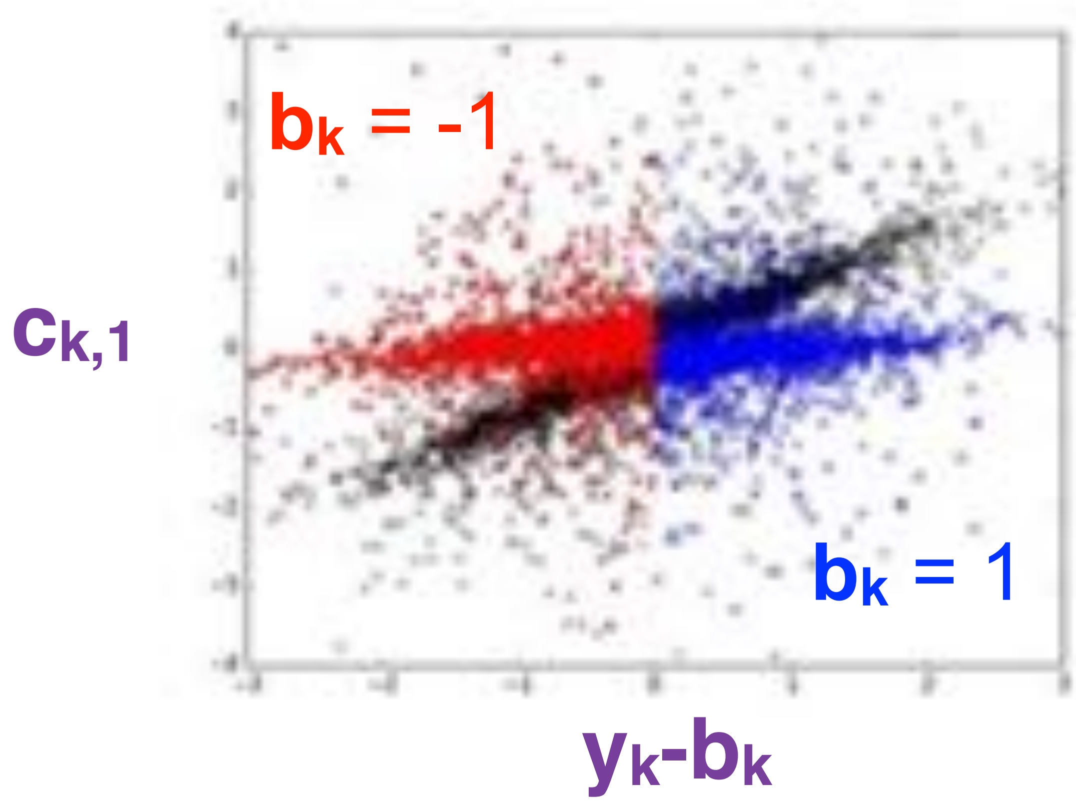


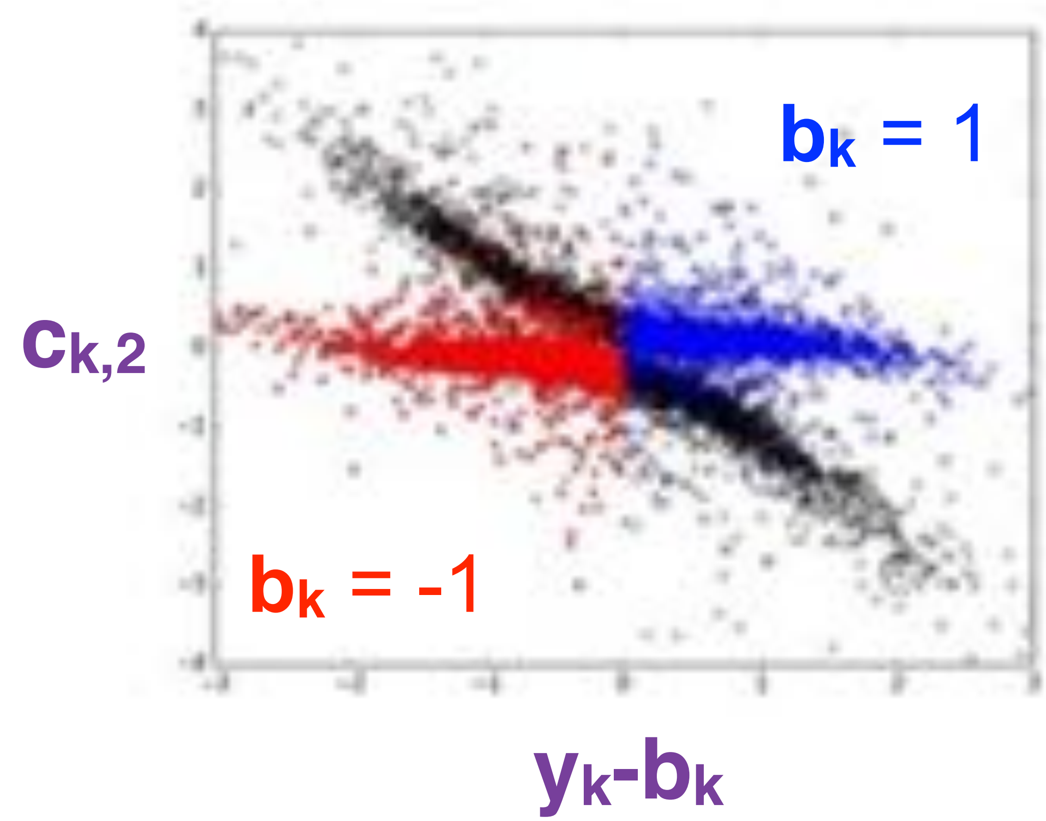
$y_k - b_k$

($y_k - b_k$: noise added to b_k in Phase I)

Interpretation of neural codes

- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on y_k-b_k?

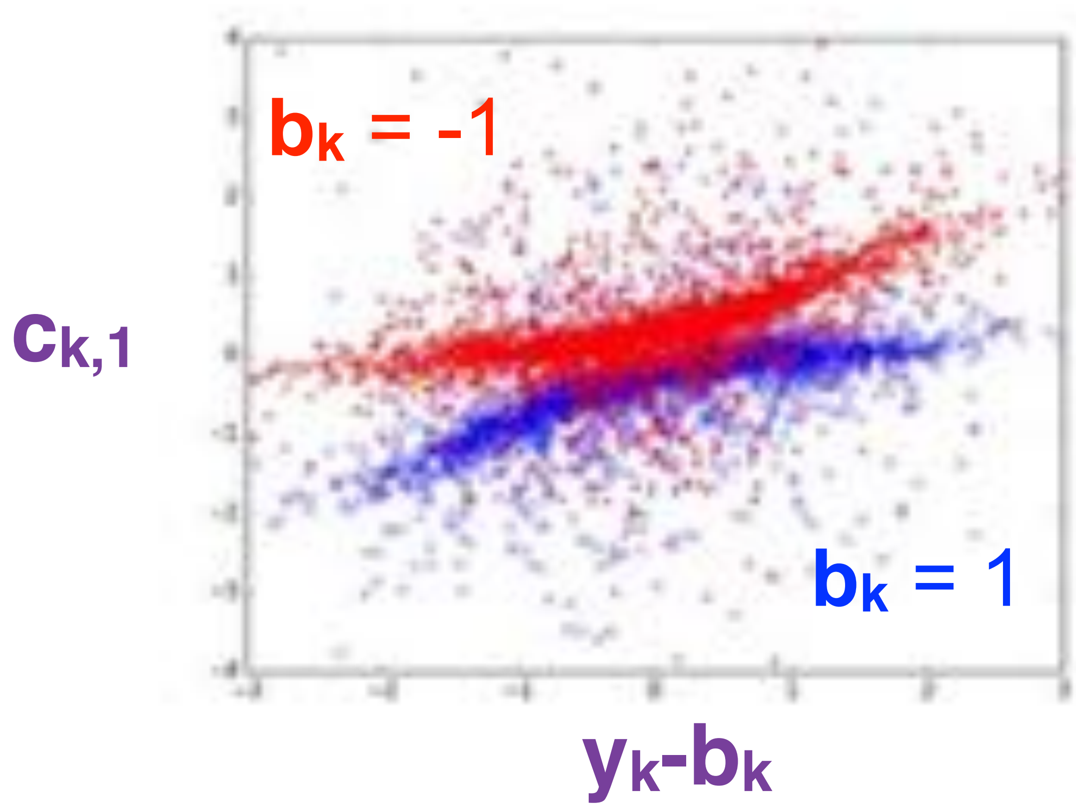


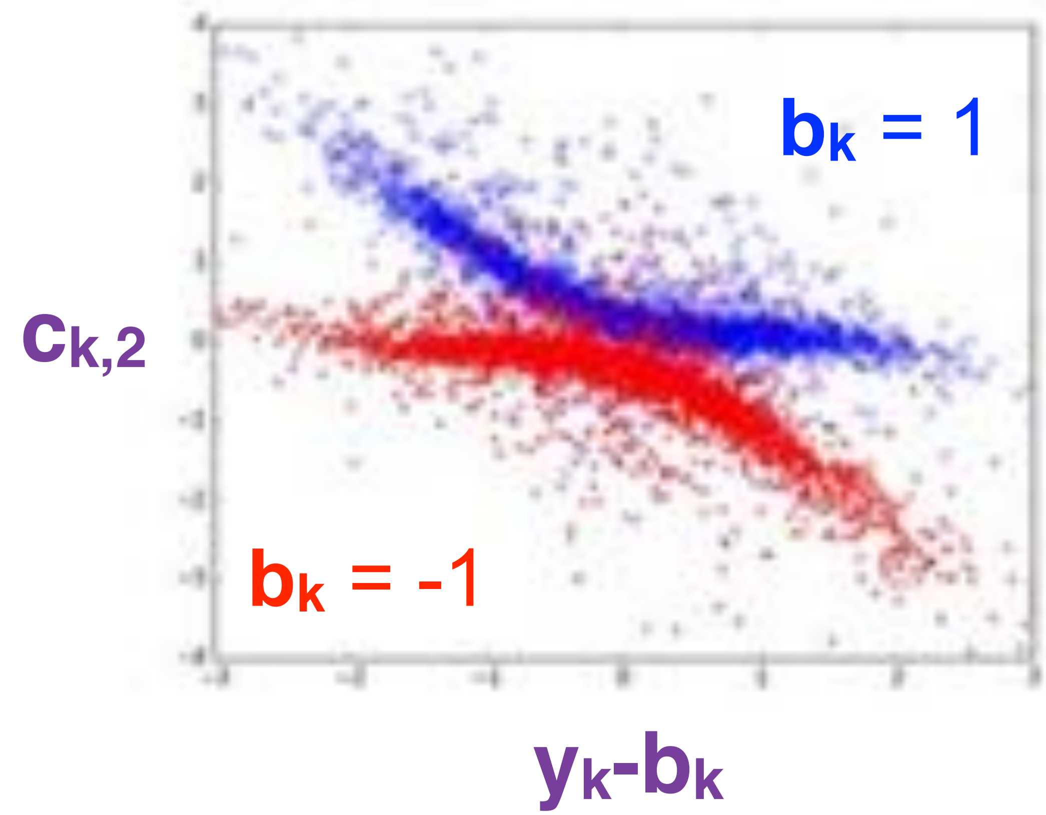


(y_k-b_k: noise added to b_k in Phase I)

Interpretation of neural codes

- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on $y_k - b_k$?





(y_k-b_k: noise added to b_k in Phase I)

Interpretation of neural codes

- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on past bits/noise $b_{k-1}, y_{k-1}, y_{c,k-1}, \ldots, b_1, y_1, y_{c1}$

Interpretation of neural codes

- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on past bits/noise $b_{k-1}, y_{k-1}, y_{c,k-1}, \ldots, b_1, y_1, y_{c1}$
  - e.g., $E[c_{k,1} b_{k-1}]=-0.24$, $E[c_{k,1} b_{k-2}]=-0.1$, $E[c_{k,1} b_{k-3}]= -0.05$

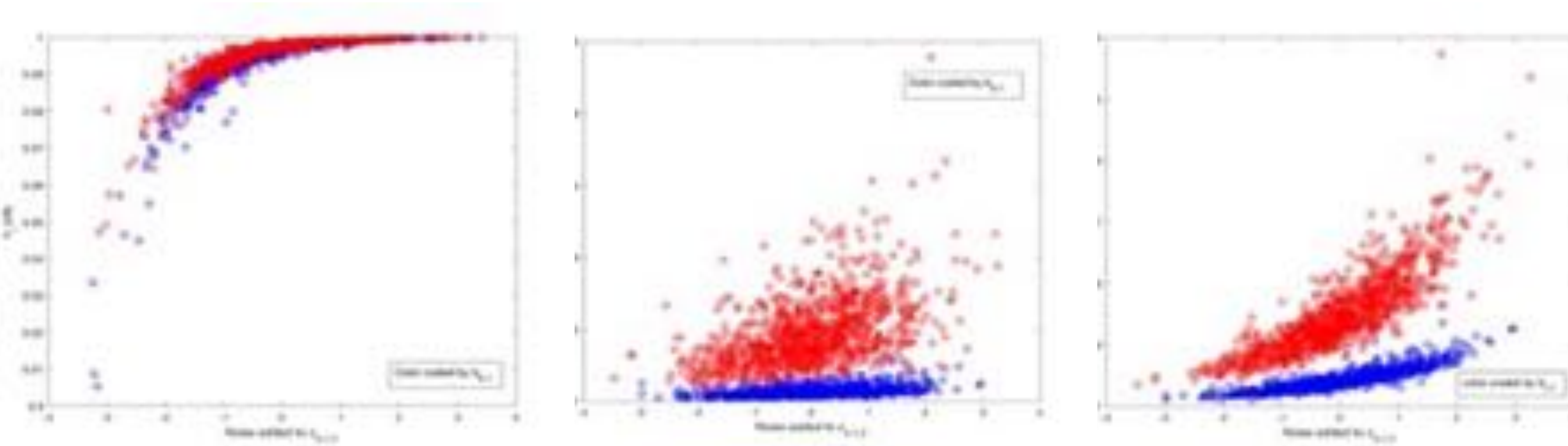
Interpretation of neural codes

- How does parity $c_k = (c_{k,1}, c_{k,2})$ depend on past bits/noise $b_{k-1}, y_{k-1}, y_{c,k-1}, \ldots, b_1, y_1, y_{c1}$
  - e.g., $E[c_{k,1} b_{k-1}] = -0.24$, $E[c_{k,1} b_{k-2}] = -0.1$, $E[c_{k,1} b_{k-3}] = -0.05$

- How encoder maps all past&current bits/feedback $\rightarrow$ parity c_k is mysterious

Interpretation of neural codes

- Neural codes require 50 diverse and complicated hidden states (in RNN)



Interpretation of neural codes

- Open problem : propose an interpretable encoder

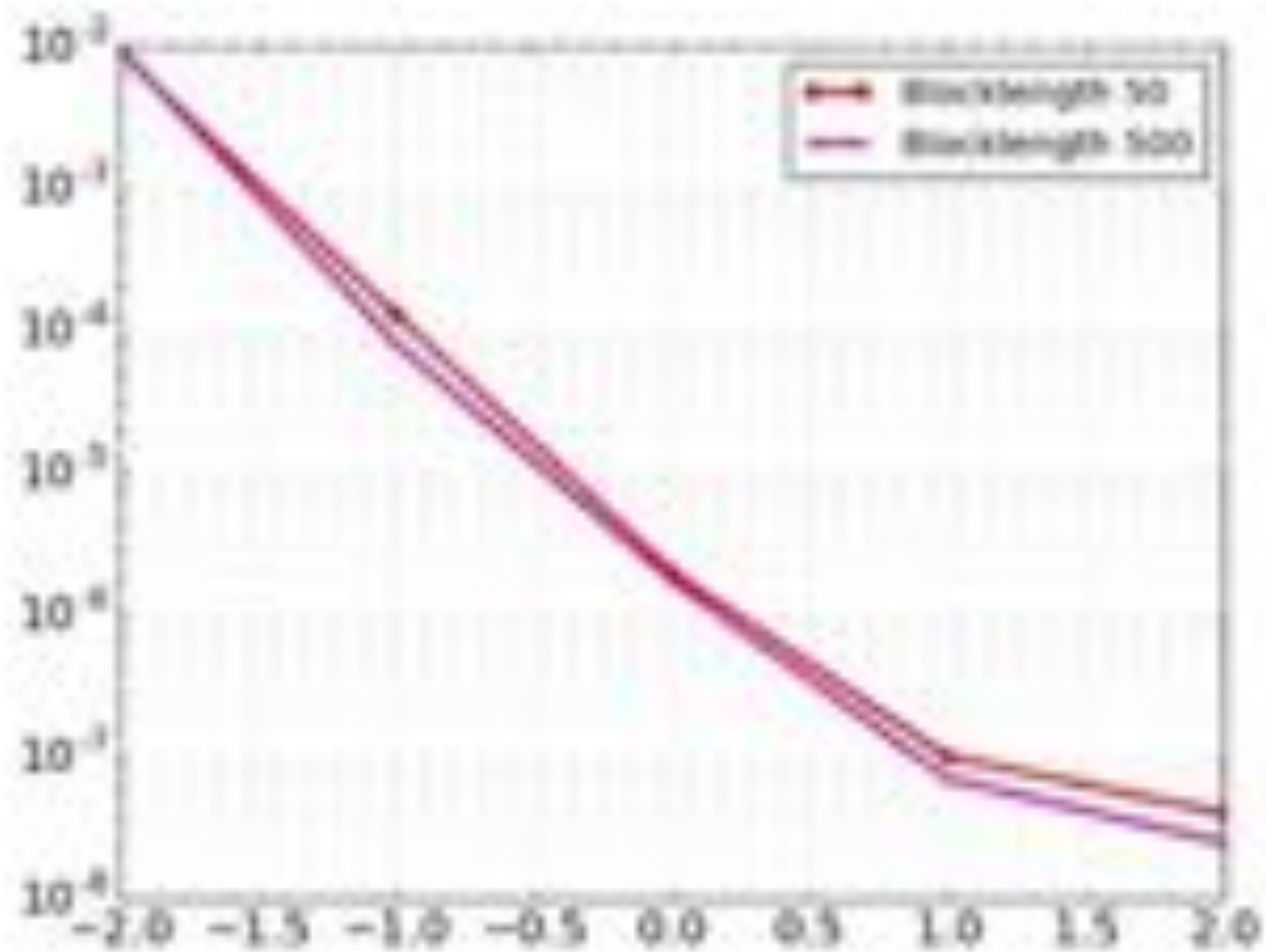
Interpretation of neural codes

- Open problem : propose an interpretable encoder
  - Train a decoder via neural network
  - Analyze the error performance

Generalization : block lengths

- BER remains the same for block lengths 50 & 500

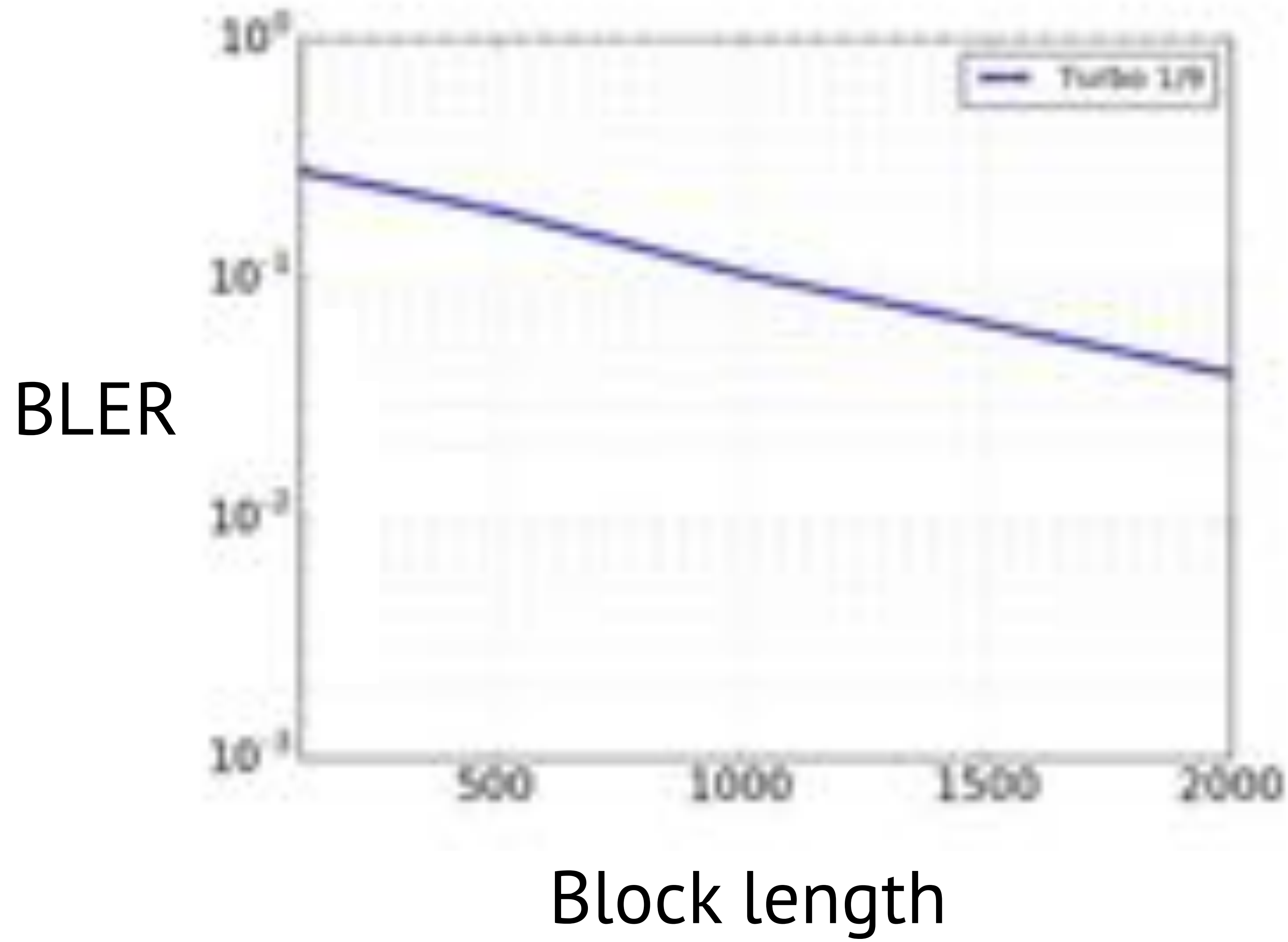
BER



SNR

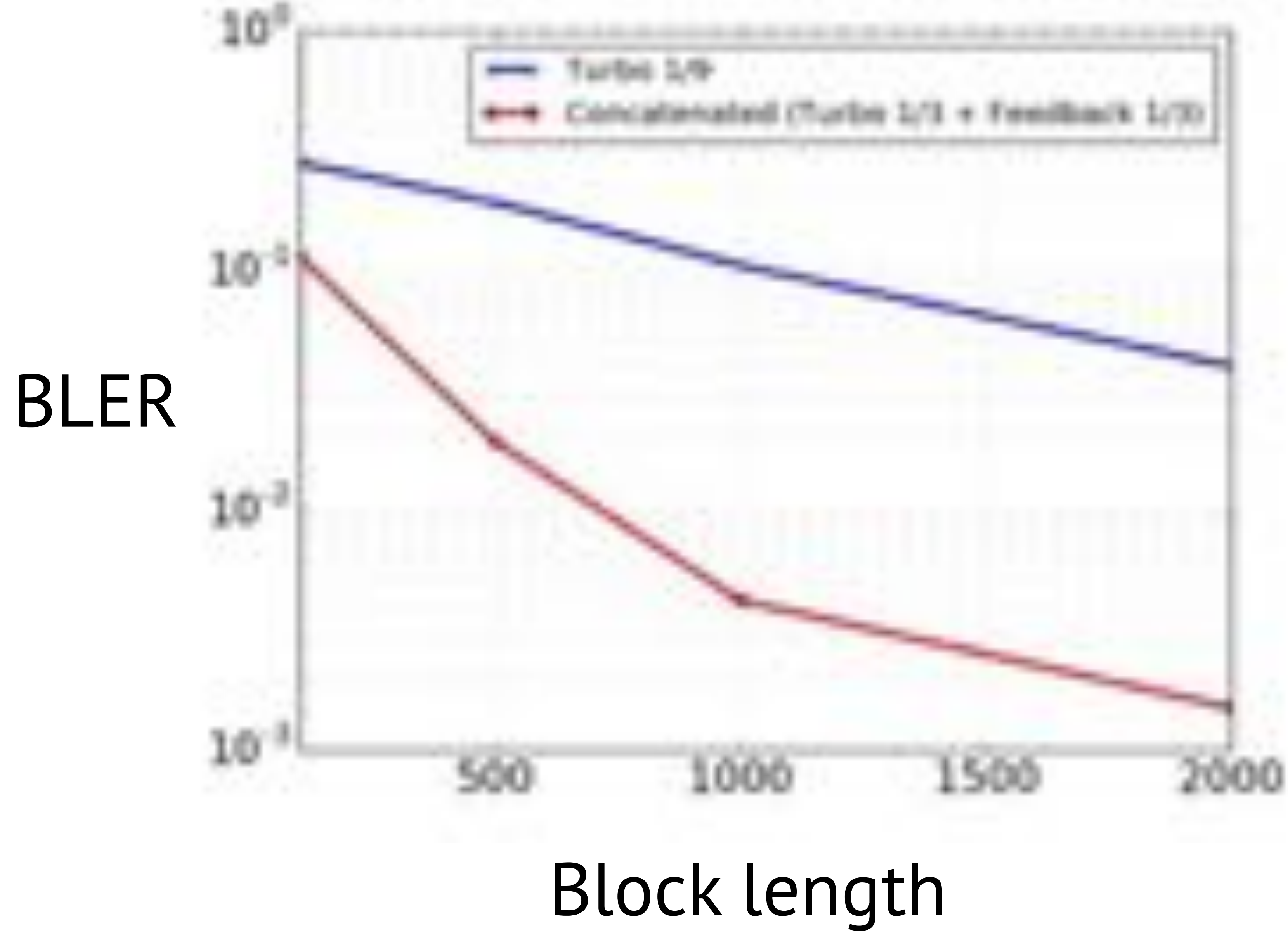
Improved error exponents

- Non-feedback scheme: BLER ↓ as block length ↑



Improved error exponents

- Concatenated code : turbo + neural feedback code
  - BLER decays faster



Open problems : Longer block lengths

- Concatenation comes with a cost, “rate”

Open problems : Longer block lengths

- Concatenation comes with a cost, “rate”
- Neural code w. long range dependency?
    - E.g. interleaver in turbo code

Open problems : Longer block lengths

- Concatenation comes with a cost, “rate”
- Neural code w. long range dependency?
  - E.g. interleaver in turbo code
  - We can put interleaver in feedback code. How to decode?

Open problems : Longer block lengths

- Concatenation comes with a cost, “rate”
- Neural code w. long range dependency?
  - E.g. interleaver in turbo code
  - We can put interleaver in feedback code. How to decode?

  Challenge: training **component dec.** for belief propagation

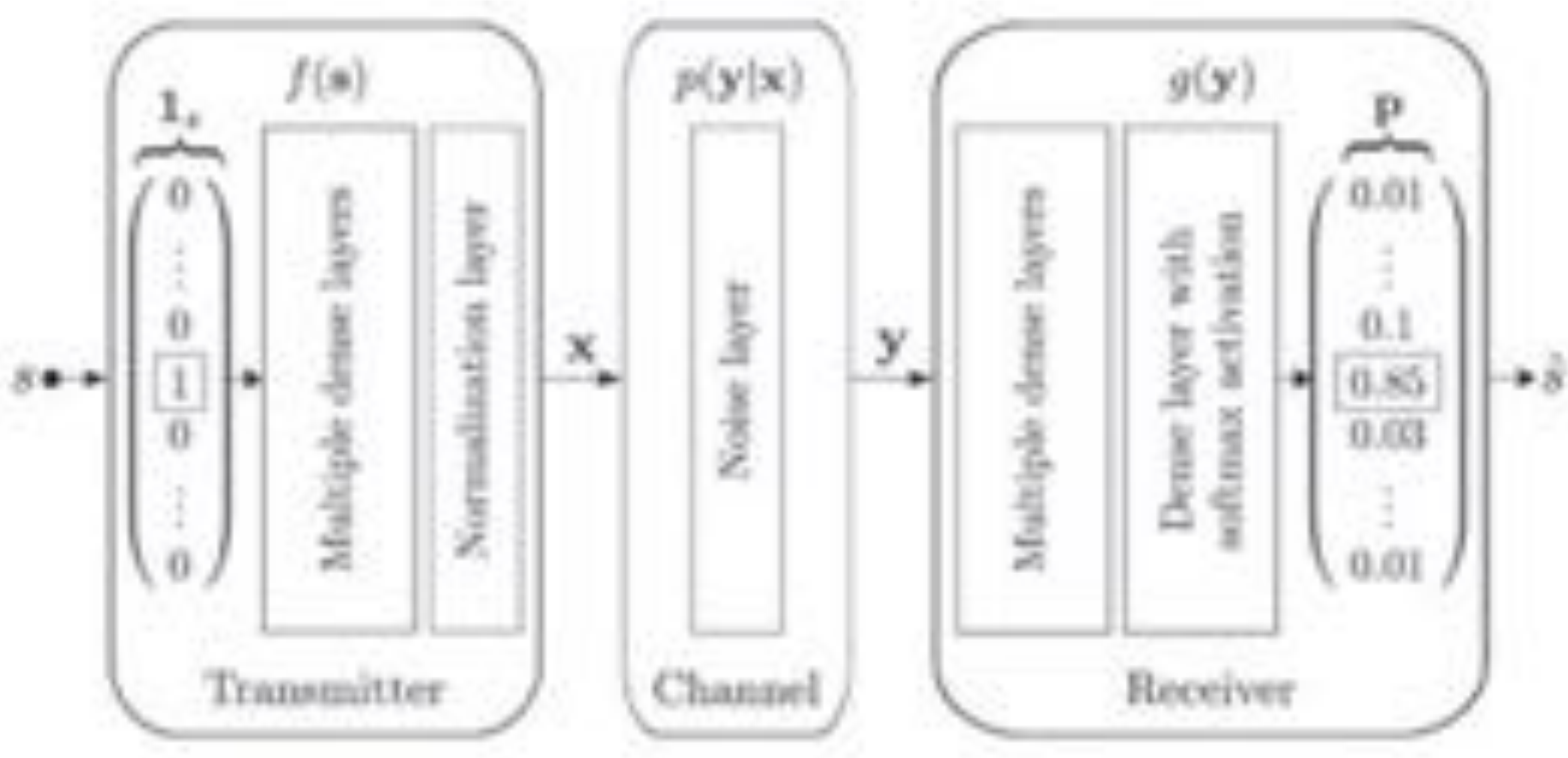
  (noisy codewords, prior likelihood) -> posterior likelihood

Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

Discovering neural codes

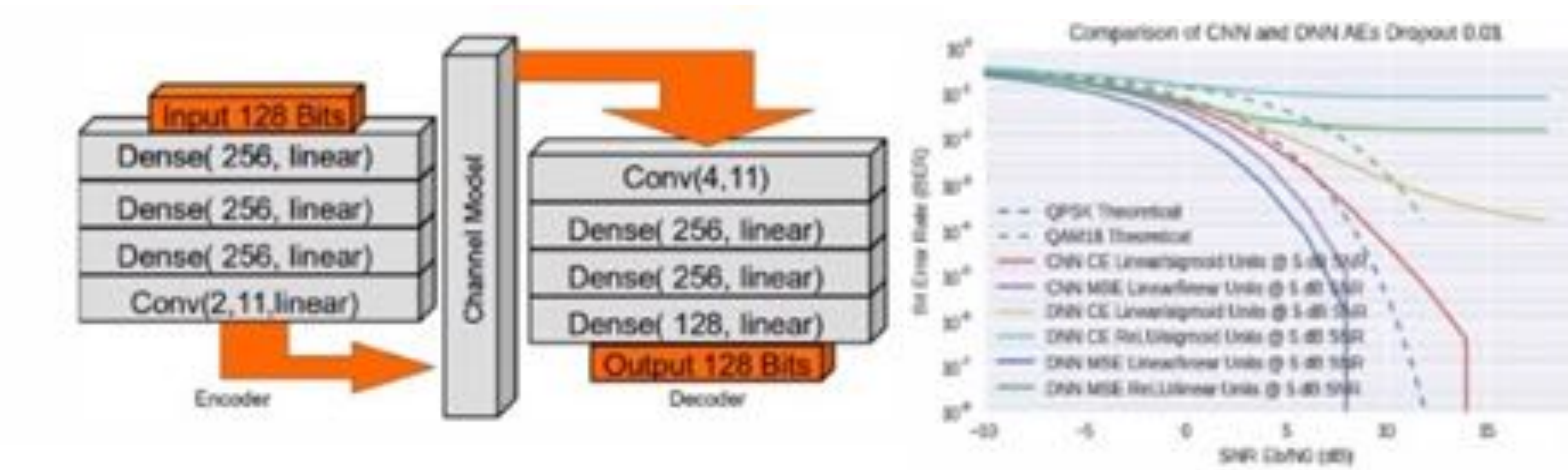
- AWGN
  - Neural (7,4) code: BER ~ BER of (7,4) Hamming code



T. O'Shea, J. Hoydis, "***An Introduction to Deep Learning for the Physical Layer***" 2017

Discovering neural codes

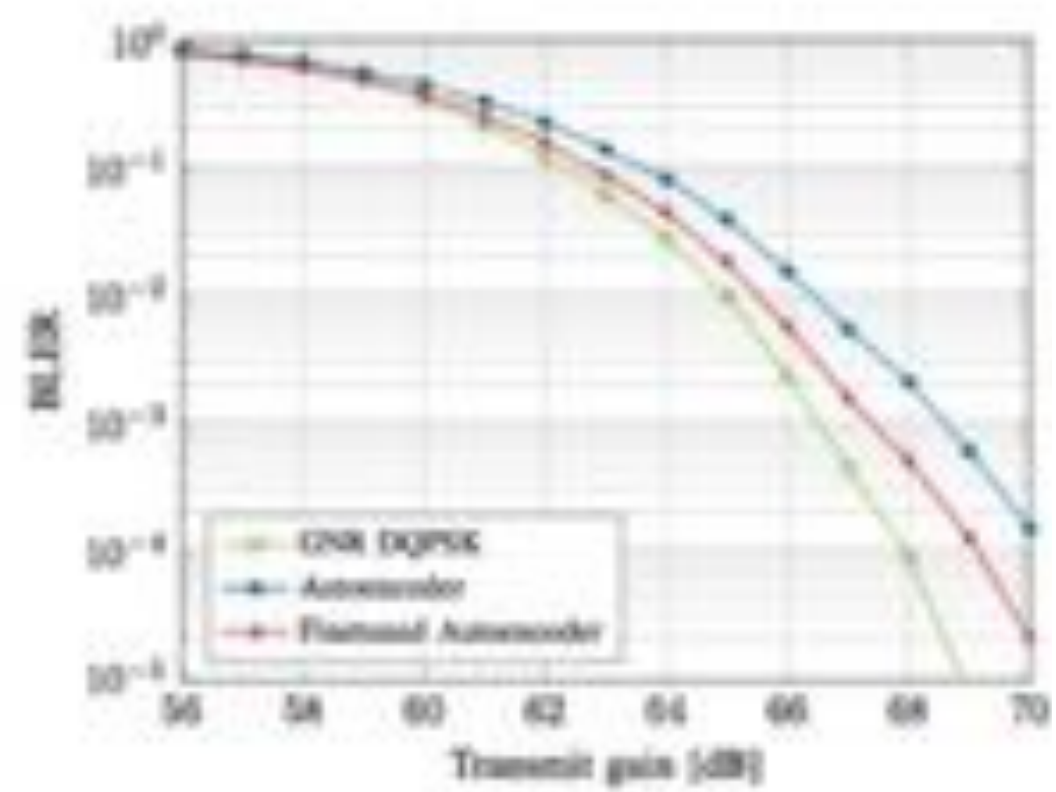
- AWGN
  - Rate 1 (128 info. bits.) BER ~ 5dB better than QPSK



T. O'Shea, K. Karra, and T. C. Clancy, "***Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention***" 2016

Discovering neural codes

- No clean model: variation of AWGN channels



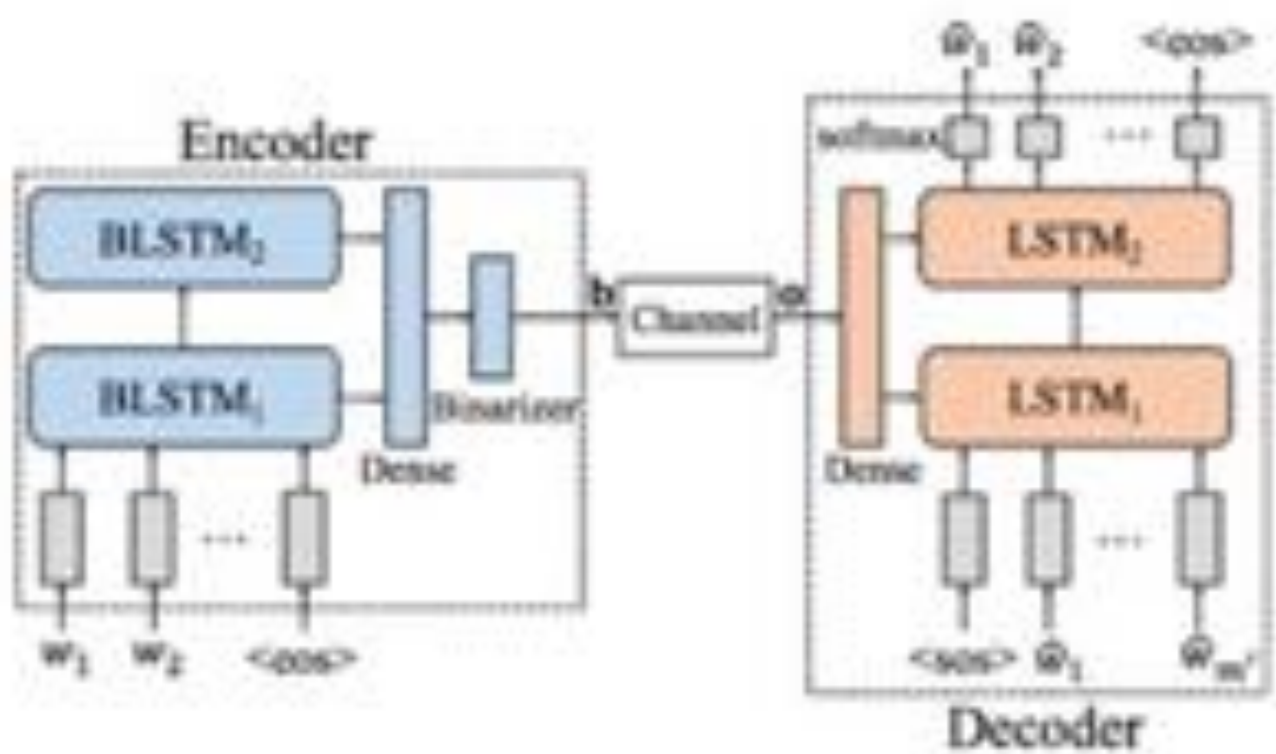
8 bits, 4 (complex) symbols
under a wireless channel

S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, “***Deep learning-based communication over the air***”, 2017

Aoudia and Jakob Hoydis, “***End-to-End Learning of Communications Systems Without a Channel Model***” 2018

Discovering neural codes

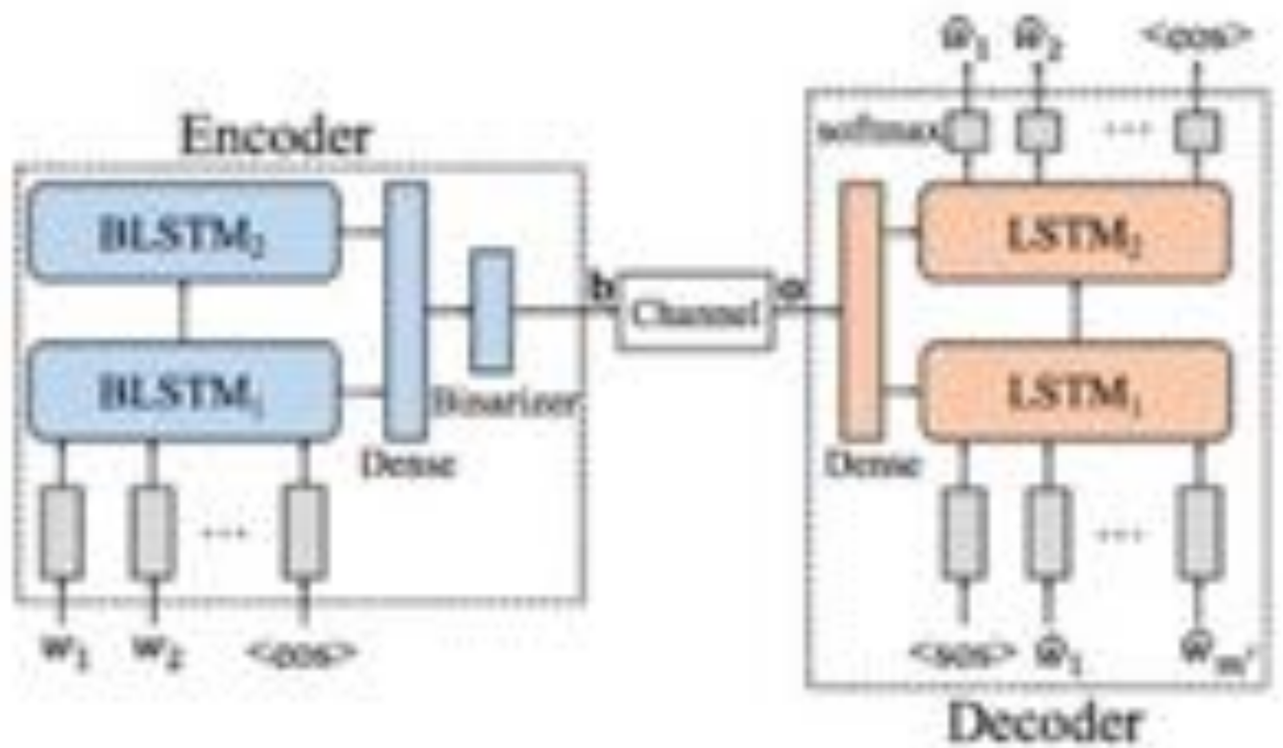
- Clean channel (erasure) / source is complicated (text)
  - Joint source channel coding



N. Farsad, M. Rao, and A. Goldsmith, "***Deep Learning for Joint Source-Channel Coding of Text***" 2018

Discovering neural codes

- Clean channel (erasure) / source is complicated (text)
  - Joint source channel coding
  - Improved reliability, evaluated by human



N. Farsad, M. Rao, and A. Goldsmith, "***Deep Learning for Joint Source-Channel Coding of Text***" 2018

Discovering neural codes

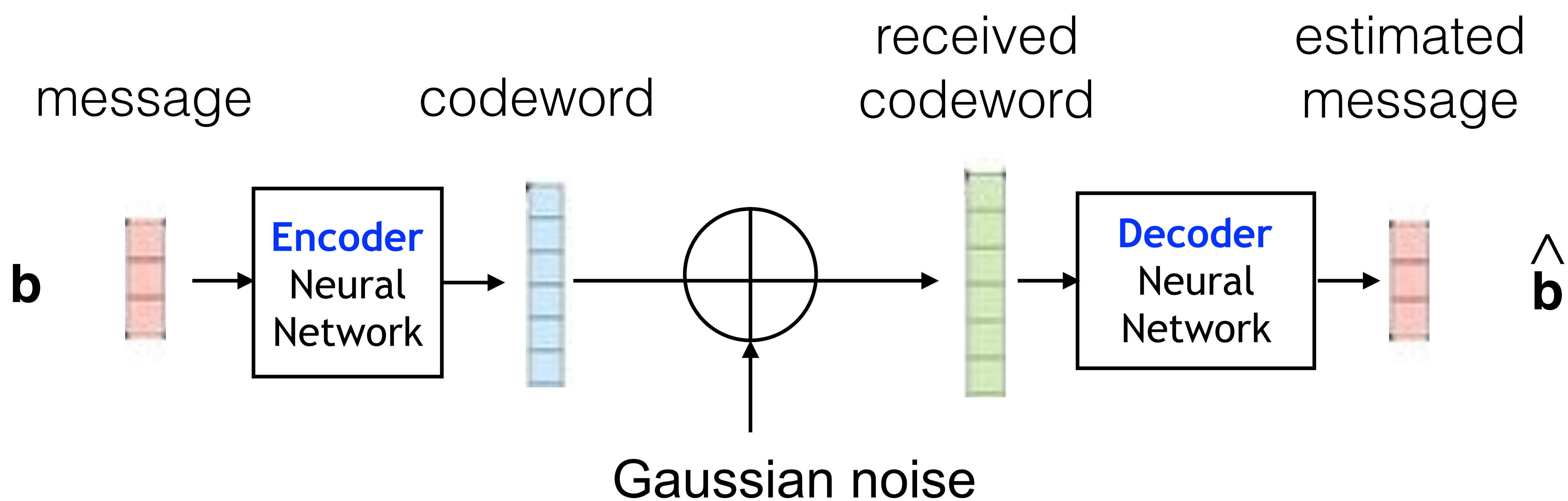
- Coded computation
  - J. Kosaian, K.V. Rashmi, and S. Venkataraman, "***Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation***", 2018
- Orthogonal frequency-division multiplexing (OFDM)
  - A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. ten Brink, "***OFDM-Autoencoder for end-to-end learning of communications systems***", 2018
  - M. Kim, W. Lee, and D. H. Cho, ***"A novel PAPR reduction scheme for OFDM system based on deep learning"***, 2018
- Multiple-Input Multiple-Output (MIMO)
  - T. J. O'Shea, T. Erpek, and T. C. Clancy, ***"Physical layer deep learning of encodings for the MIMO fading channel"***, *2017*

Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

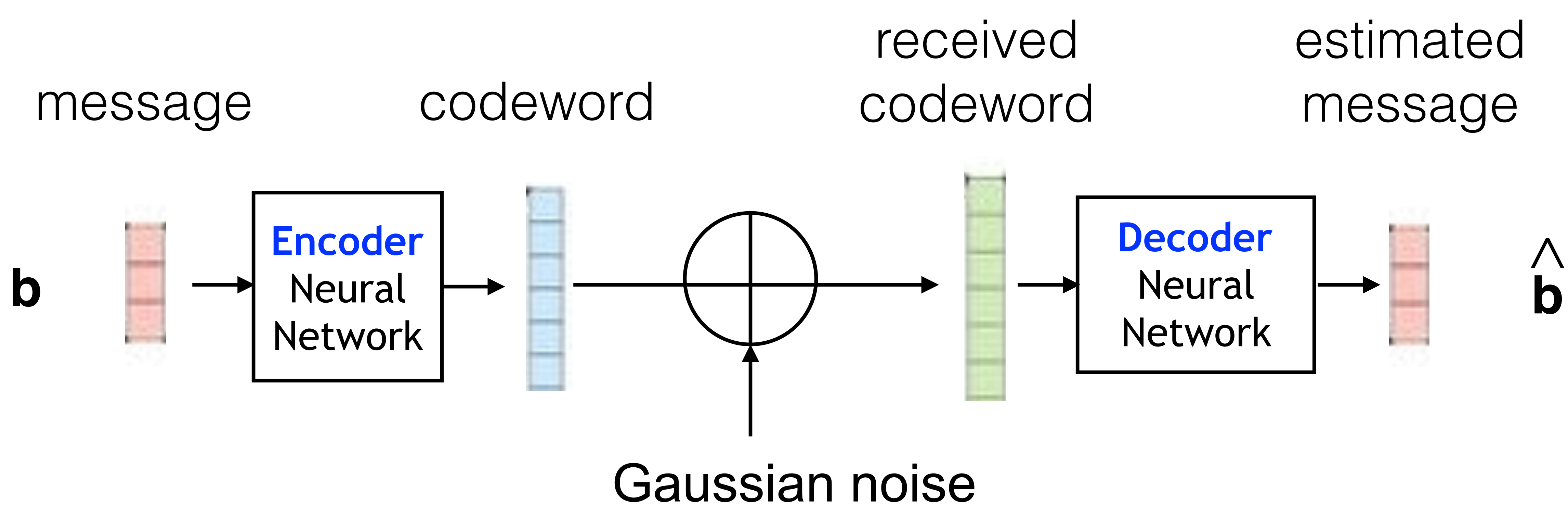
Open problems

- Canonical and benchmark : AWGN



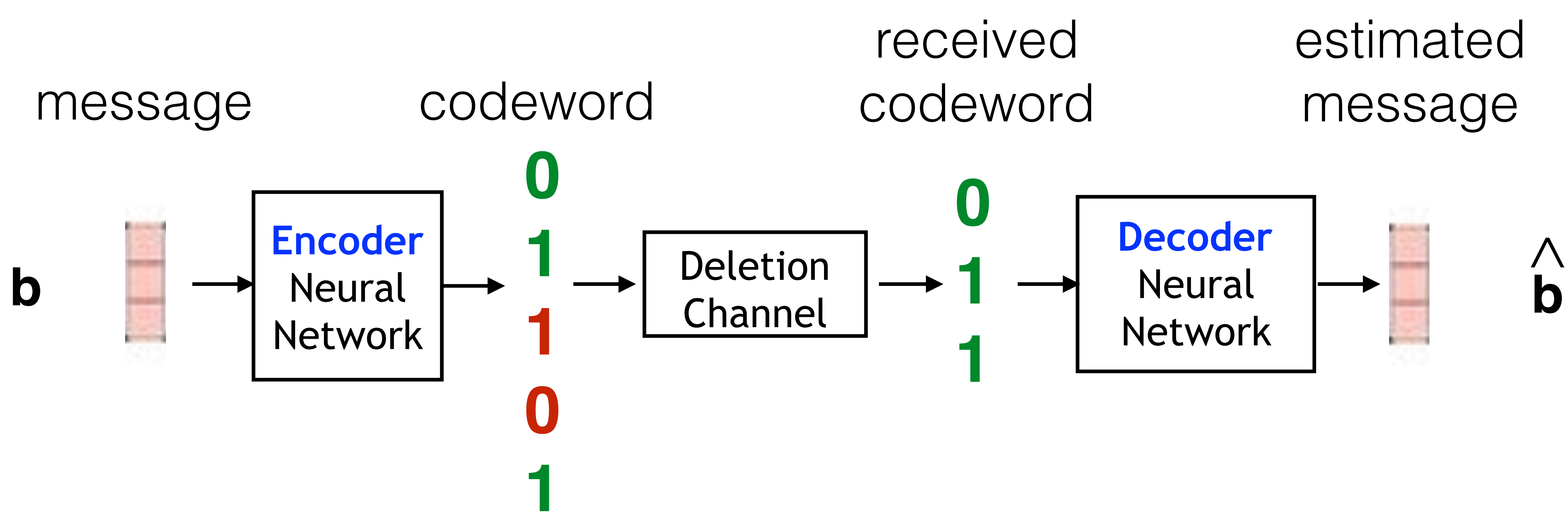
Open problems

- Canonical and benchmark : AWGN
  - Challenge 1. neural code that has a long range memory
  - Challenge 2. jointly training Enc./Dec.



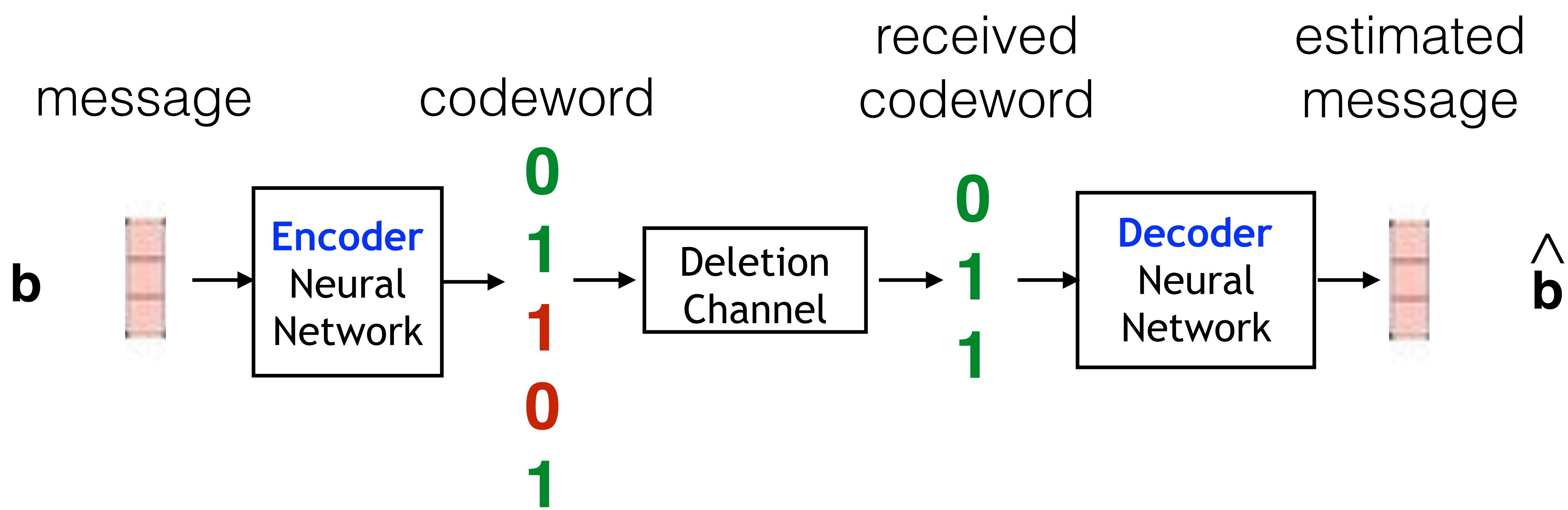
Open problems

- Channels with no good codes: deletion channel
  - Practical (e.g. lack of synchronization, DNA sequencing)



Open problems

- Channels with no good codes: deletion channel
  - Practical (e.g. lack of synchronization, DNA sequencing)
  - Optimal codes known only if deletion probability v. small
  - No practical code exists; capacity unknown in general



Open problems

- Channels with no good codes: deletion channel
  - Practical (e.g. lack of synchronization, DNA sequencing)
  - Optimal codes known only if deletion probability v. small
  - No practical code exists; capacity unknown in general
- Many network settings
  - Relay, interference, Coordinated Multipoint (CoMP)

Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

Open problem 2

Learning a decoder

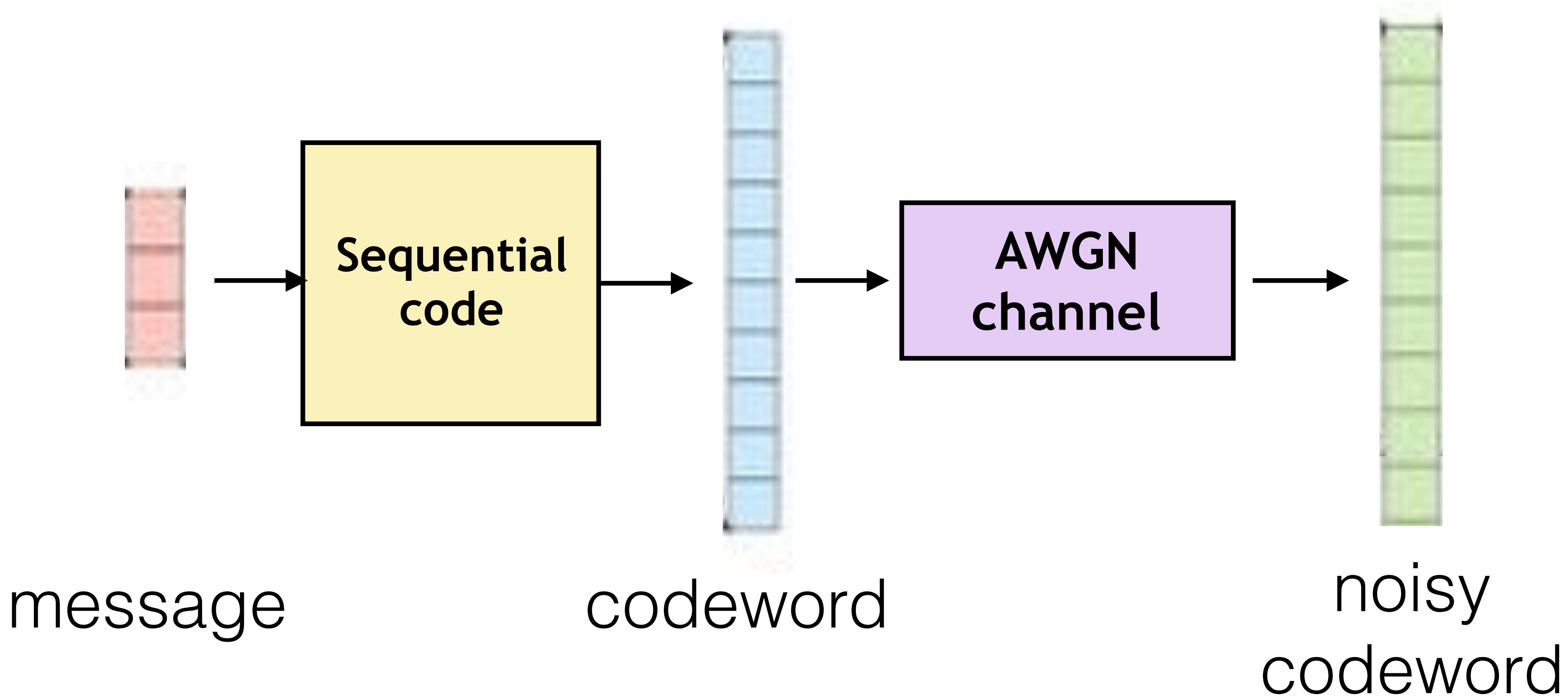
under practical channels

H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, P. Viswanath, "***Communication algorithms via deep learning***" 2018

Sequential codes

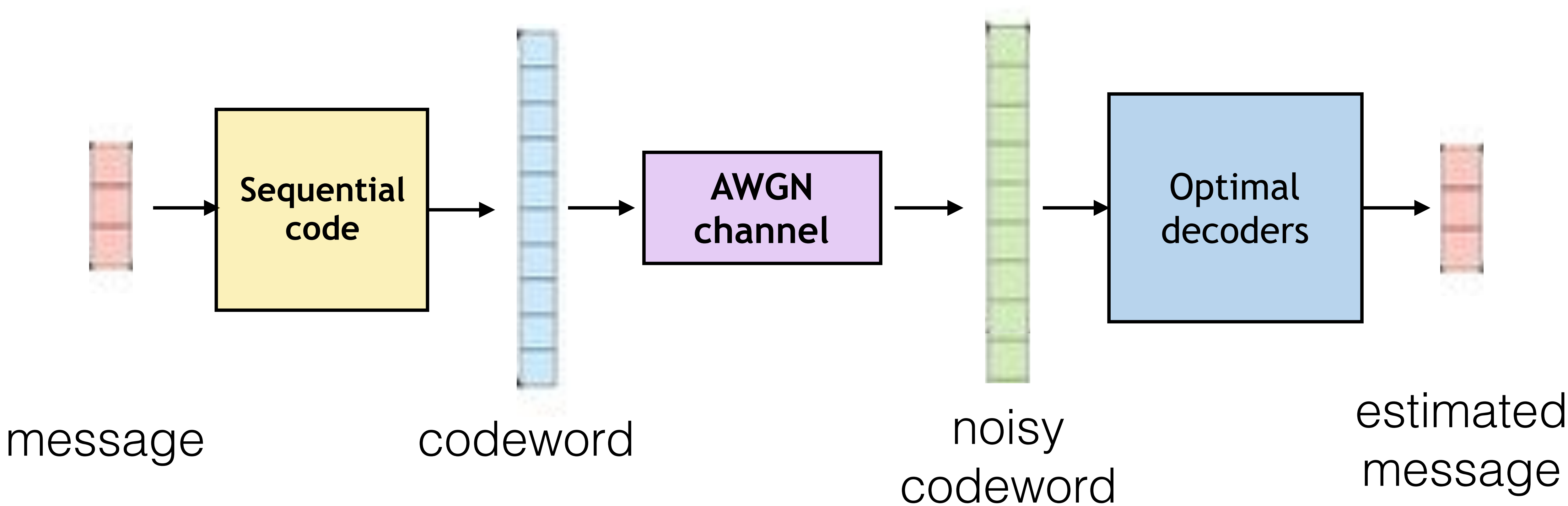
- Convolutional codes, turbo codes
- Practical
  - 3G/4G mobile communications (e.g., in UMTS and LTE)
  - (Deep space) satellite communications
- Achieve performance close to fundamental limit
- Have a natural recurrent structure aligned with RNN

Sequential codes under AWGN

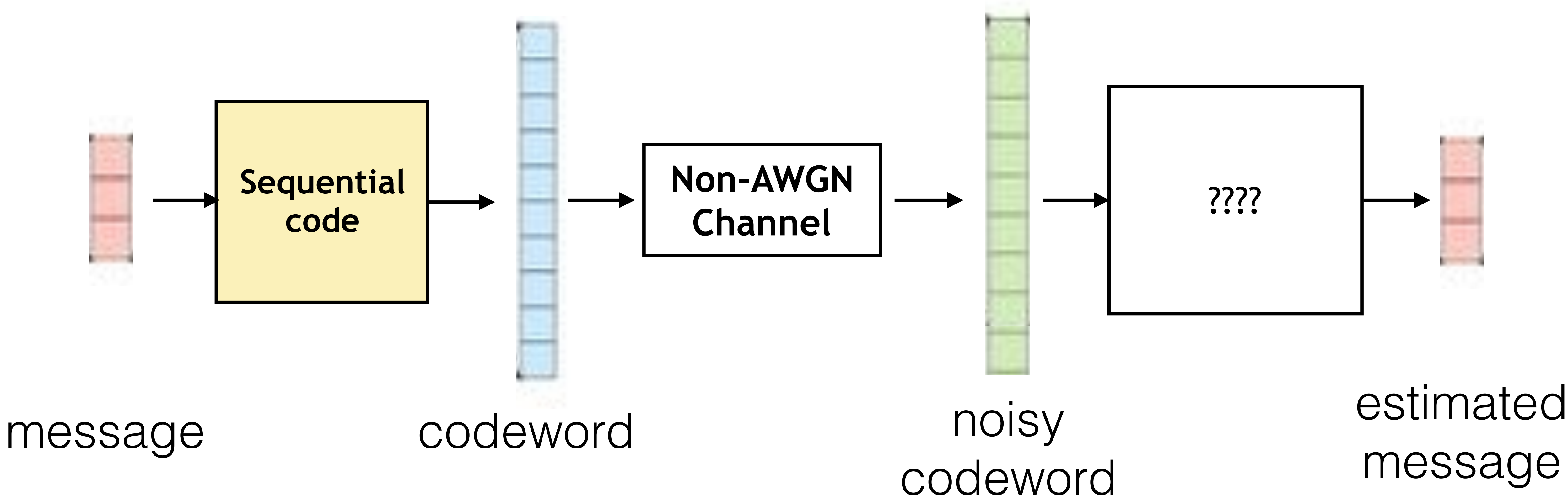


Sequential codes under AWGN

- Optimal decoders under AWGN
  - e.g. Viterbi, BCJR decoder for convolutional codes



Non-AWGN channel



Bursty noise

- High-power noise is added occasionally

Bursty noise

- High-power noise is added occasionally

- Heuristic decoders are used

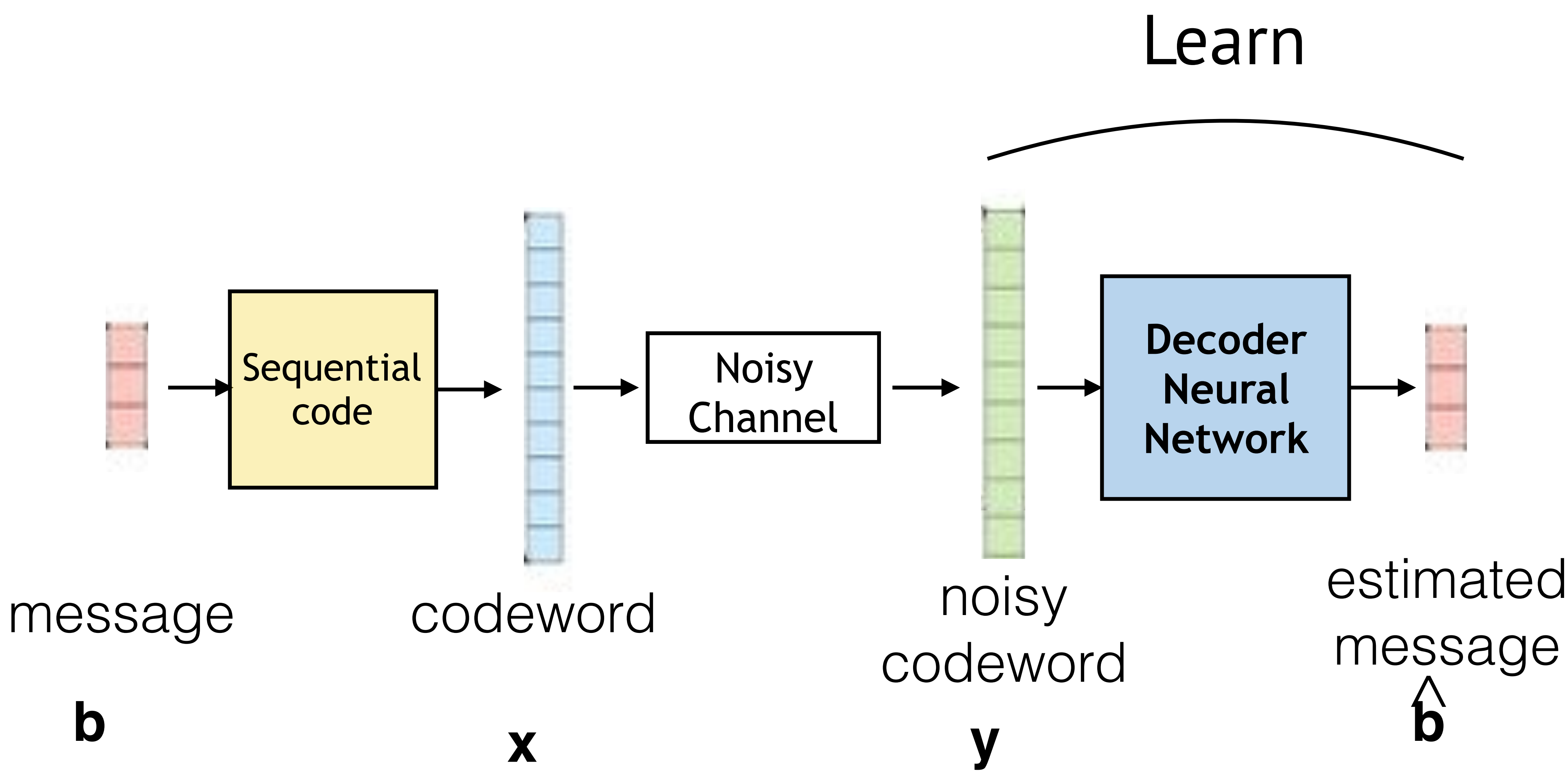
Bursty noise

- High-power noise is added occasionally

- Heuristic decoders are used
- Train a neural network to decode

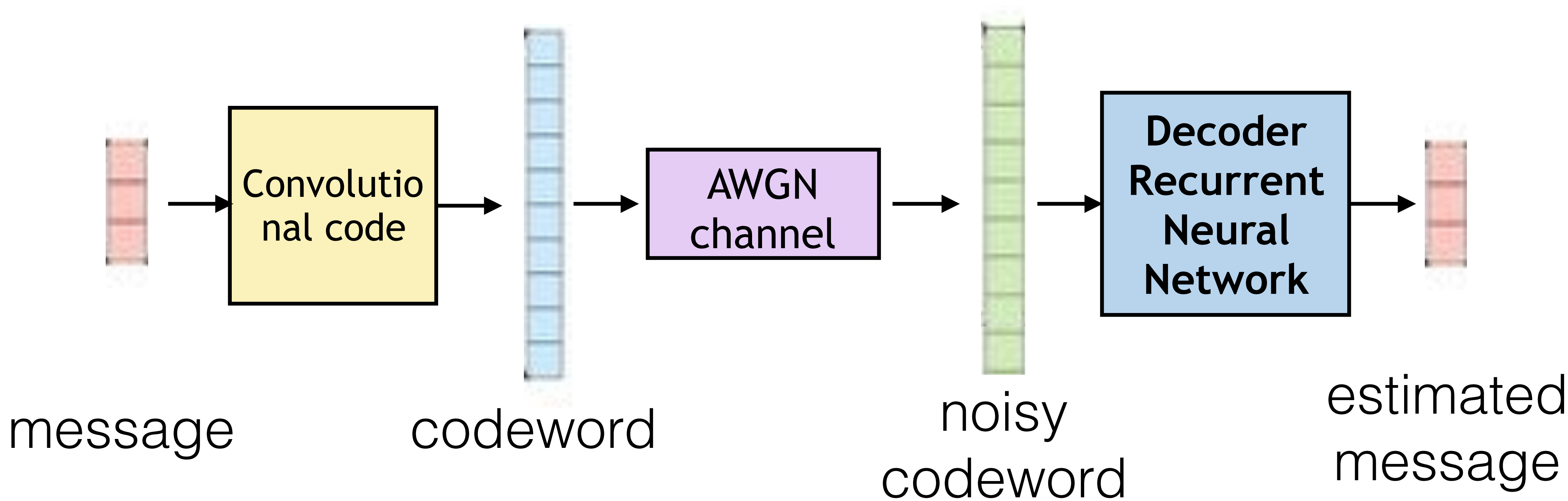
Neural decoder

- Supervised training with (noisy codeword **y**, message **b**)



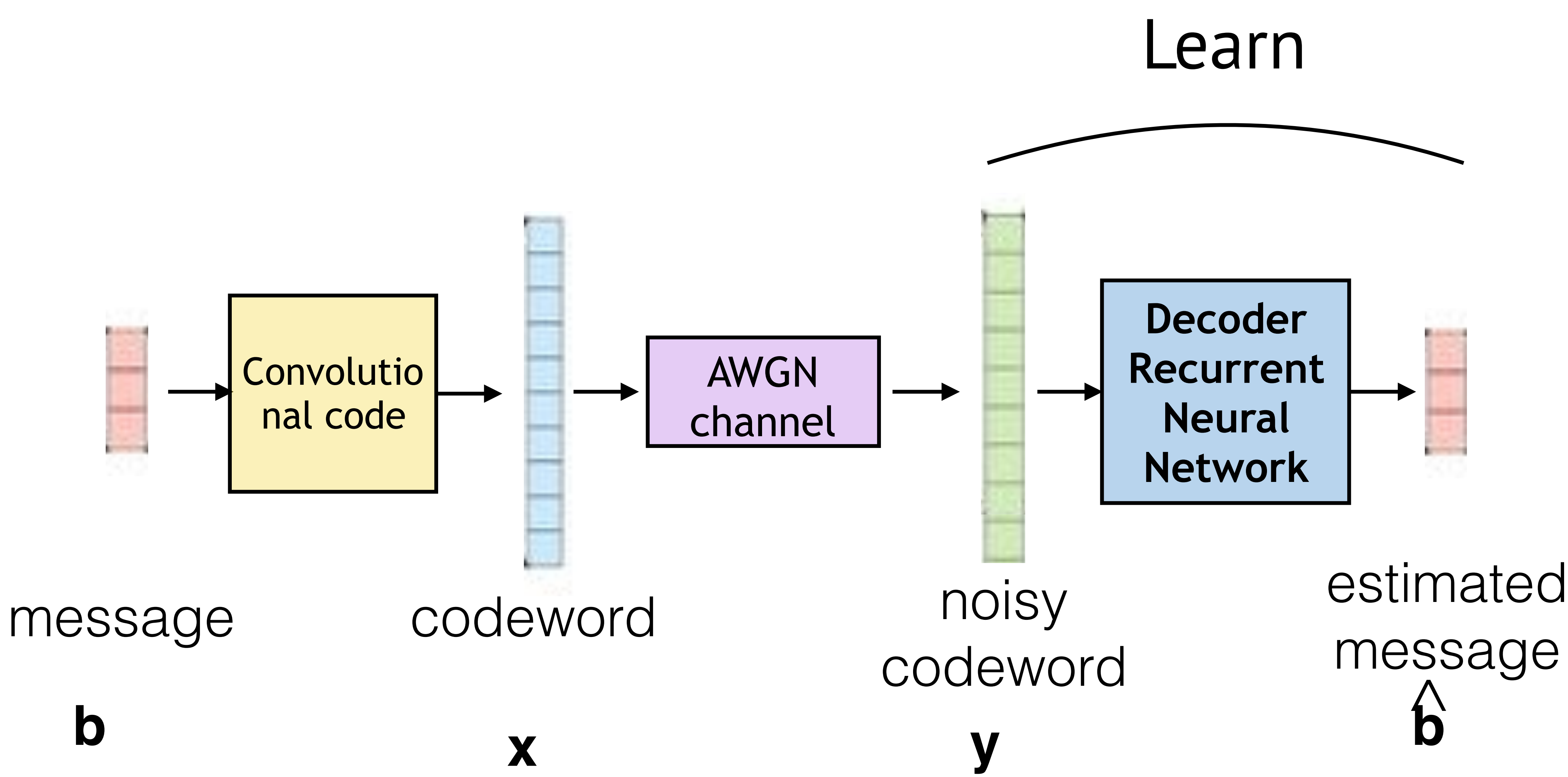
Neural decoder under AWGN

- Convolutional codes
- Model decoder as a Recurrent Neural Network (RNN)



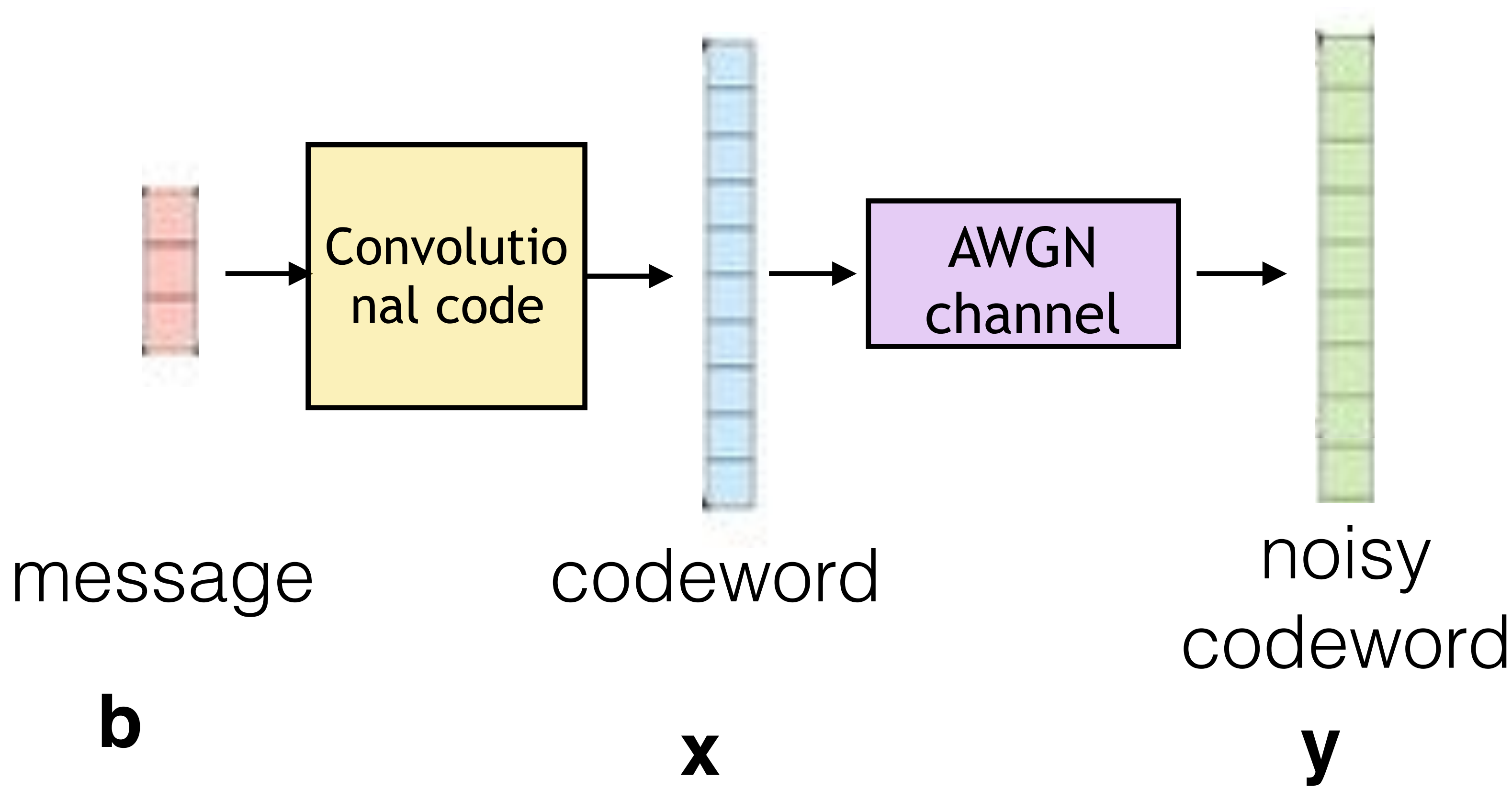
Training

- Supervised training with (noisy codeword **y**, message **b**)
- Loss $\mathrm{E}[(\mathbf{b}-\hat{\mathbf{b}})^2]$



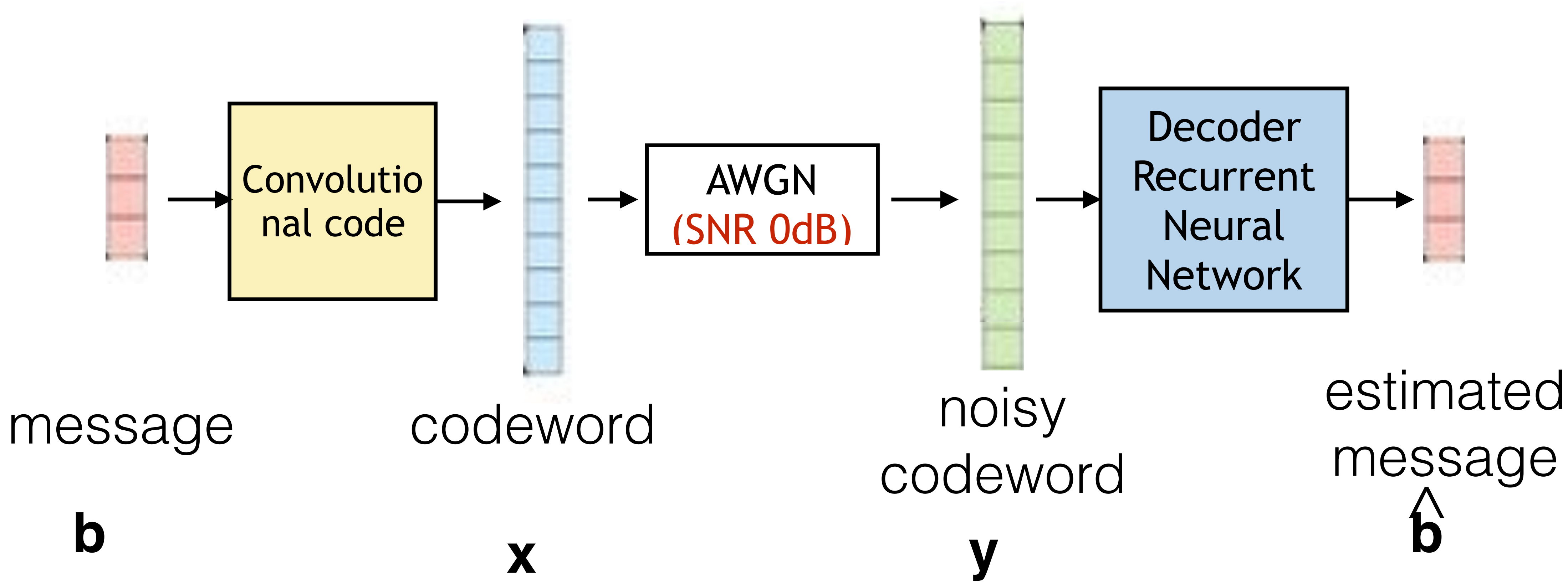
Choice of training examples

- Training examples (**y**, **b**) :
  - Length of message bits **b** = (b_1, …, b_K)
  - SNR of the noisy codeword **y**



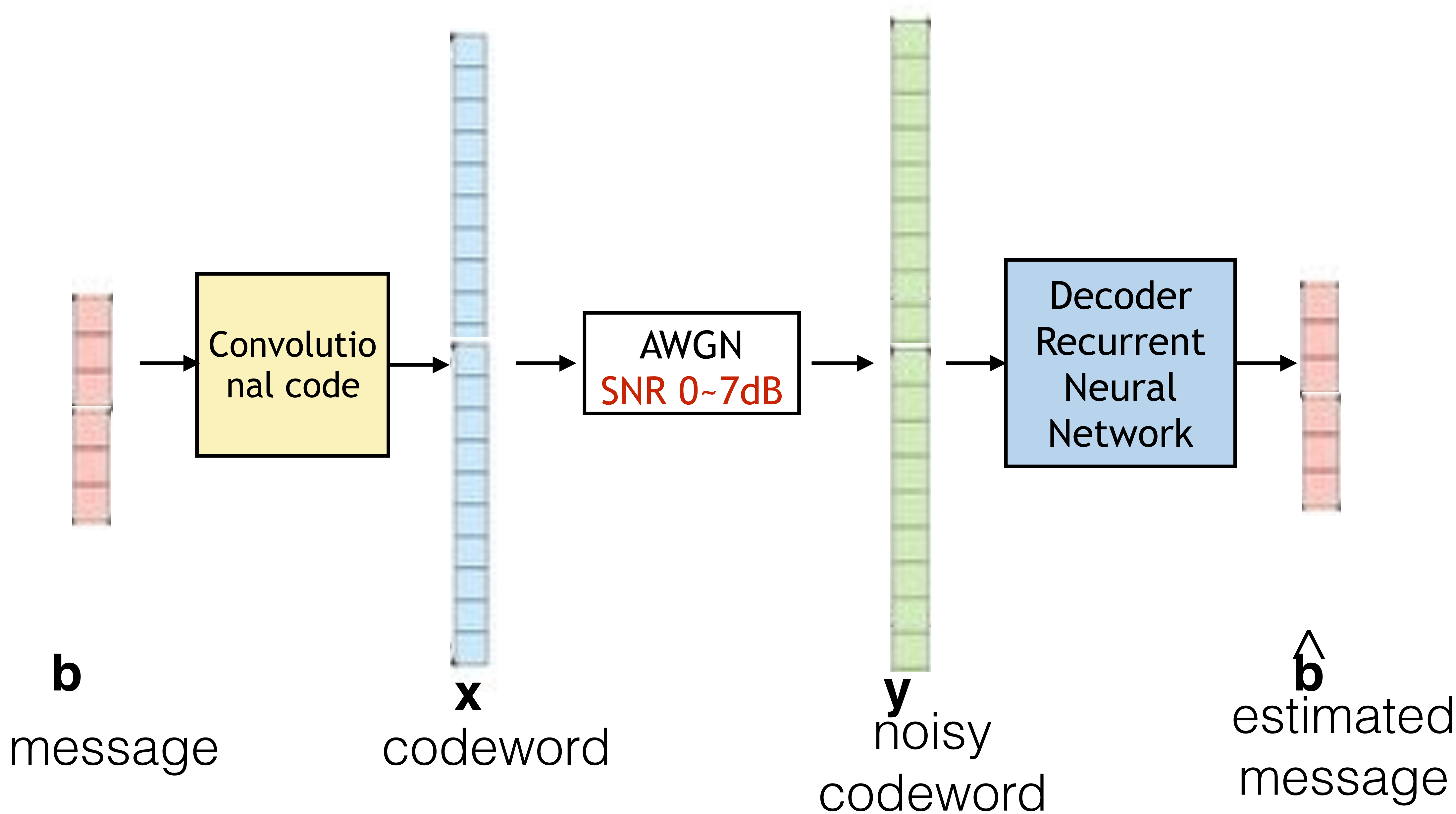
Choice of training examples

- Train at a block length 100, fixed SNR (0dB)



Choice of training examples

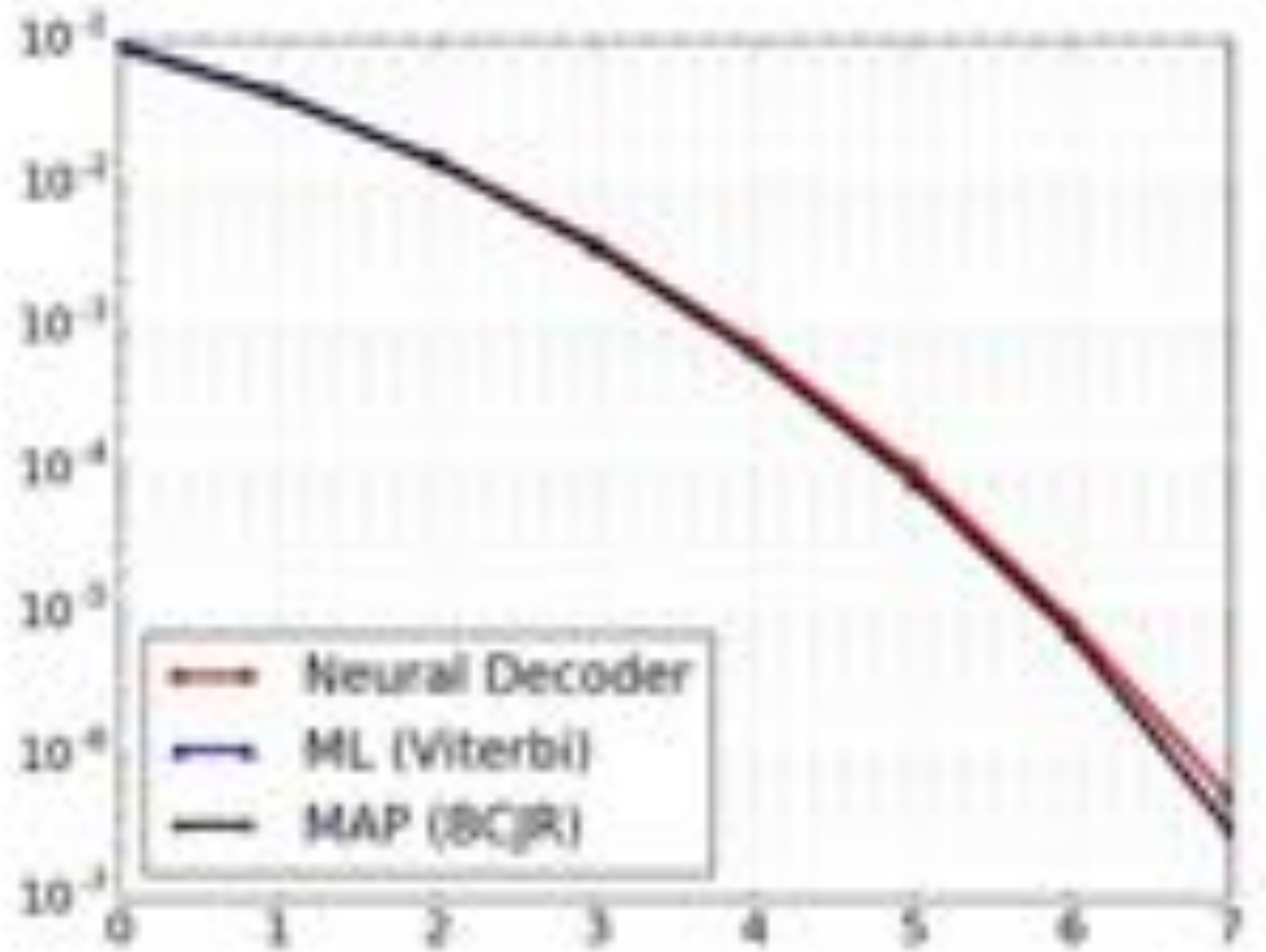
- Train at a block length 100, fixed SNR (0dB)
- Optimal performance for every block lengths, across SNR



Results

- Neural decoder learns decoding convolutional codes

BER

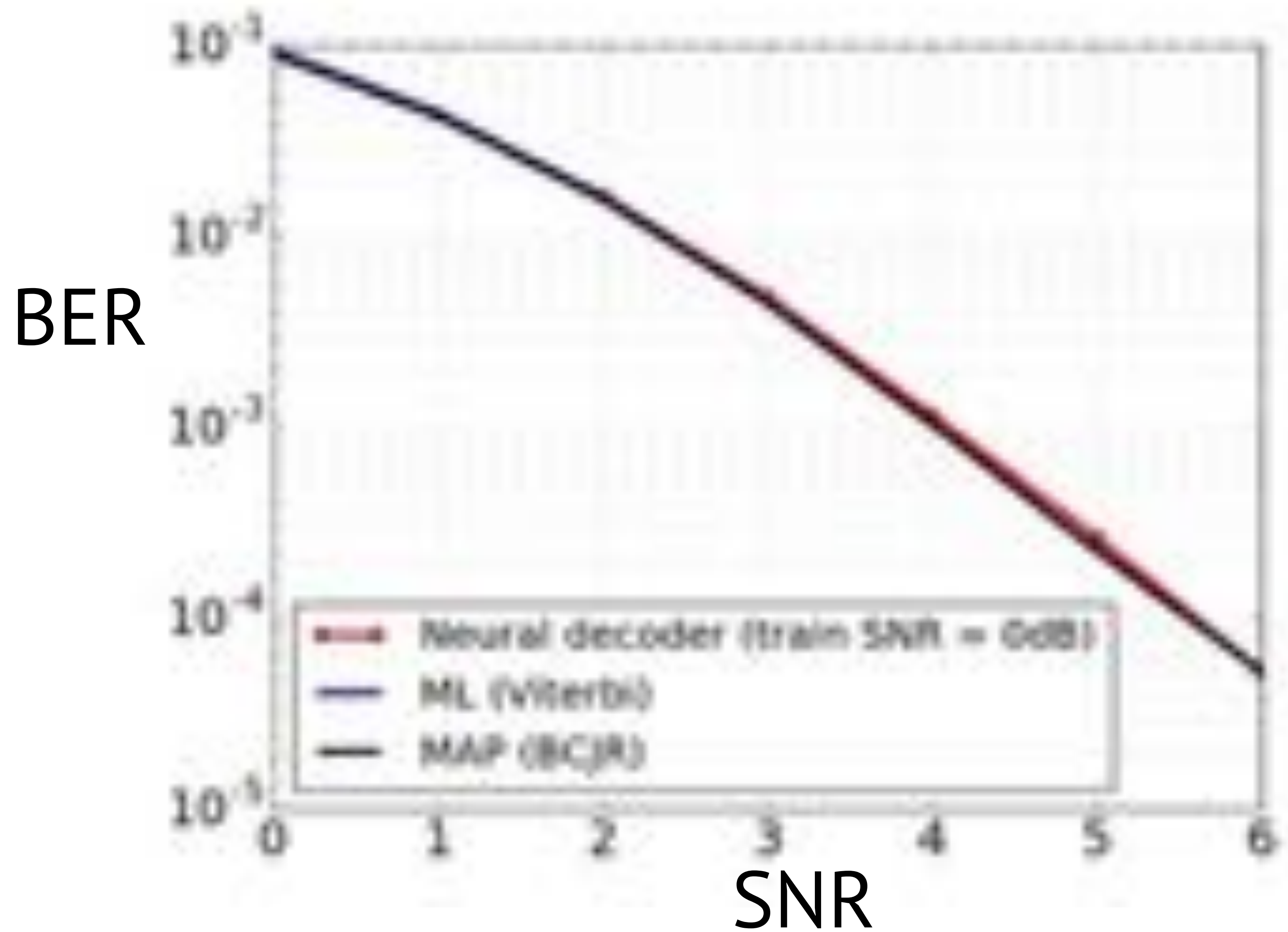


SNR

Train: block length = 100, SNR=0dB   Test: block length = 10K

Results

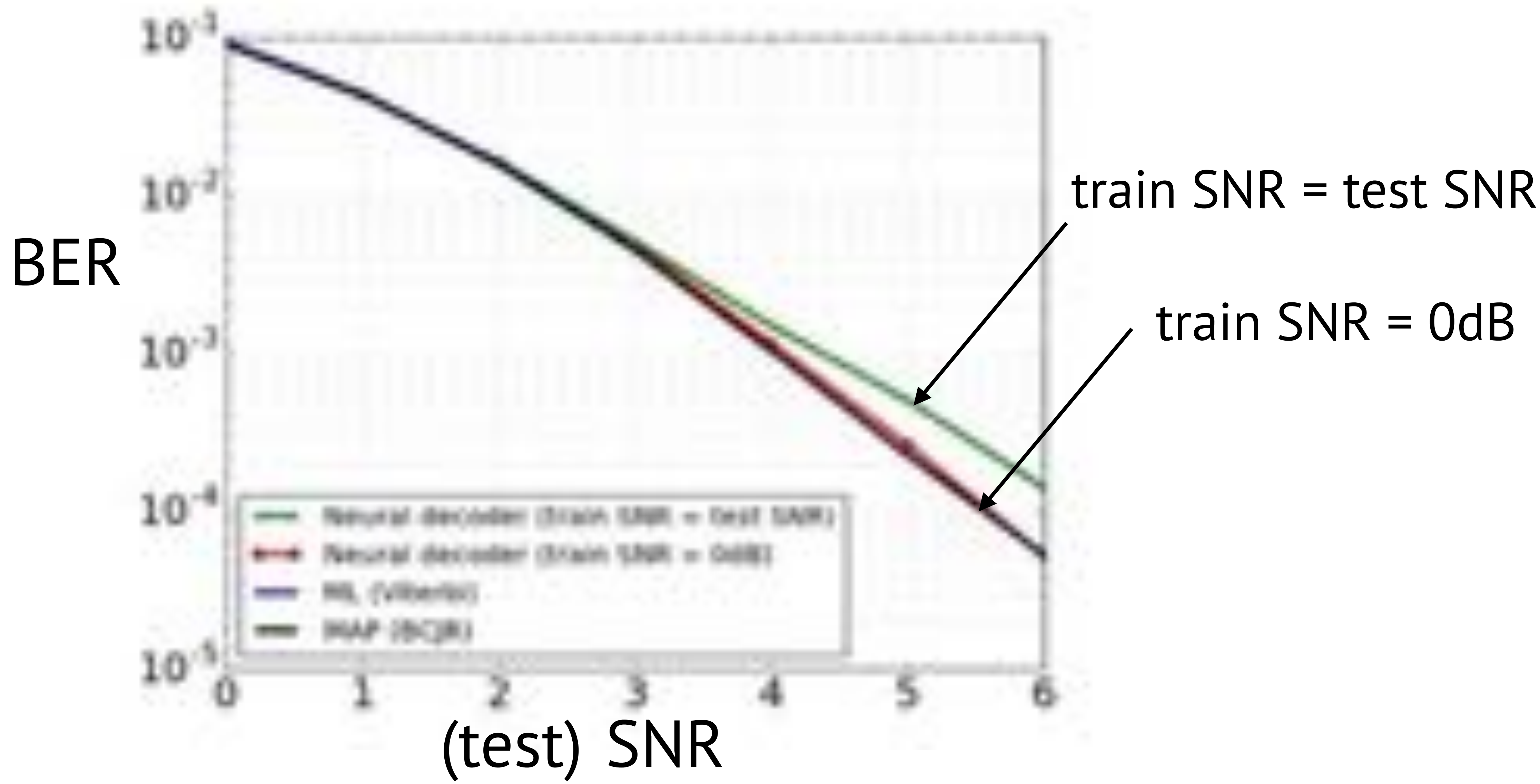
- Neural decoder learns decoding convolutional codes



Train: block length = 100, SNR=0dB   Test: block length = 100

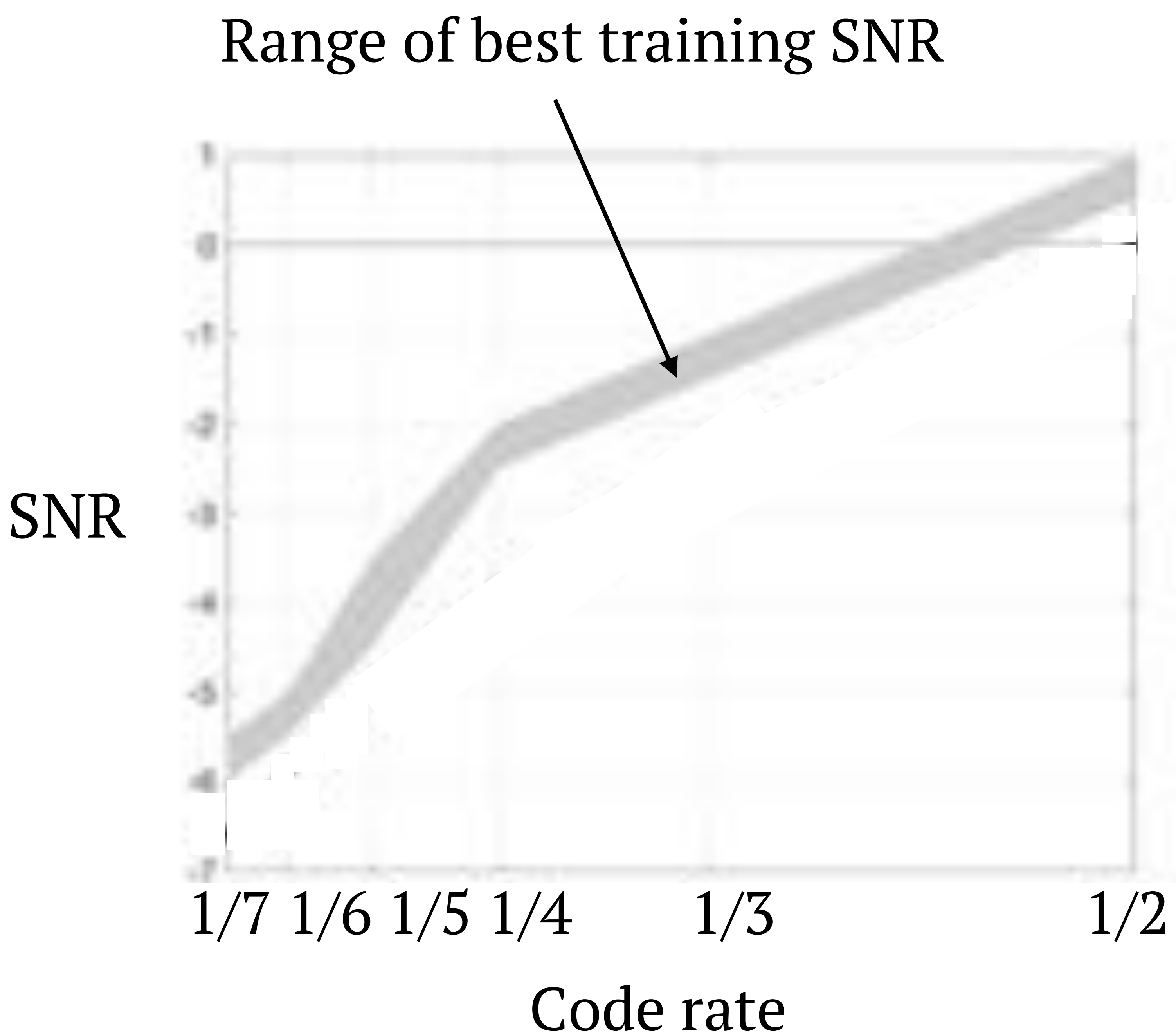
Choice of training examples

- Training with noisy codewords at test SNR?



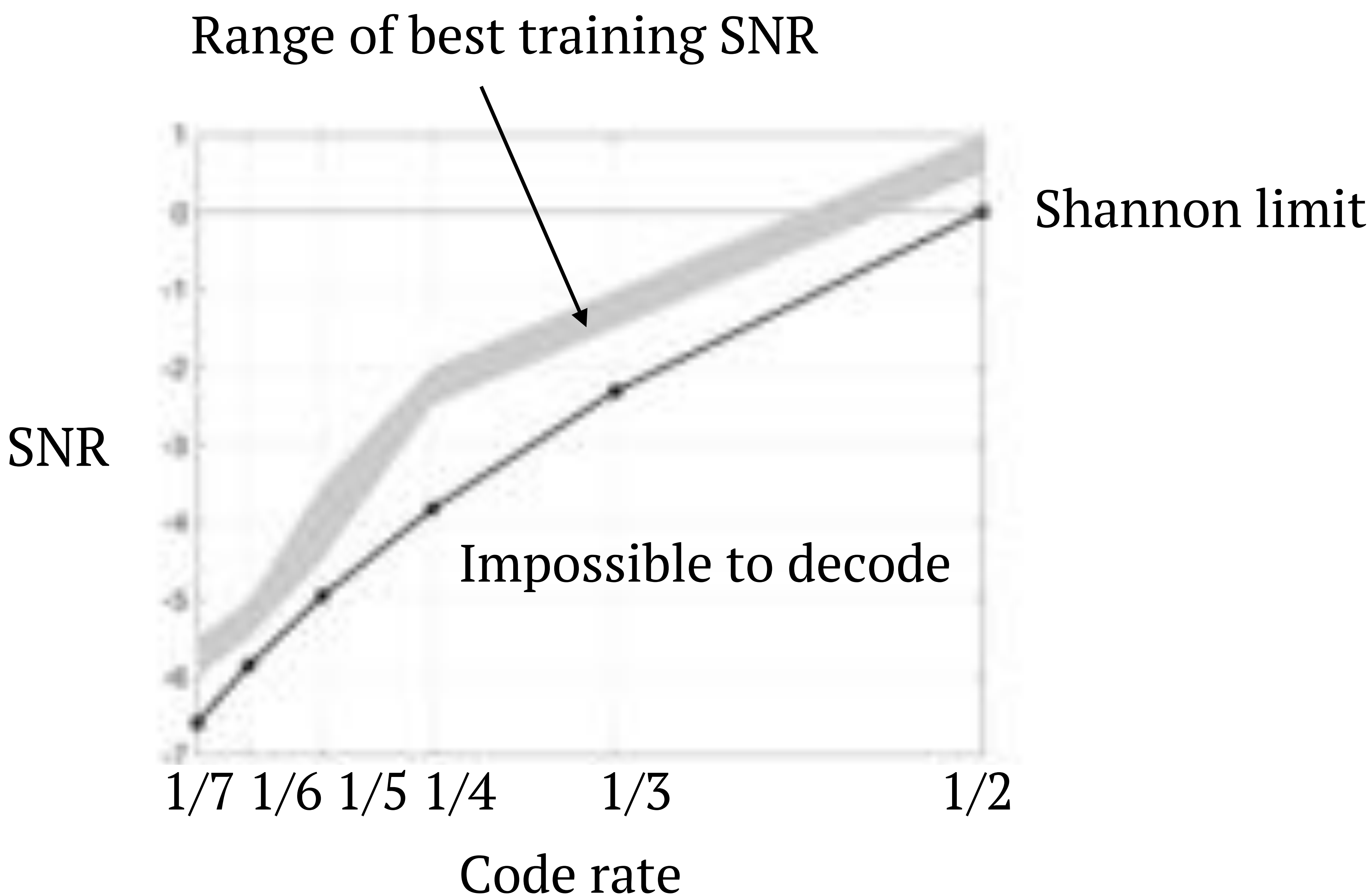
Choice of training examples

- Empirically find best training SNR for different code rates



Choice of training examples

- Hardest training examples



Adversarial training

- Idea of hardest training examples
  - Training with noisy examples
  - Applied to problems s.t. training examples can be chosen

Decoding turbo codes under AWGN

- Decoding of turbo codes:

  belief propagation of **BCJR component decoder**s

  (noisy codeword, prior likelihood) —> posterior likelihood

Decoding turbo codes under AWGN

- Decoding of turbo codes:

  belief propagation of **BCJR component decoder**s

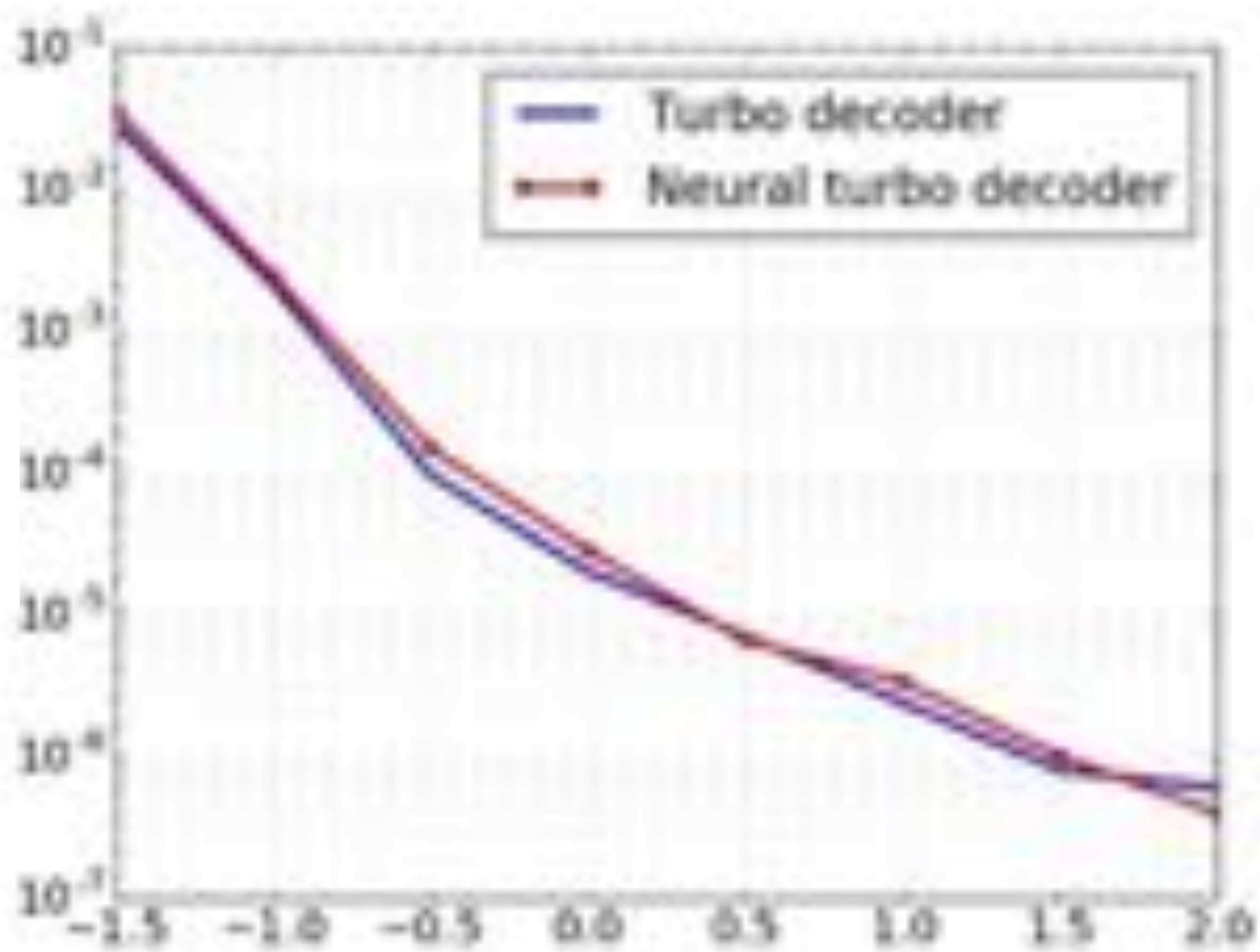
  (noisy codeword, prior likelihood) —> posterior likelihood

- Learning neural turbo decoder:
  - Train a **neural component decoder with BCJR labels**
  - Stack component decoders and train the BP decoder

Decoding turbo codes under AWGN

- Neural decoder performance ~ turbo codes

BER

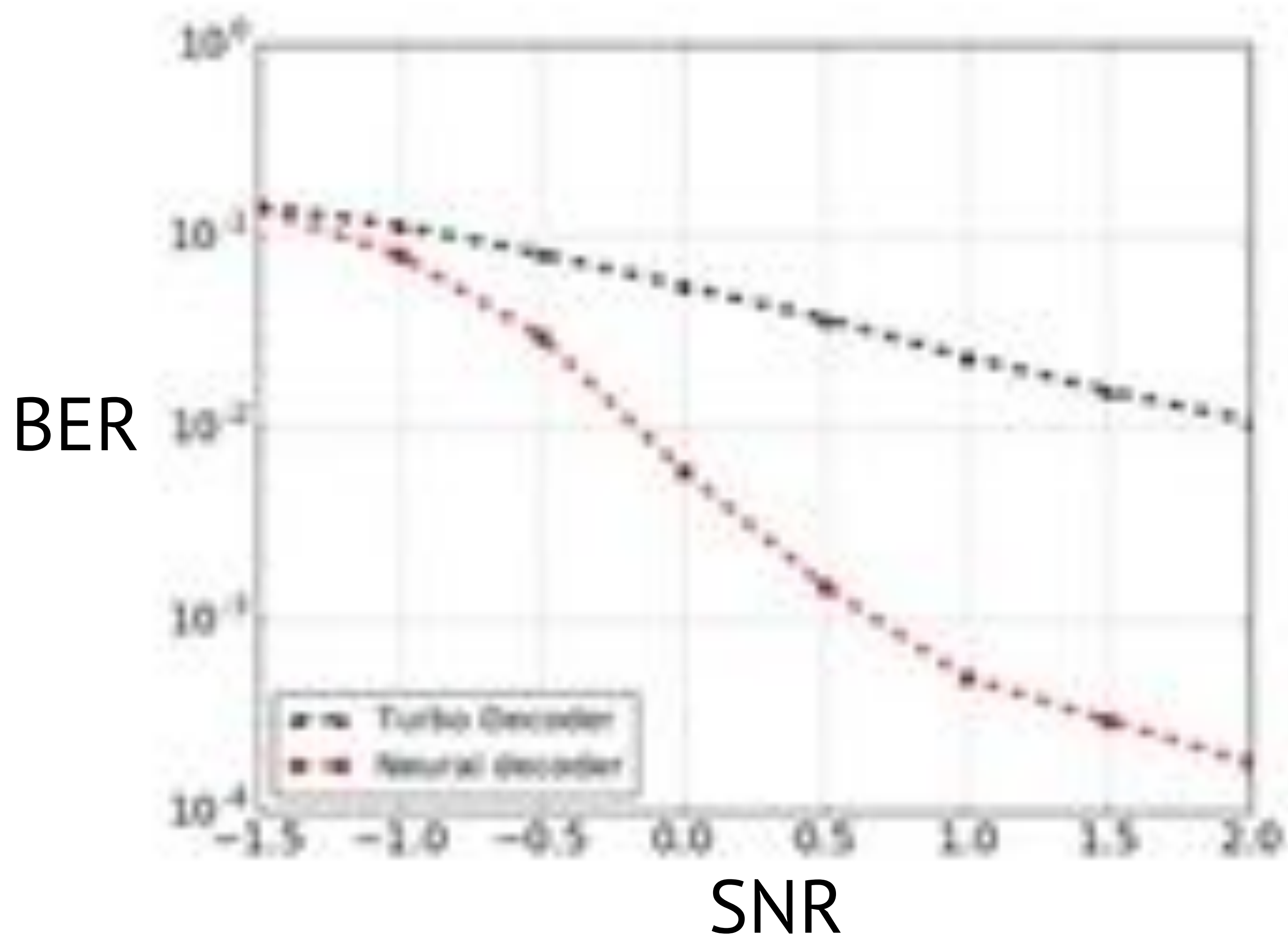


SNR

(block length = 1000)

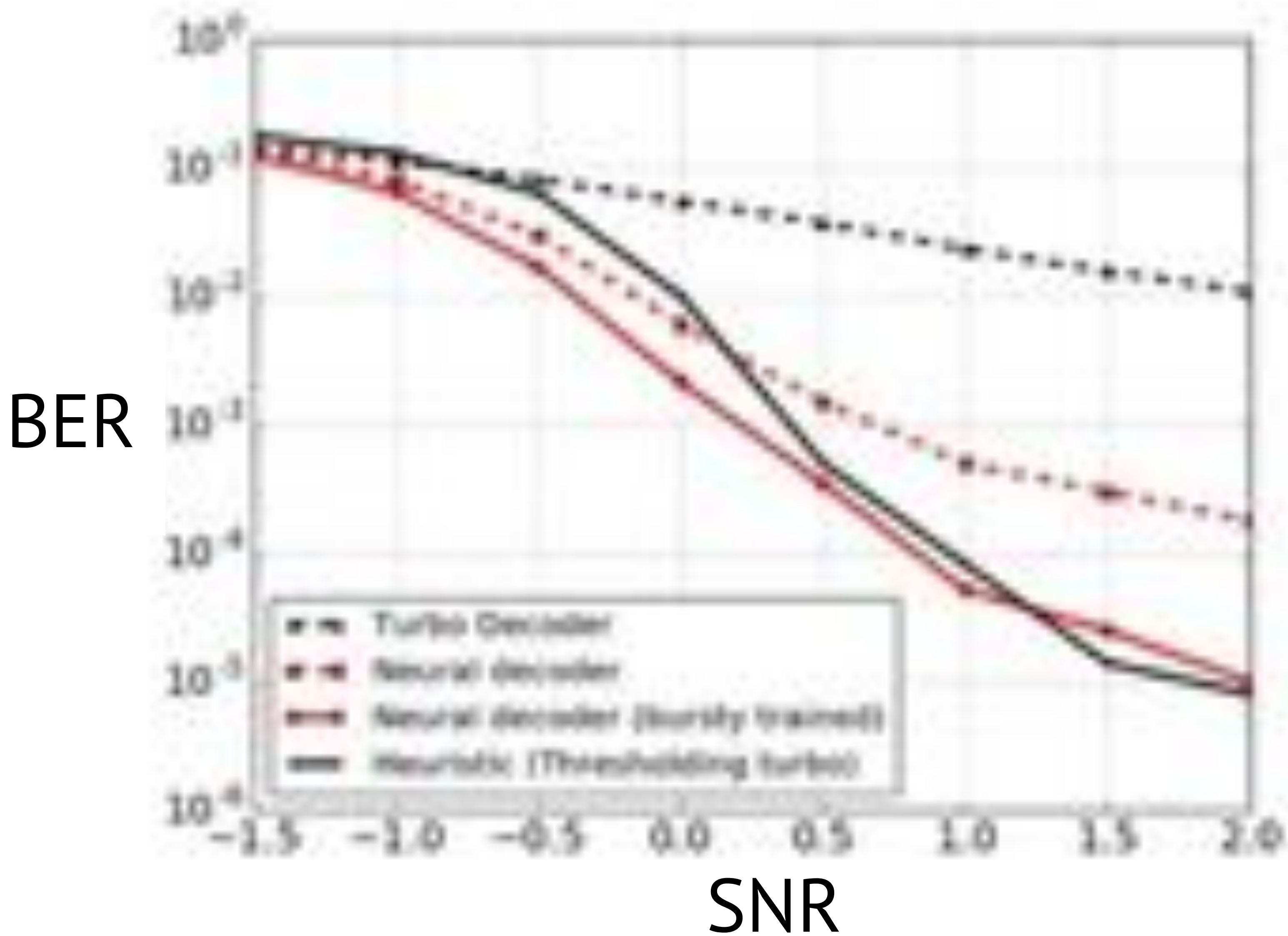
Robustness: Decoding turbo codes under bursty noise

- Neural decoder is more reliable under bursty noise



Adaptivity: Decoding turbo codes under bursty noise

- Neural decoder performs better than heuristic decoders



Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

Neural decoders

- Decoding linear codes
  - Generalized BP decoder

Eliya Nachmani, Yair Be'ery, David Burshtein,
"**Learning to decode linear codes using deep learning**", 2016

Eliya Nachmani, Yaron Bachar, Elad Marciano, David Burshtein, Yair Be'ery,
"**Near Maximum Likelihood Decoding with Deep Learning**", 2018

Neural decoders

- Decoding polar codes
  - Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, Stephan ten Brink, "***On deep learning-based channel decoding***", 2017

- Decoding under molecular channels
  - Nariman Farsad, Andrea Goldsmith, *"* ***Neural Network Detection of Data Sequences in Communication Systems****"*, 2018

Outline

- Part I. Discovering neural **codes**
  - Example: channels with feedback
  - Literature
  - Open problems
- Part II. Discovering neural **decoders**
  - Example: robust/adaptive neural decoding
  - Literature
  - Open problems

Open problems

- Decoding under
  - channels with memory, deletion channels
  - practical channels with intractable model

Open problems

- Decoding under
  - channels with memory, deletion channels
  - practical channels with intractable model
- Adaptive and robust decoders
  - fast adaptation to varying channels

Summary

- Human ingenuity has been the driving force behind designing codes for past century

- We provide an alternative approach — training neural networks — and demonstrate its powerfulness with feedback code design

- It has great potential to provide new solutions to numerous challenges in communications

Summary

- It is critical to bring intuitions and knowledge from communications and information theory

- Along the way, we bring new ideas and intuition to deep learning methodology

- By interpreting neural communication algorithms, we gain new ideas and insights in code design

Collaborators

Yihan Jiang

Ranvir Rana

Sreeram Kannan

Sewoong Oh

Pramod Viswanath

Deep Learning
for Statistical Inference

Organization: This Tutorial

Part-1: Deep learning for information theory

1a. Deep learning for communication

1b. Deep learning for statistical inference

Part-2: Information theory for deep learning

2a. Theory for GAN

2b. Learning Gated Neural Networks

Collaborators

Rajat
Sen

UT, Austin

Karthikeyan
Shanmugan

IBM Research

Arman
Rahimzamani

UW, Seattle

Himanshu
Asnani

UW, Seattle

Beyond Coding

Two successes of Deep Learning

- Strong classifiers
- Powerful Generative Models

Beyond Coding

Two successes of Deep Learning

- Strong classifiers
- Powerful Generative Models

Statistical Inference Applications

- Conditional Independence Testing
- Estimating Information Measures
- Compressed Sensing
- Community Detection

Classifiers

- Deep NN and boosted random forests achieve state-of-the-art performance
- Works very well even in practice when X is high dimensional.
- Exploits generic inductive bias:
  - Invariance
  - Hierarchical Structure
  - Symmetry

Classifiers

- Deep NN and boosted random forests achieve state-of-the-art performance
- Works very well even in practice when X is high dimensional.
- Exploits generic inductive bias:
  - Invariance
  - Hierarchical Structure
  - Symmetry

Classifiers

- Deep NN and boosted random forests achieve state-of-the-art performance
- Works very well even in practice when X is high dimensional.
- Exploits generic inductive bias:
  - Invariance
  - Hierarchical Structure
  - Symmetry

Theoretical guarantees lag severely behind practice!

Generative Models

Generative Models

$z \longrightarrow$ Generator $\longrightarrow x$

Low-dimensional Latent Space

High-dimensional data Space

- Trained Real Samples of x

- Can generate any number of new samples

Generative Models

z → **Generator** → x

Low-dimensional Latent Space → High-dimensional data Space

- Trained Real Samples of x

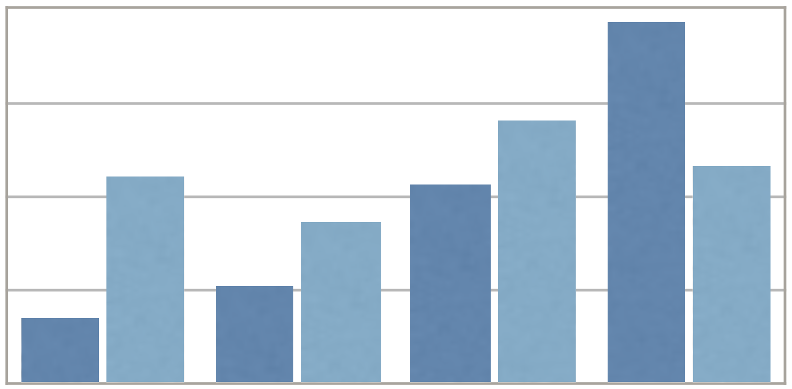
- Can generate any number of new samples

Statistical Inference Applications

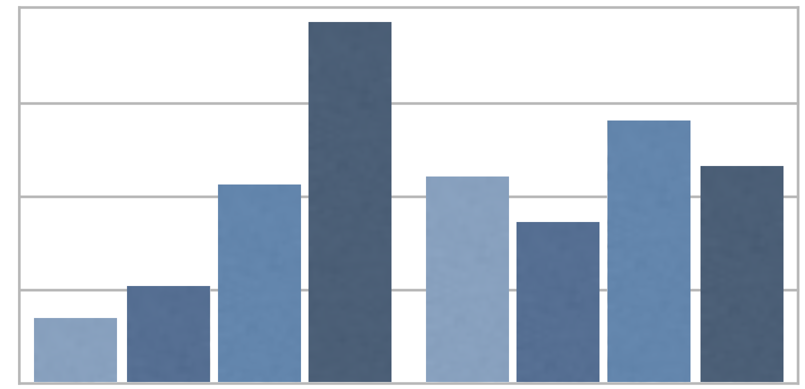
- Conditional Independence Testing
- Estimating Information Measures
- Compressed Sensing
- Community Detection

Conditional Independence Testing

Estimating Total Variation Distance

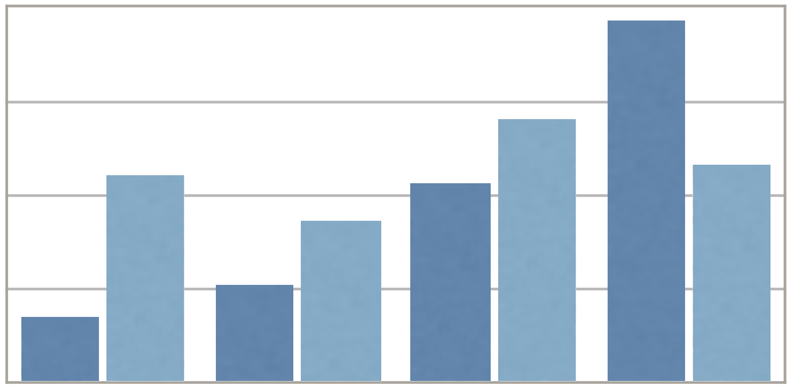


P



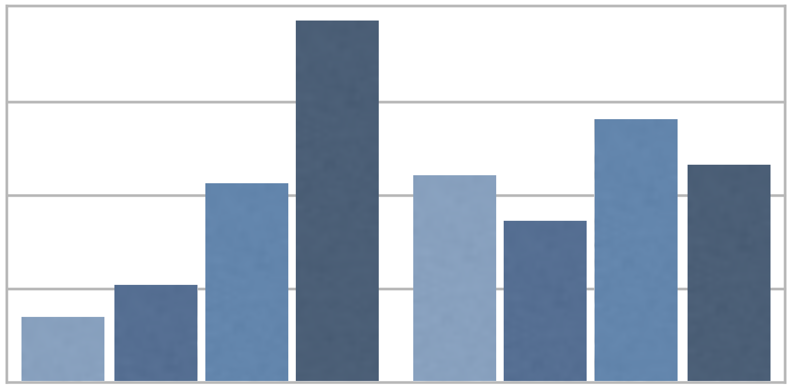
Q

Estimating Total Variation Distance



P

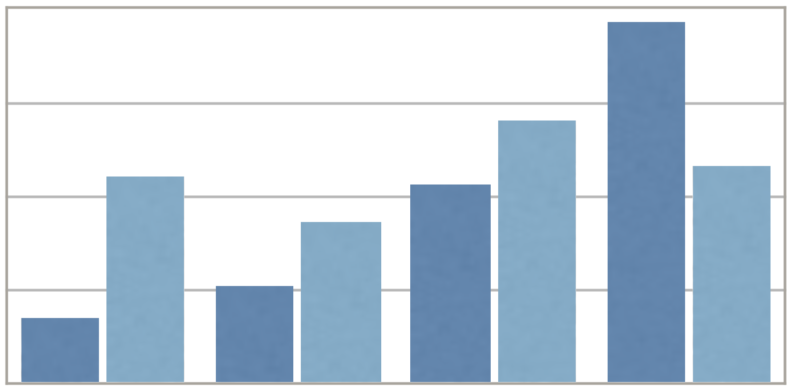
n samples



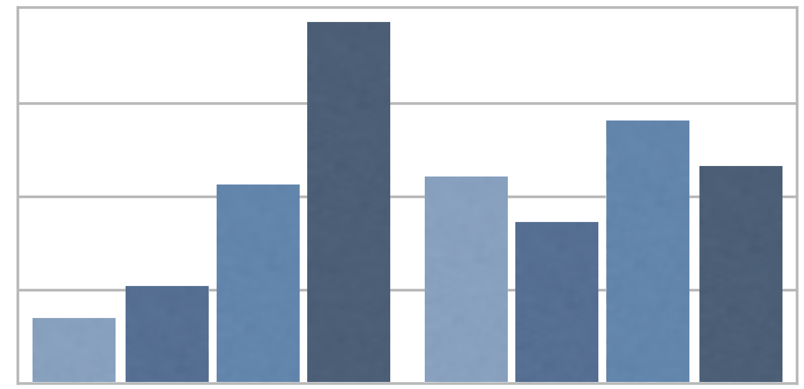
Q

n samples

Estimating Total Variation Distance



P



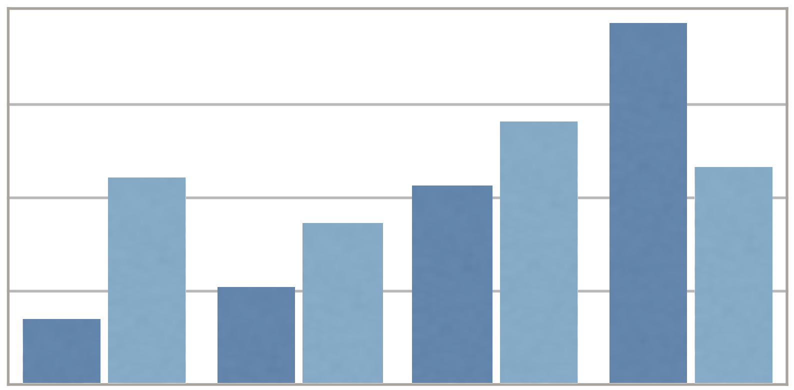
Q

n samples

n samples

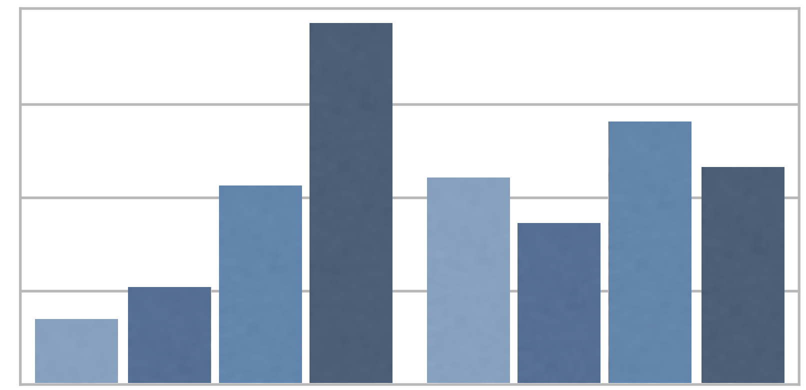
Estimate $D_{TV}(P, Q)$?

Estimating Total Variation Distance



P

n samples



Q

n samples

Estimate $D_{TV}(P,Q)$?

P and Q can be arbitrary.

Search beyond Traditional Density Estimation Methods

Total Variation Estimation : Prior Art

- Lots of work in information theory on D_{TV} testing
- Based on closeness testing between P and Q
- Sample complexity = $O(n^{2/3})$, where n = alphabet size
- Not much is known in the real-valued case

* Chan et al, **Optimal Algorithms for testing closeness of discrete distributions,** *SODA 2014*

* Sriperumbudur et al, **Kernel choice and classifiability for RKHS embeddings of probability distributions,** *NIPS 2009*

Total Variation Estimation : Prior Art

- Lots of work in information theory on D_{TV} testing
- Based on closeness testing between P and Q
- Sample complexity = $O(n^{2/3})$, where n = alphabet size

Curse of dimensionality

- Not much is known in the real-valued case

* Chan et al, **Optimal Algorithms for testing closeness of discrete distributions**, *SODA 2014*

* Sriperumbudur et al, **Kernel choice and classifiability for RKHS embeddings of probability distributions**, *NIPS 2009*

Total Variation Estimation : Prior Art

- Lots of work in information theory on D_{TV} testing
- Based on closeness testing between P and Q
- Sample complexity = $O(n^{2/3})$, where n = alphabet size

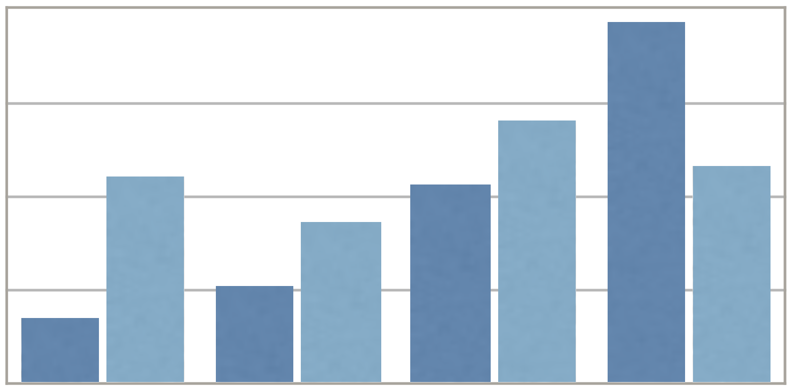
Curse of dimensionality

- Not much is known in the real-valued case

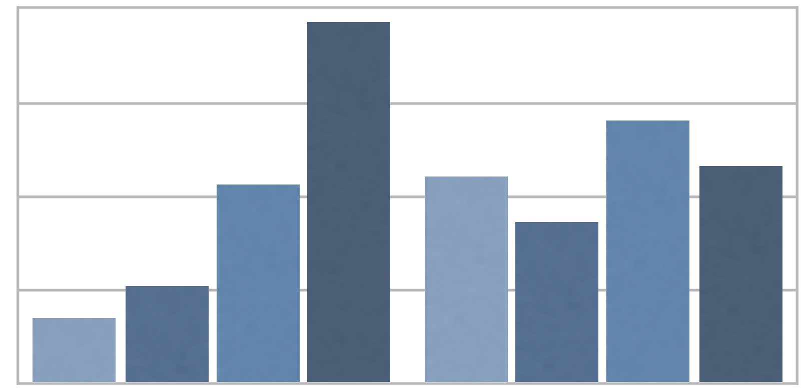
Leverage classifiers which exploit generic inductive bias!

* Chan et al, **Optimal Algorithms for testing closeness of discrete distributions**, *SODA 2014*

Distance Estimation via Classification

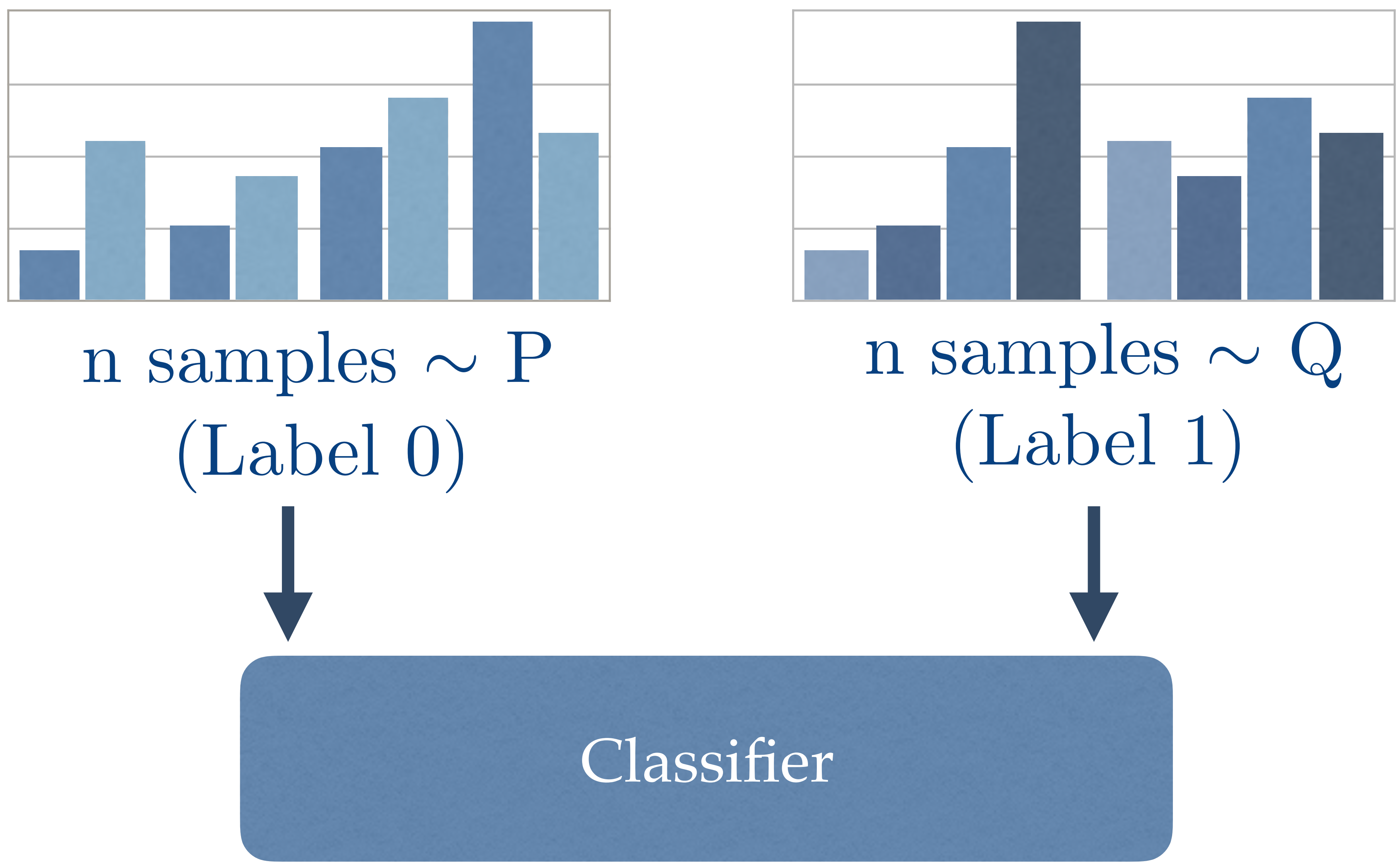


n samples $\sim$ P

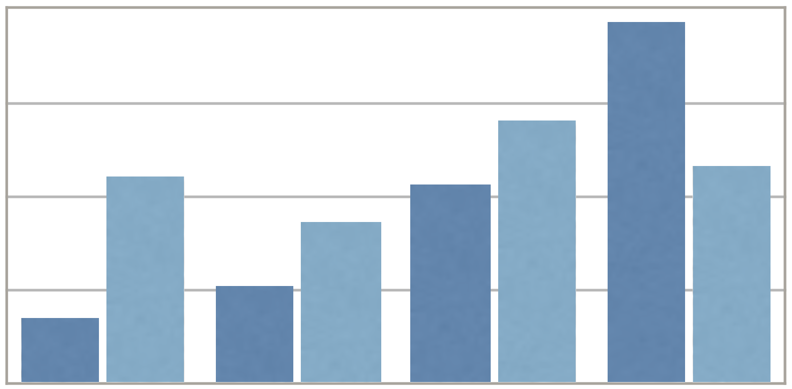


n samples $\sim$ Q

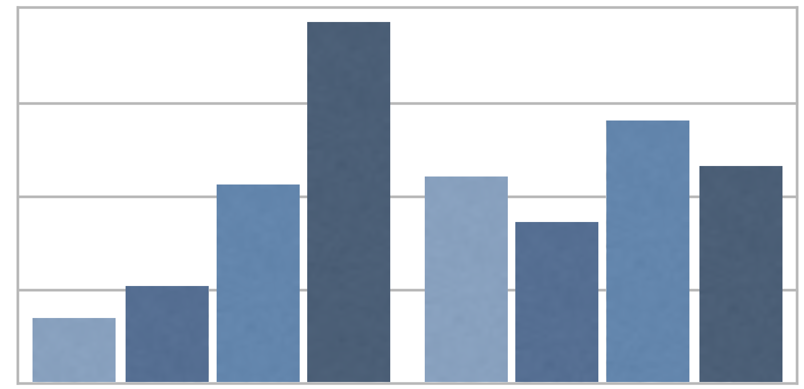
Distance Estimation via Classification



Distance Estimation via Classification



n samples $\sim$ P
(Label 0)

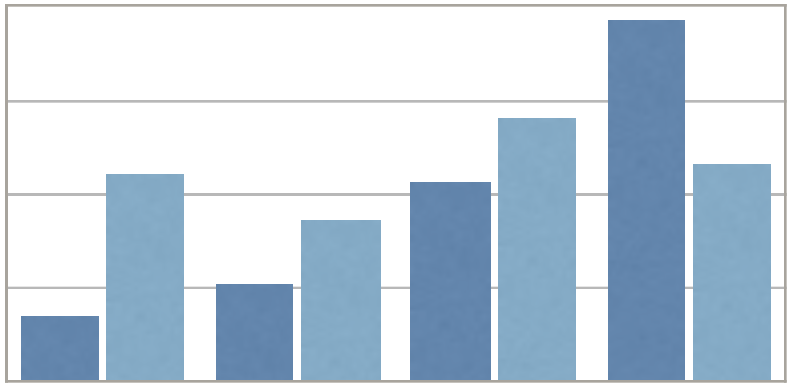


n samples $\sim$ Q
(Label 1)

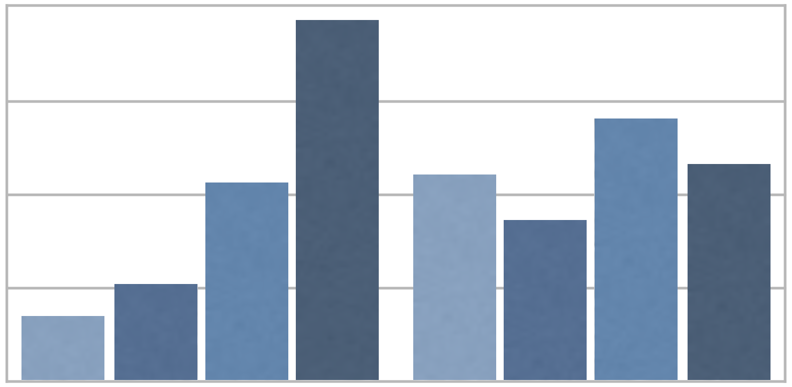
Classifier

Classification Error of Optimal Bayes Classifier $= \frac{1}{2} - \frac{1}{2} D_{\mathrm{TV}}(P, Q).$

Distance Estimation via Classification



n samples $\sim$ P
(Label 0)



n samples $\sim$ Q
(Label 1)

Deep NN, Boosted Trees etc.

Classification Error of Optimal Classifier $= \frac{1}{2} - \frac{1}{2} D_{\mathrm{TV}}(P, Q).$

* Lopez-Paz et al, **Revisiting Classifier two-sample tests**, *ICLR 2017*

* Sriperumbudur et al, **Kernel choice and classifiability for RKHS embeddings of probability distributions,** *NIPS 2009*

Independence Testing

n samples $\{x_i, y_i\}_{i=1}^{n}$

* Lopez-Paz et al, **Revisiting Classifier two-sample tests,** *ICLR 2017*

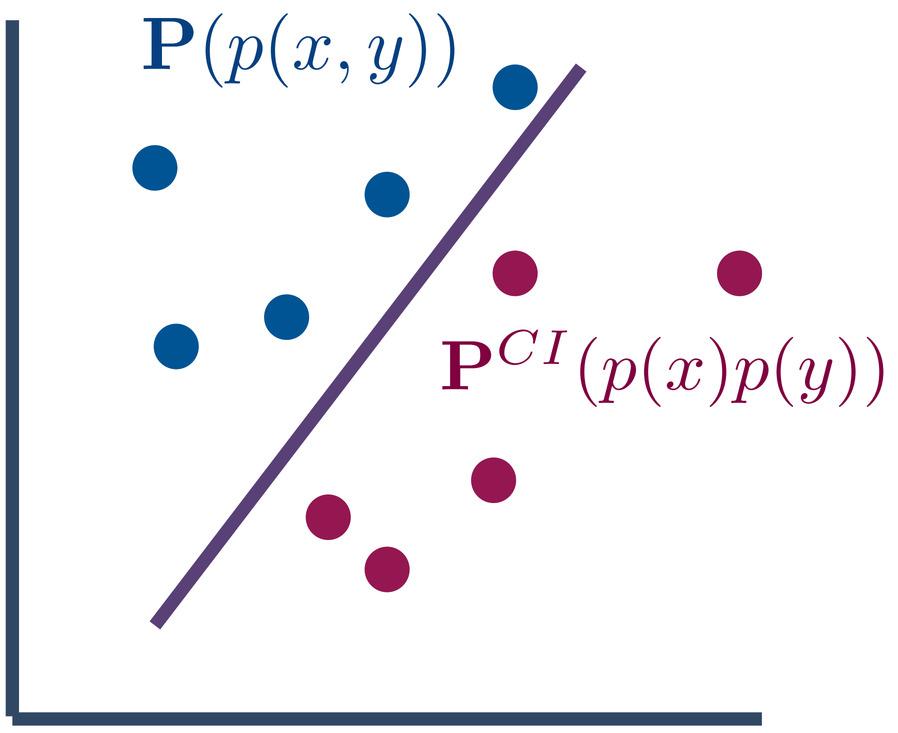
* Sriperumbudur et al, **Kernel choice and classifiability for RKHS embeddings of probability distributions,** *NIPS 2009*

Independence Testing

$$\text{n samples } \{x_i, y_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y \, (\mathbf{P}^{CI}) \\ \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y \, (\mathbf{P}) \end{cases}$$

Independence Testing

$$\text{n samples } \{x_i, y_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y (\mathbf{P}^{CI}) \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y (\mathbf{P}) \end{cases}$$

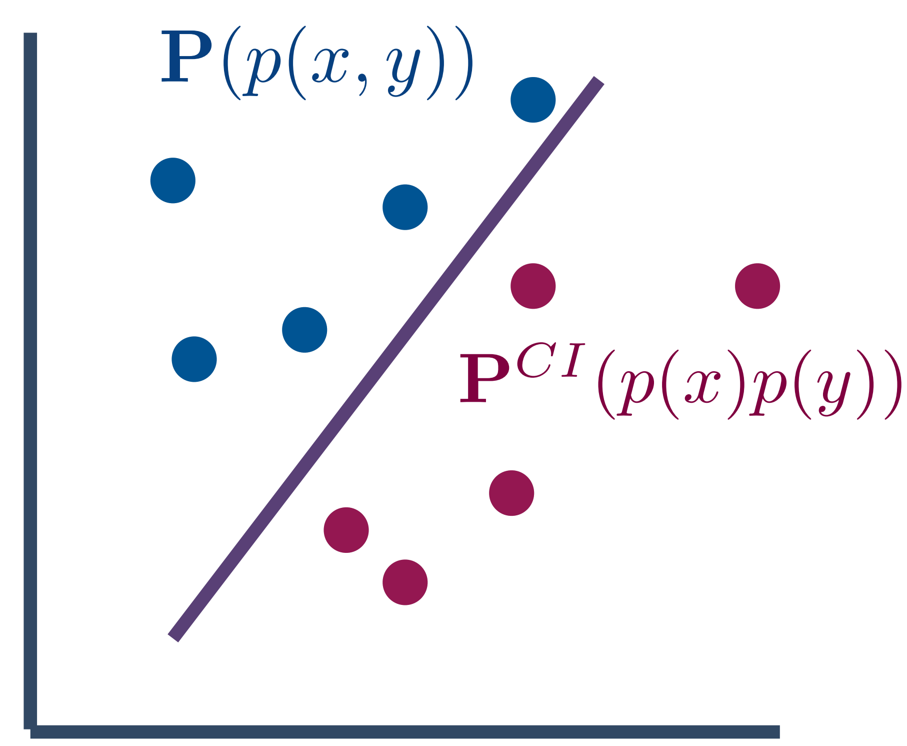


Classify

Independence Testing

$$\text{n samples } \{x_i, y_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y (\mathbf{P}^{CI}) \\ \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y (\mathbf{P}) \end{cases}$$

$$\mathbf{P}^{CI}(p(x)p(y)) \longrightarrow$$

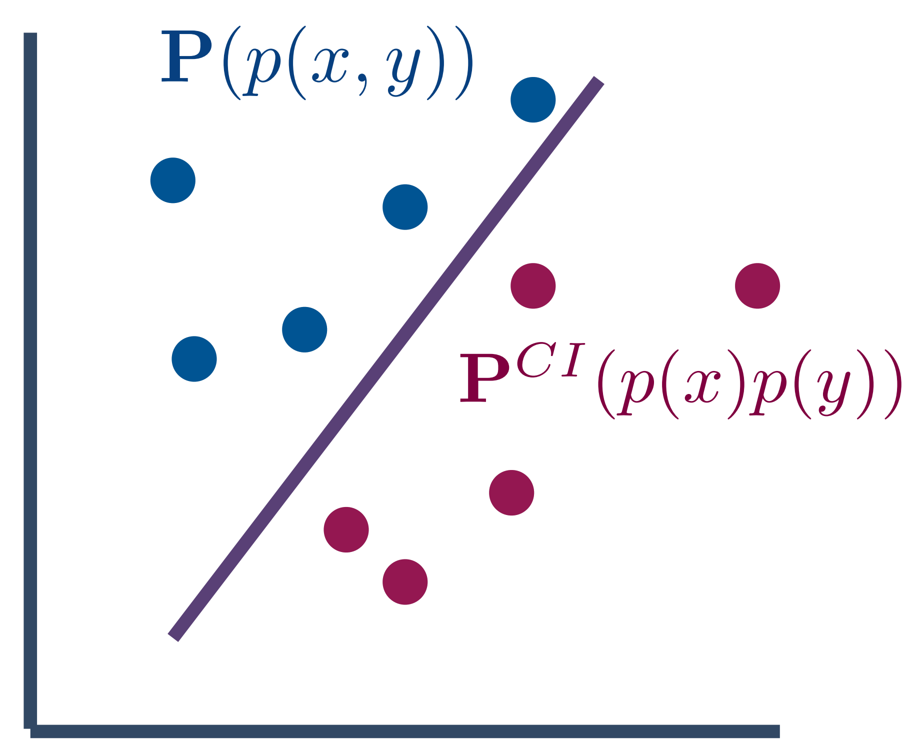


Classify

Independence Testing

$$\text{n samples } \{x_i, y_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y \, (\mathbf{P}^{CI}) \\ \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y \, (\mathbf{P}) \end{cases}$$

$$\mathbf{P}^{CI}(p(x)p(y))$$



Permutation

Classify

Independence Testing

n samples $\{x_i, y_i\}_{i=1}^{n}$

Split

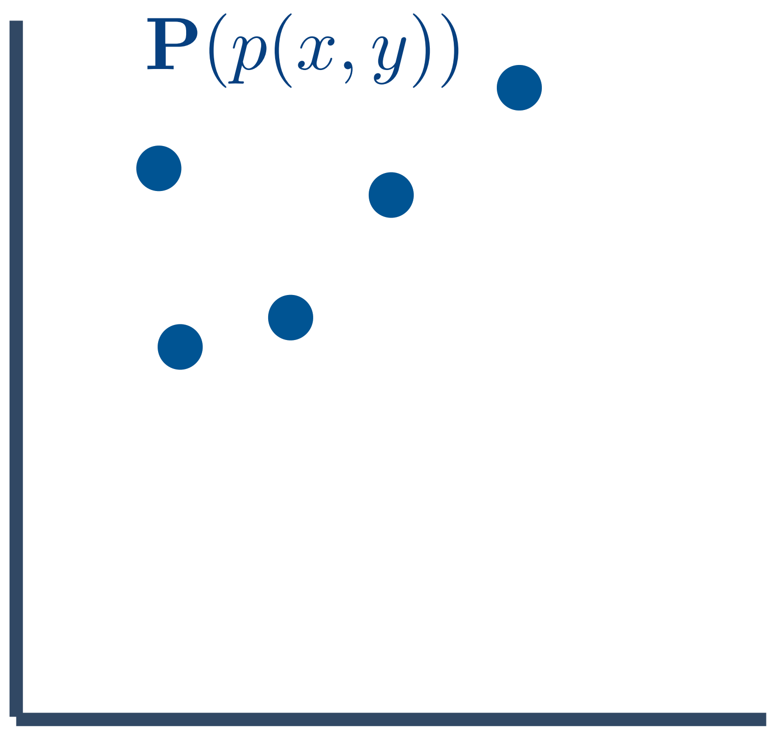
Equally

Independence Testing

n samples $\{x_i, y_i\}_{i=1}^{n}$

Split

Equally

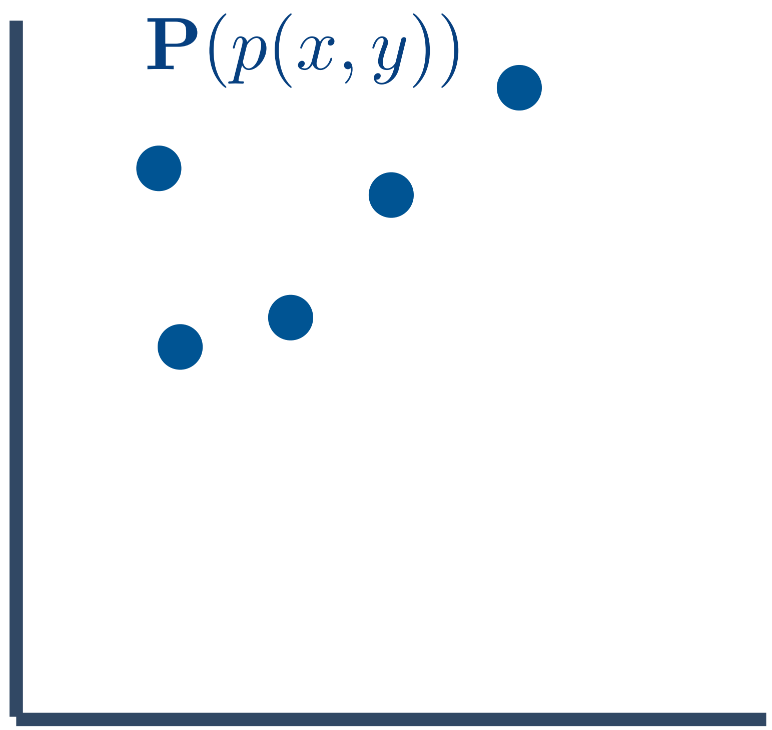


Independence Testing

n samples $\{x_i, y_i\}_{i=1}^{n}$

Split
Equally

Label 0



Independence Testing

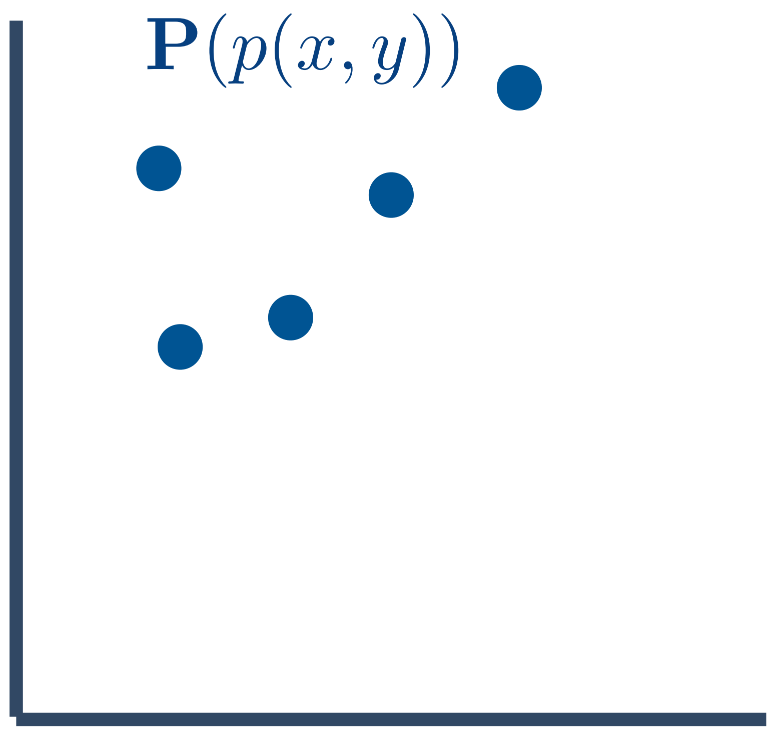
n samples $\{x_i, y_i\}_{i=1}^{n}$

Split

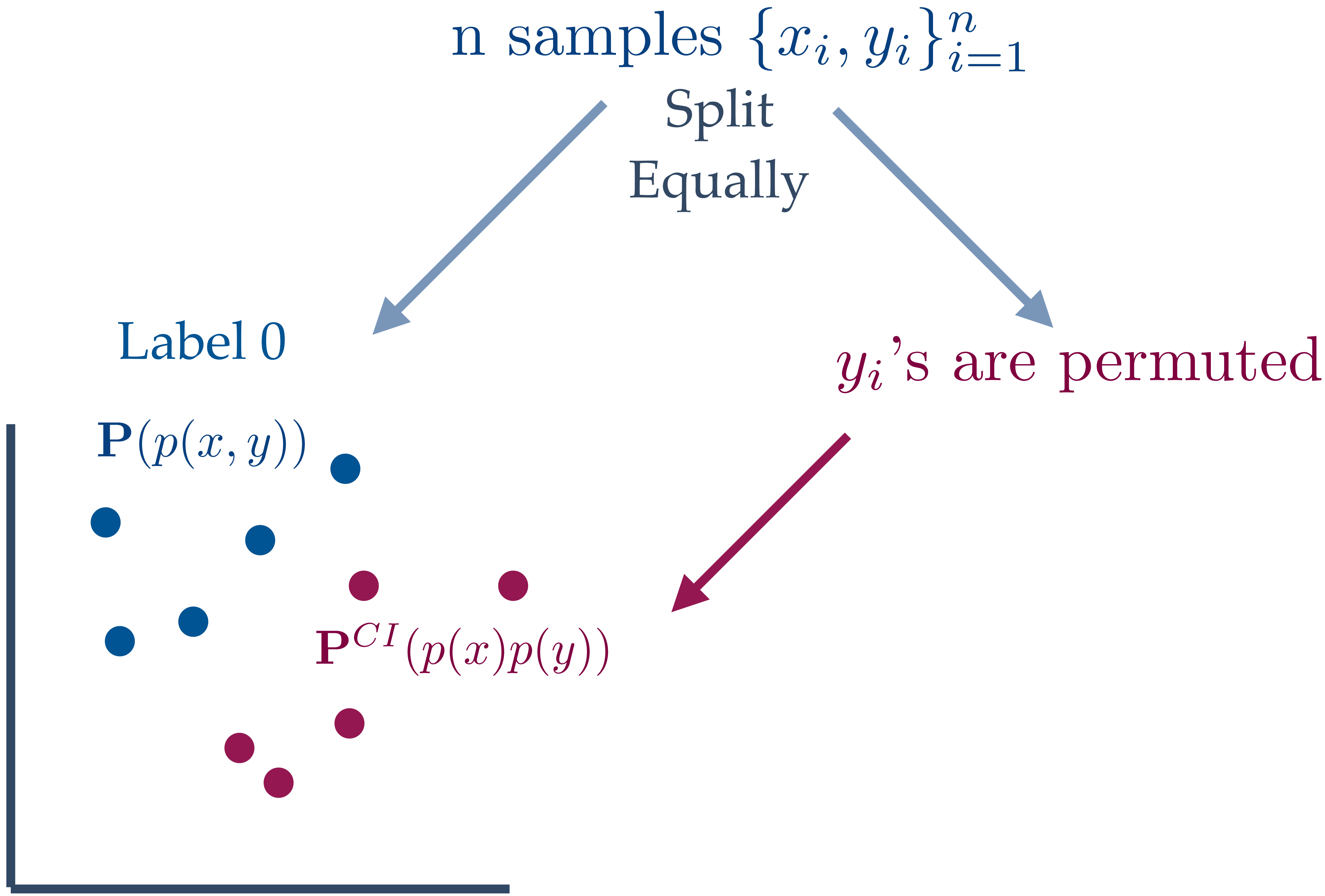
Equally

Label 0

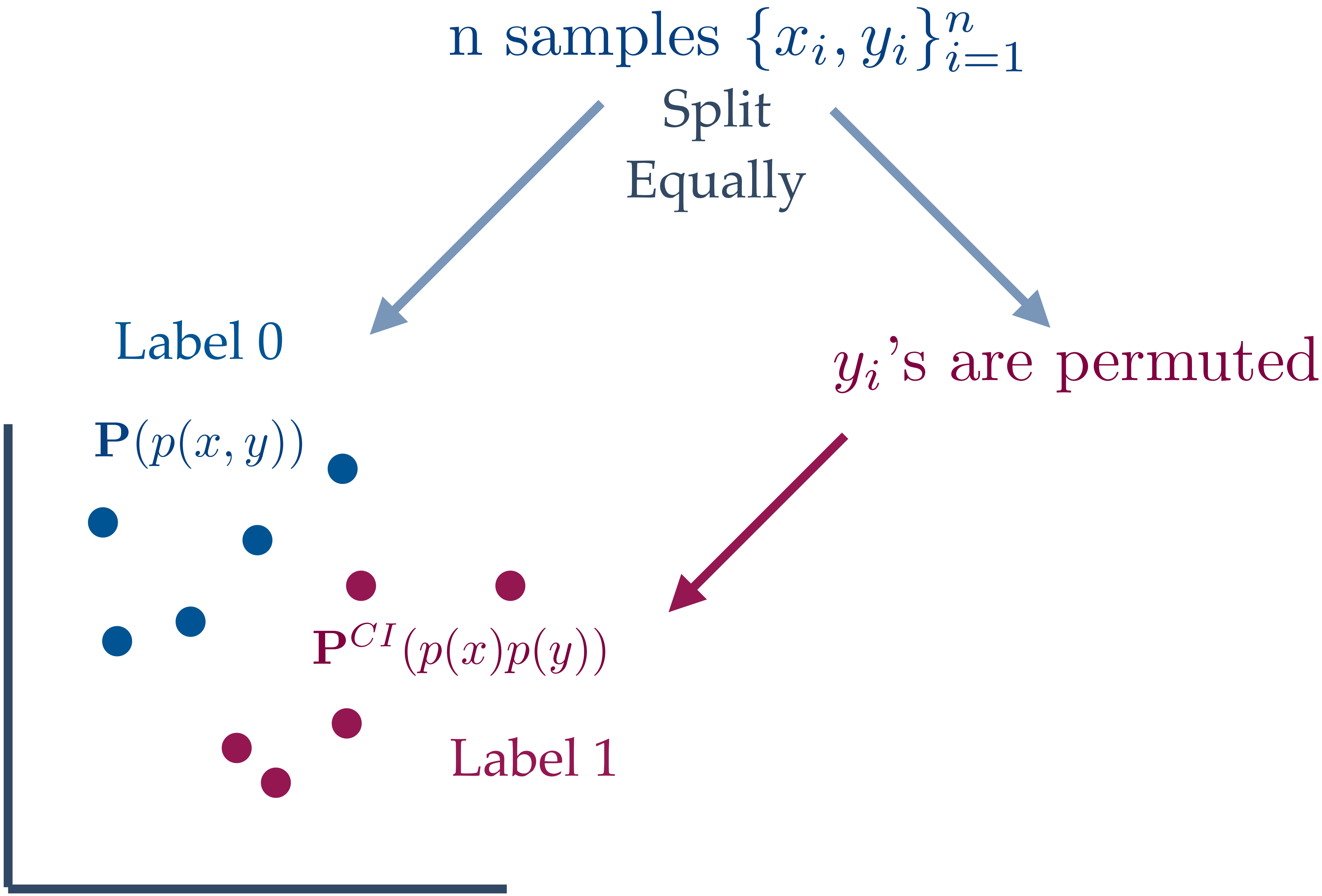
y_i's are permuted



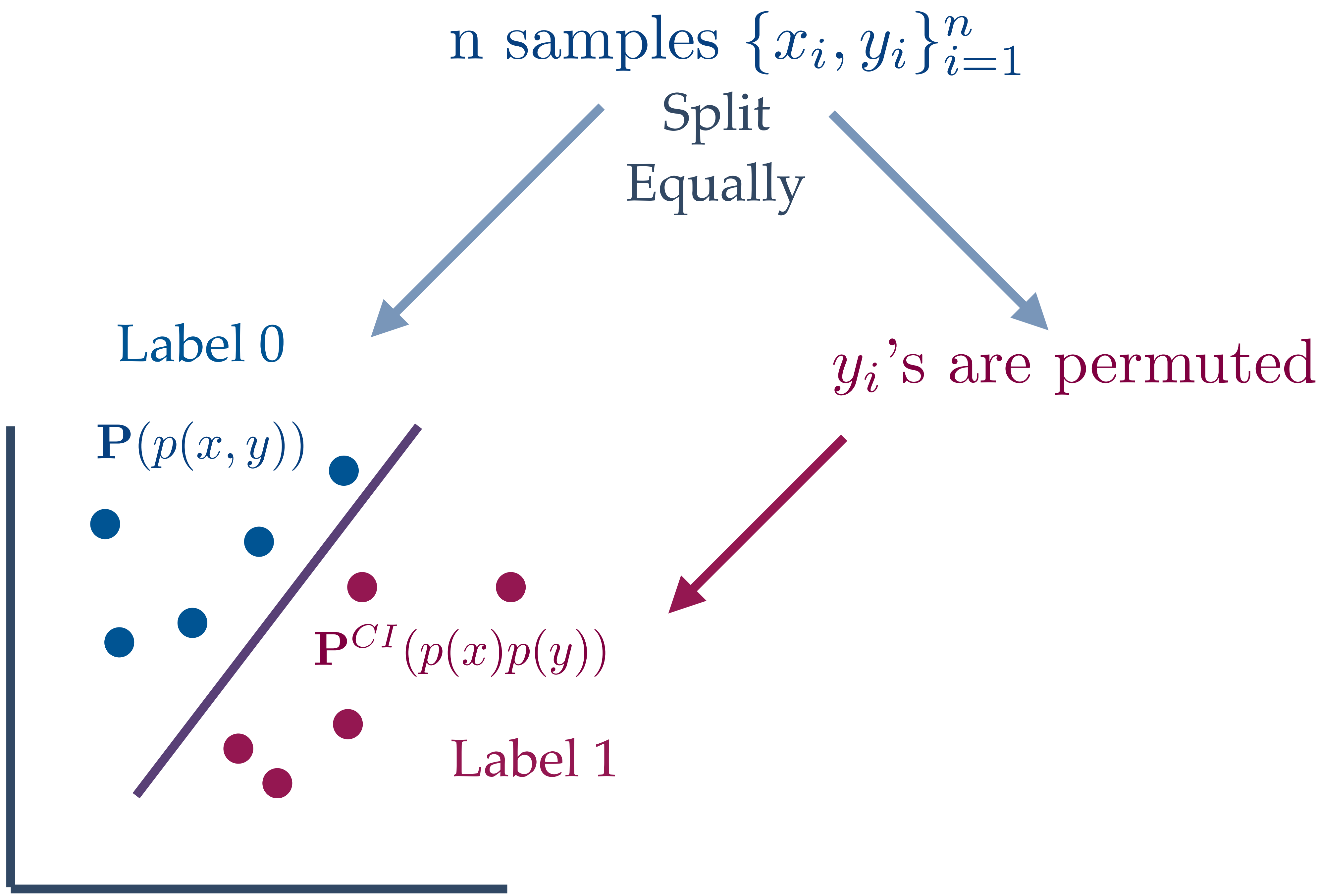
Independence Testing



Independence Testing



Independence Testing



*Lopez-Paz et al, **Revisiting Classifier two-sample tests**, *ICLR 2017*

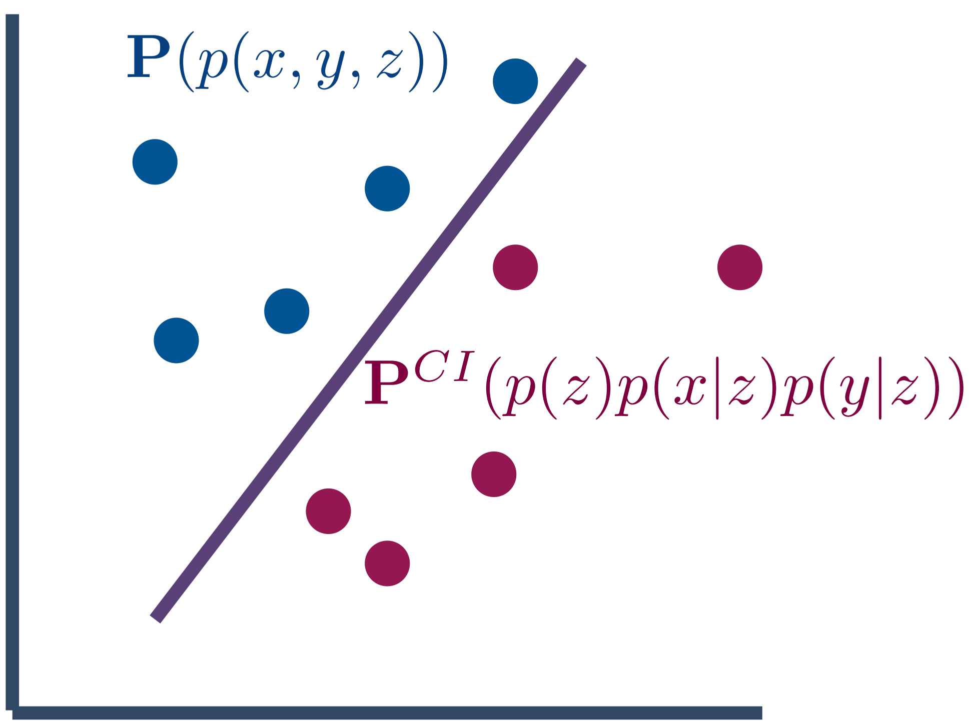
* Sriperumbudur et al, **Kernel choice and classifiability for RKHS embeddings of probability distributions**, *NIPS 2009*

Conditional Independence Testing

$$\text{n samples } \{x_i, y_i, z_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \ (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\!\perp\!\!\!\perp Y | Z \ (\mathbf{P}) \end{cases}$$

Conditional Independence Testing

n samples $\{x_i, y_i, z_i\}_{i=1}^n$ $\begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \ (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y | Z \ (\mathbf{P}) \end{cases}$

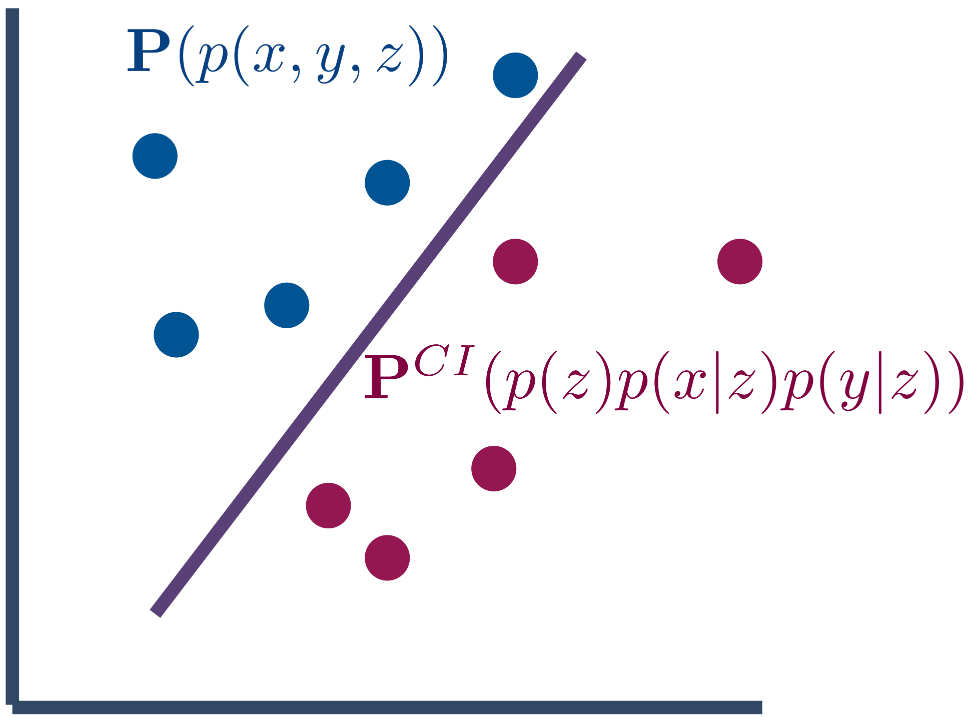


Classify

Conditional Independence Testing

n samples $\{x_i, y_i, z_i\}_{i=1}^{n}$ $\begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \;(\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y | Z \;(\mathbf{P}) \end{cases}$

How to get $\mathbf{P}^{CI}(p(z)p(x|z)p(y|z)$?

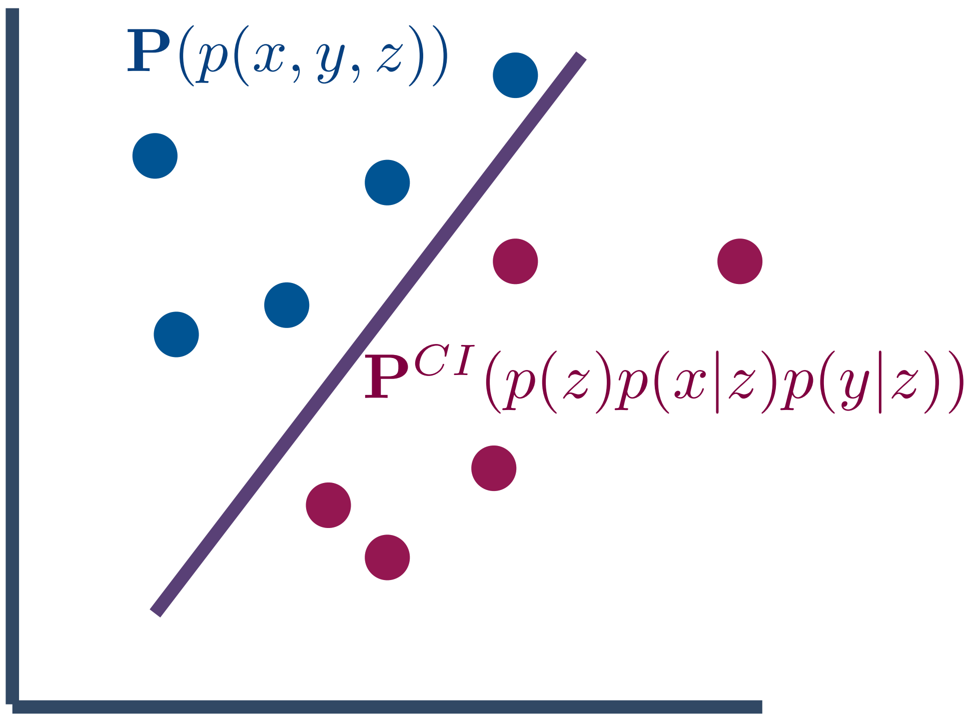


Classify

Conditional Independence Testing

$$\text{n samples } \{x_i, y_i, z_i\}_{i=1}^n \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \ (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y | Z \ (\mathbf{P}) \end{cases}$$

Given samples $\sim p(x, z)$
How to emulate $p(y|z)$?



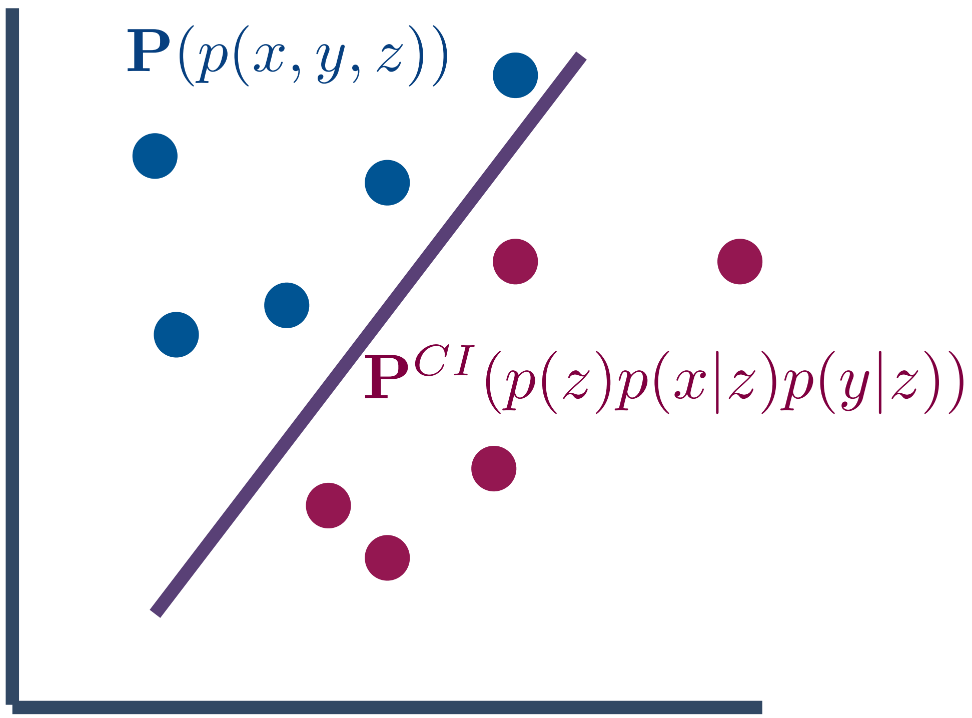
Classify

Conditional Independence Testing

$$\text{n samples } \{x_i, y_i, z_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \; (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y | Z \; (\mathbf{P}) \end{cases}$$

Emulate $p(y|z)$ as $q(y|z)$

- KNN Based Methods
- Kernel Methods



Classify

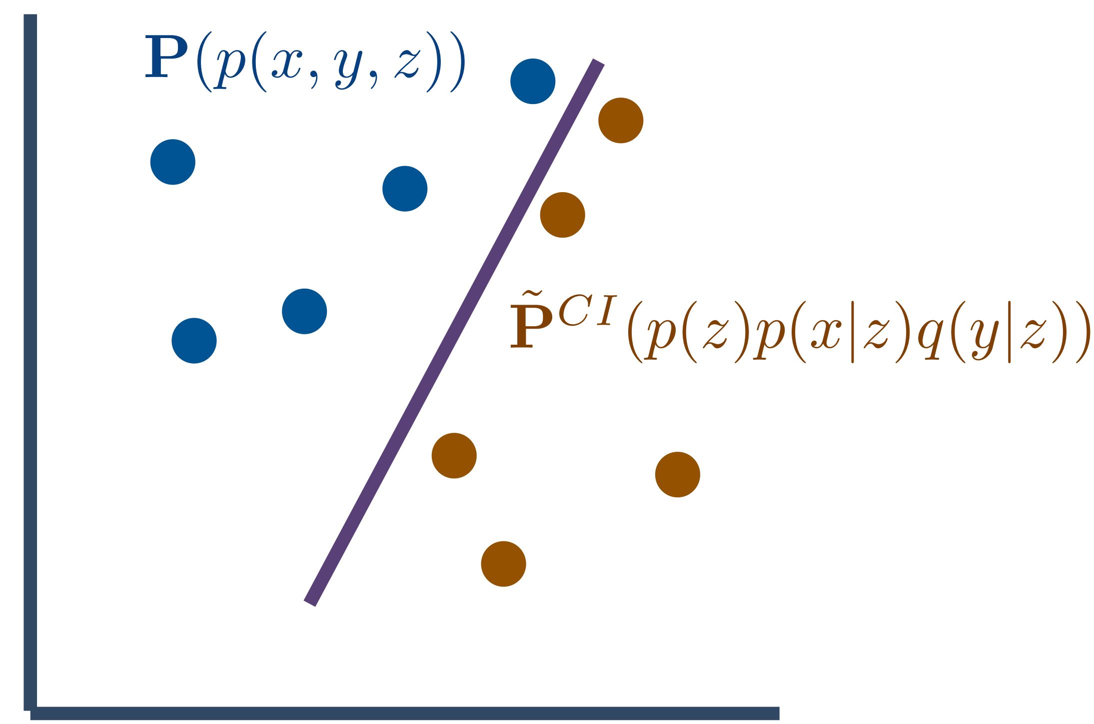
Conditional Independence Testing

n samples $\{x_i, y_i, z_i\}_{i=1}^{n}$
$$\begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \; (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\!\perp\!\!\!\perp Y | Z \; (\mathbf{P}) \end{cases}$$

Emulate $p(y|z)$ as $q(y|z)$

- KNN Based Methods
- Kernel Methods

$\tilde{\mathbf{P}}^{CI}(p(z)p(x|z)q(y|z))$



Classify

Conditional Independence Testing

$$\text{n samples } \{x_i, y_i, z_i\}_{i=1}^{n} \begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \ (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y | Z \ (\mathbf{P}) \end{cases}$$

- [KCIT] Gretton et al, **Kernel-based conditional independence test and application in causal discovery,** *NIPS 2008*
- [KCIPT] Doran et al, **A permutation-based kernel conditional independence test,** *UAI 2014*
- [CCIT] Sen et al, **Model-Powered Conditional Independence Test,** *NIPS 2017*
- [RCIT] Strobl et al, **Approximate Kernel-based Conditional Independence Tests for Fast Non-Parametric Causal Discovery,** *arXiv*

Classify

Conditional Independence Testing

n samples $\{x_i, y_i, z_i\}_{i=1}^{n}$ $\begin{cases} \mathcal{H}_0 : X \perp\!\!\!\perp Y | Z \ (\mathbf{P}^{CI}) \\ \text{vs} \\ \mathcal{H}_1 : X \not\!\perp\!\!\!\perp Y | Z \ (\mathbf{P}) \end{cases}$

Emulate $p(y|z)$ as $q(y|z)$

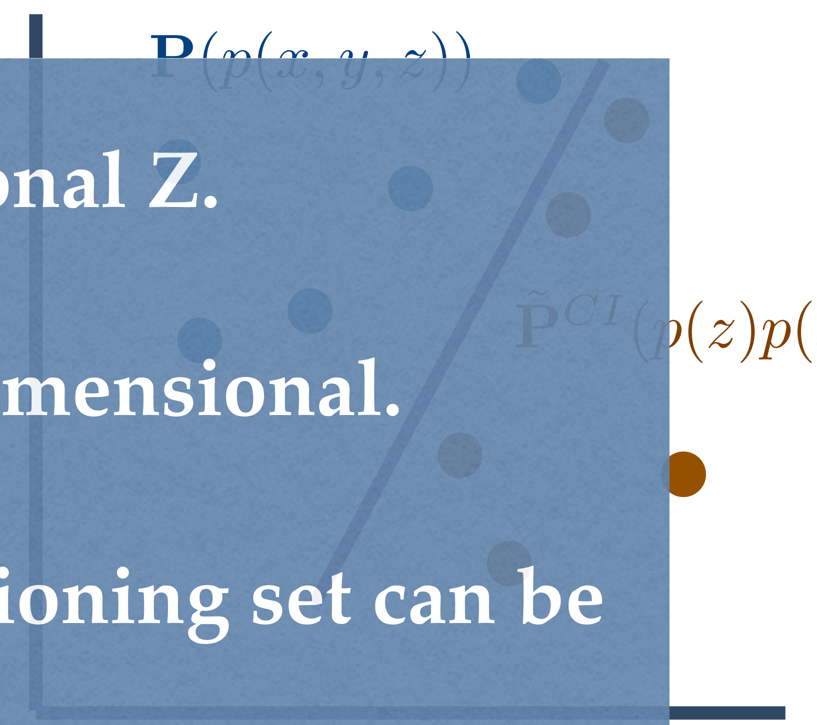
- KNN Based Methods
- Kernel Methods

$\tilde{\mathbf{P}}^{CI}(p(z)p(x|z)q(y|z))$

Limited to low-dimensional Z.

In practice, Z is often high dimensional.

(Eg. In graphical model, conditioning set can be entire graph.)



Classify

How loose can the estimate be for $\tilde{\mathbf{P}}^{CI}$ or $q(y|z)$?

How loose can the estimate be for $\tilde{\mathbf{P}}^{CI}$ or $q(y|z)$?

Novel Bias Cancellation Method in Mimic-and-Classify works

$$\boxed{\text{As long as the density function } q(\mathbf{y}|\mathbf{z}) > 0 \text{ whenever } p(\mathbf{y},\mathbf{z}) > 0.}$$

How loose can the estimate be for $\tilde{\mathbf{P}}^{CI}$ or $q(y|z)$?

Novel Bias Cancellation Method in Mimic-and-Classify works

$\boxed{\text{As long as the density function } q(\mathbf{y}|\mathbf{z}) > 0 \text{ whenever } p(\mathbf{y}, \mathbf{z}) > 0.}$

Mimic Functions : GANs, Regressors etc.

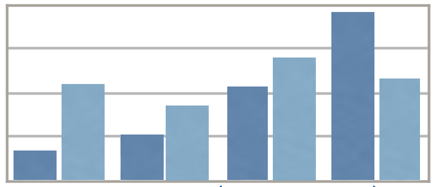
Mimic and Classify

Mimic Step

Classify Step

Mimic and Classify

Mimic
Step

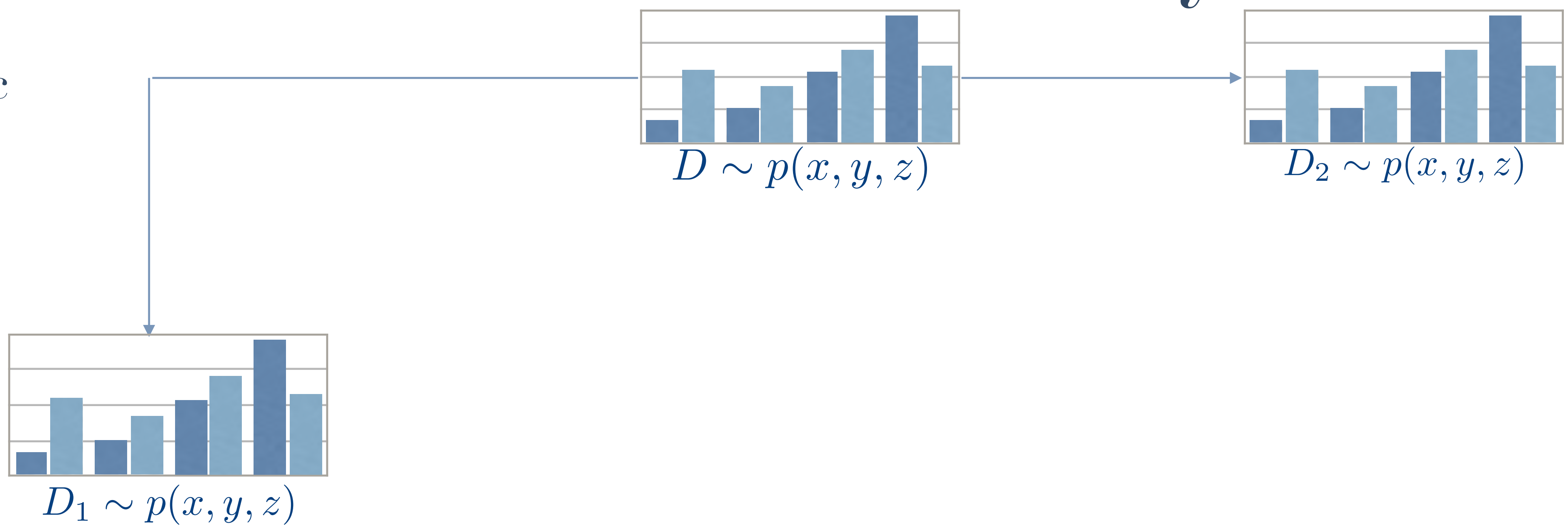


$D \sim p(x, y, z)$

Classify
Step

Mimic and Classify

Mimic Step



Classify Step

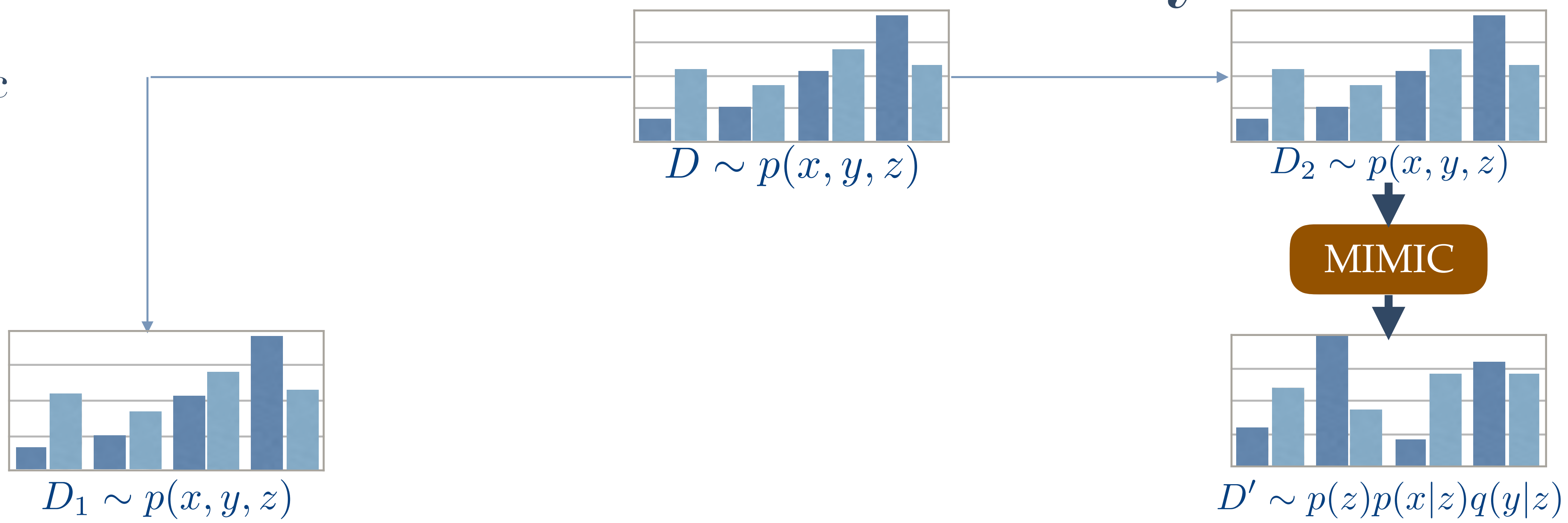
Mimic and Classify

Mimic Step

Classify Step

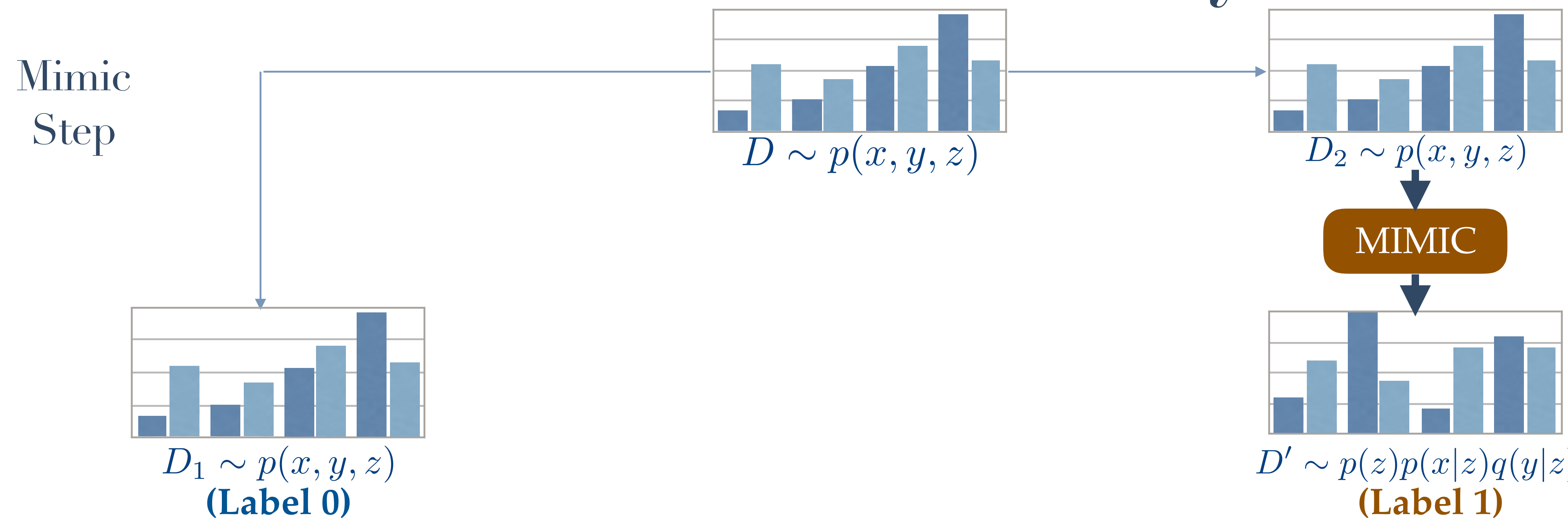
Mimic and Classify

Mimic
Step



Classify
Step

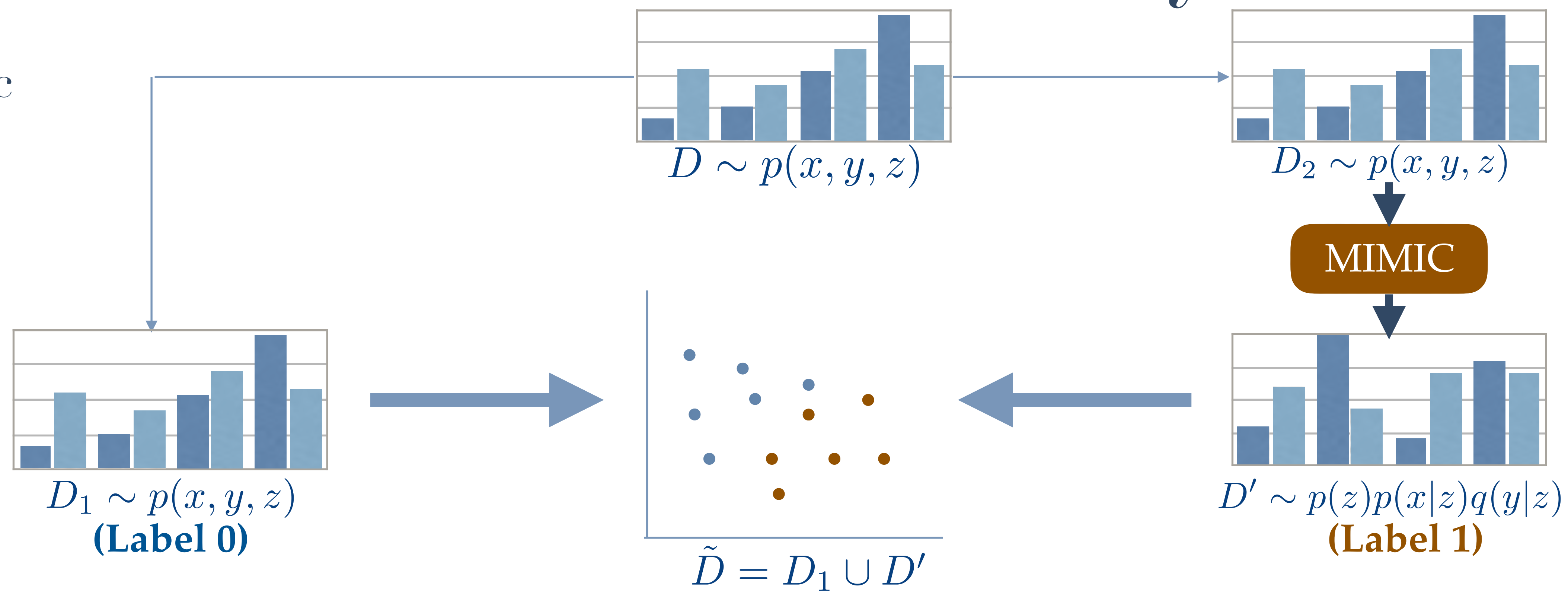
Mimic and Classify



Classify Step

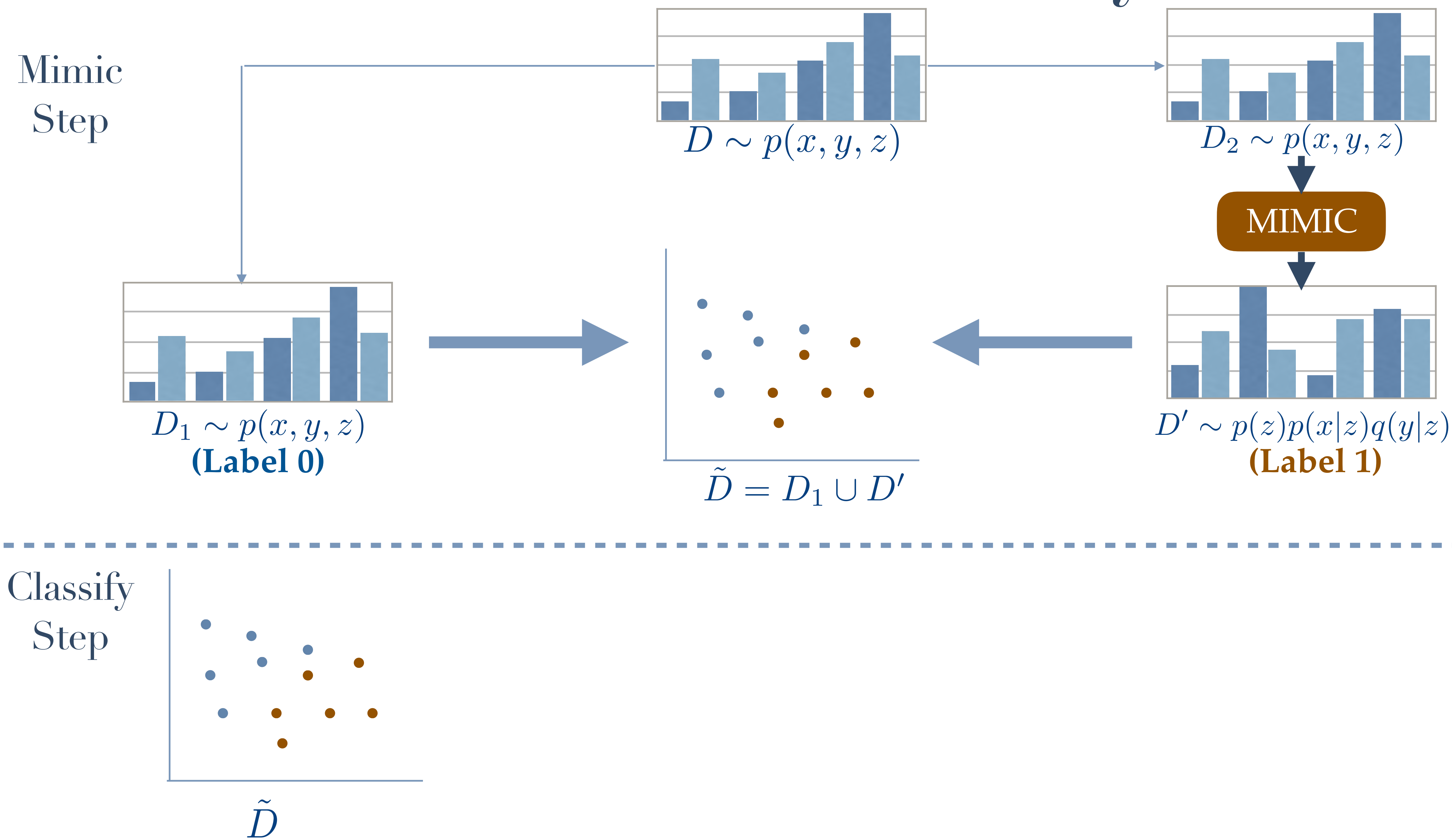
Mimic and Classify

Mimic Step

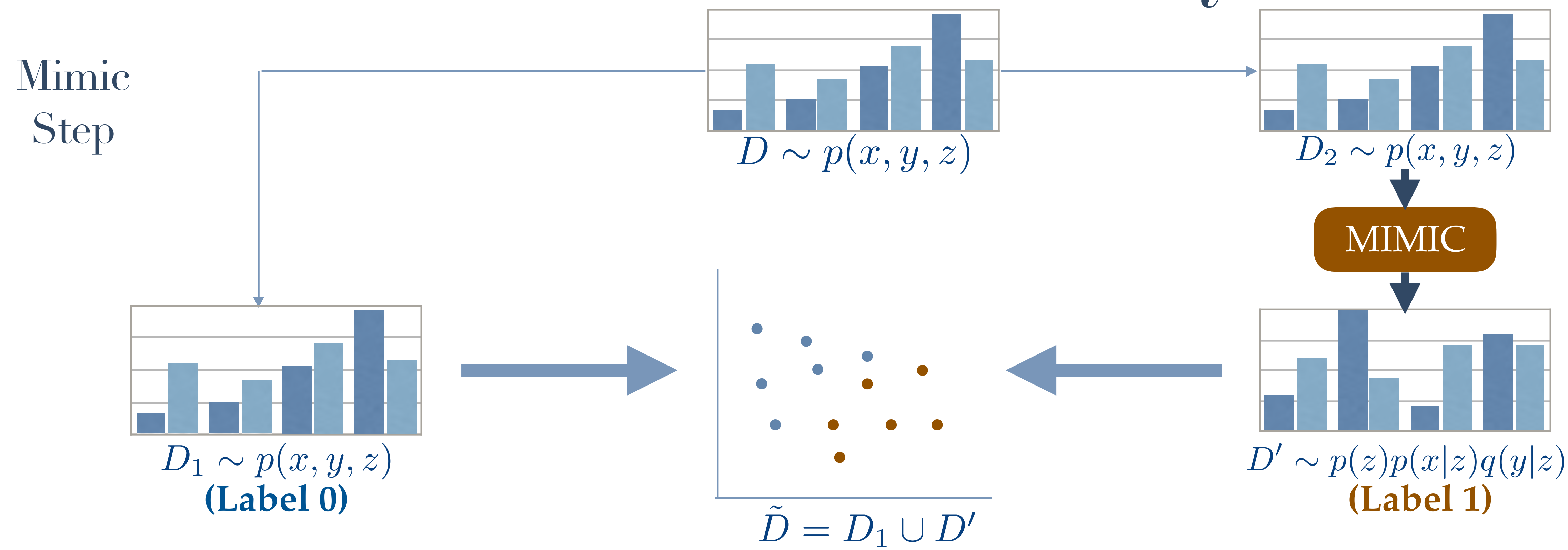


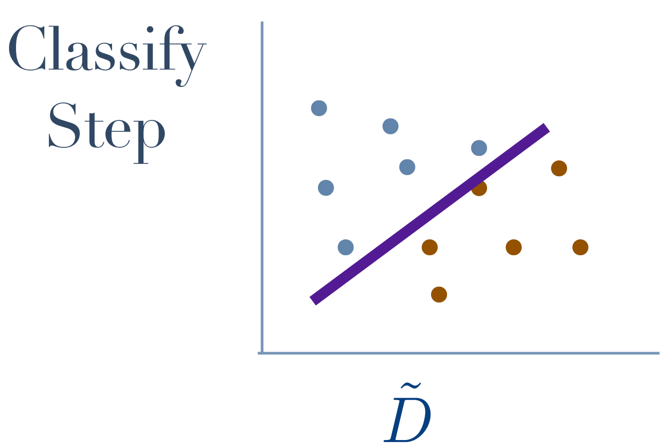
Classify Step

Mimic and Classify



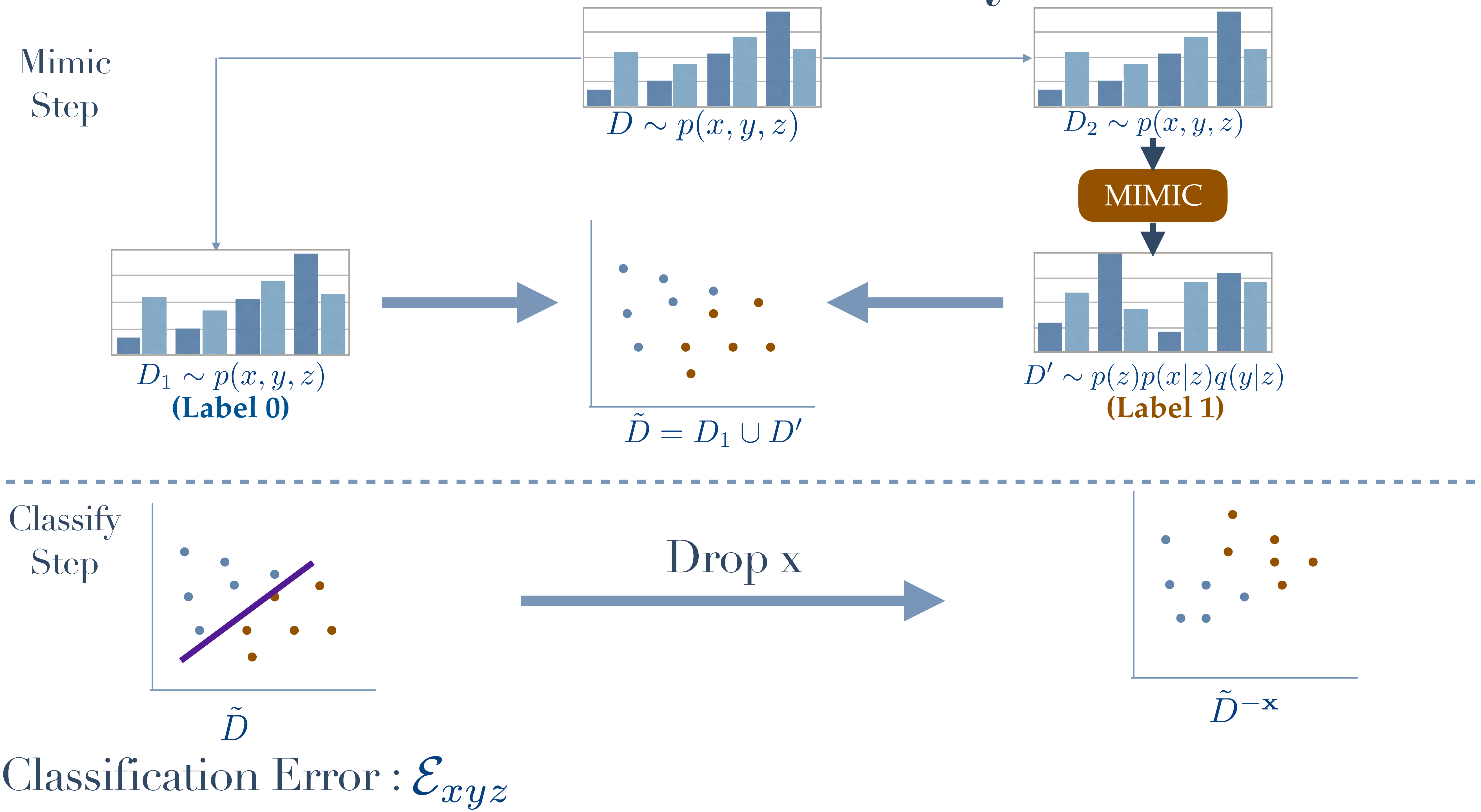
Mimic and Classify



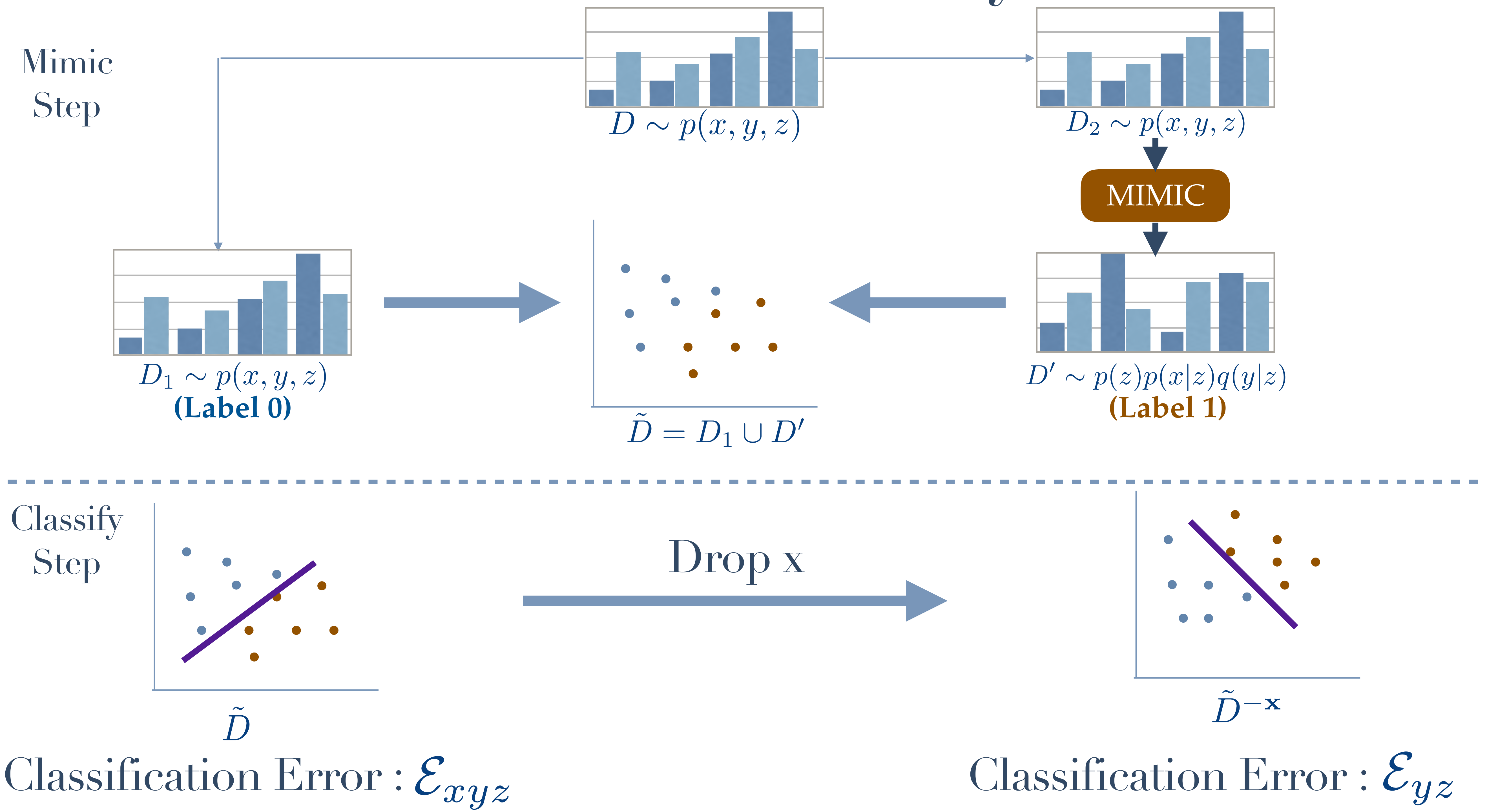


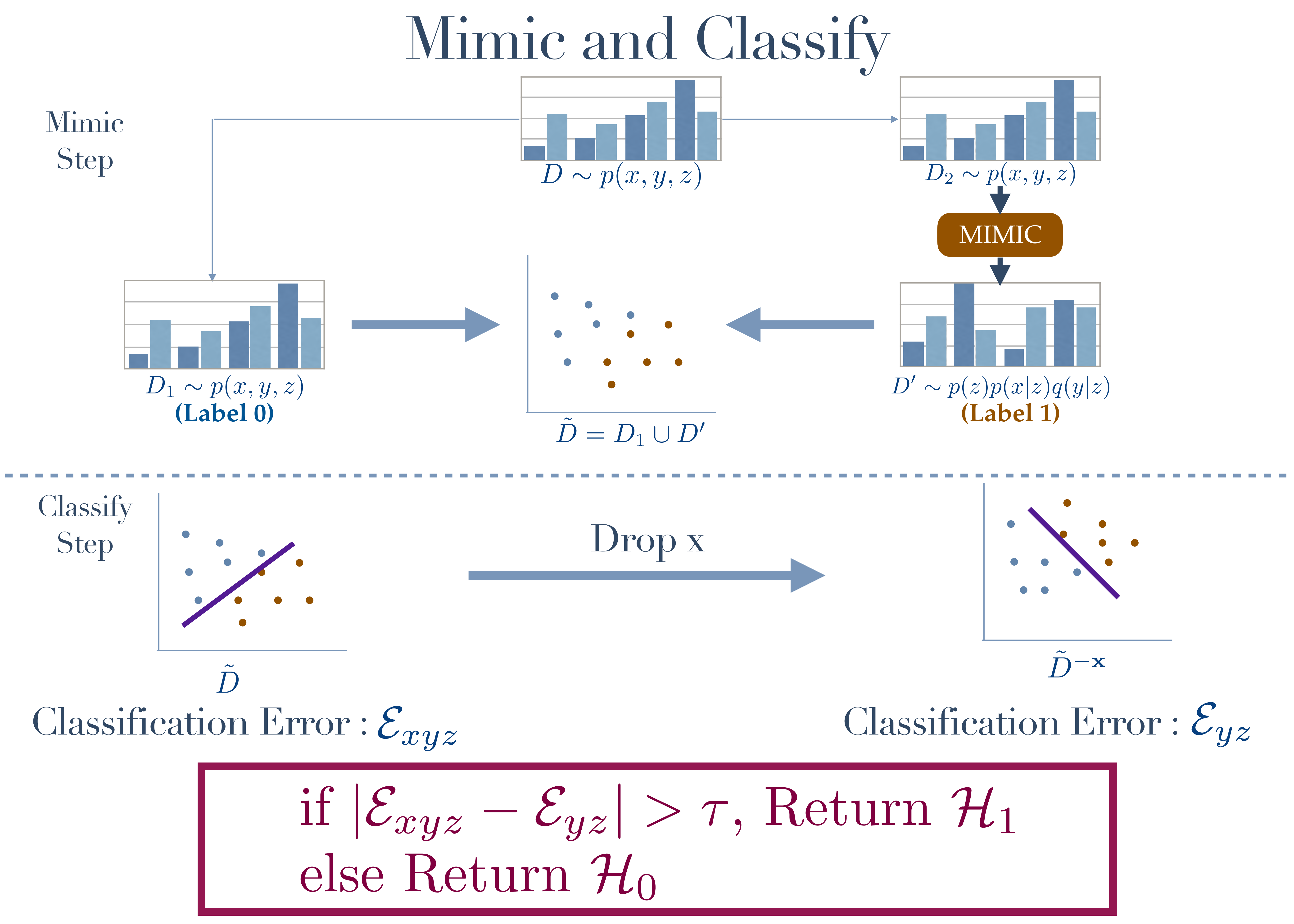
Classification Error : $\mathcal{E}_{xyz}$

Mimic and Classify



Mimic and Classify





Mimic and Classify
Mimic Step
$D \sim p(x, y, z)$
$D_2 \sim p(x, y, z)$
MIMIC
$D_1 \sim p(x, y, z)$
(Label 0)
$\tilde{D} = D_1 \cup D'$
$D' \sim p(z)p(x|z)q(y|z)$
(Label 1)
Classify Step
Drop x
$\tilde{D}$
$\tilde{D}^{-\mathbf{x}}$
Classification Error : $\mathcal{E}_{xyz}$
Classification Error : $\mathcal{E}_{yz}$
if $|\mathcal{E}_{xyz} - \mathcal{E}_{yz}| > \tau$, Return $\mathcal{H}_1$
else Return $\mathcal{H}_0$

Mimic and Classify

Mimic
Step

As long as the density function $q(\mathbf{y}|\mathbf{z}) > 0$ whenever $p(\mathbf{y}, \mathbf{z}) > 0$.

Classify
Step

Mimic and Classify

Mimic Step

As long as the density function $q(\mathbf{y}|\mathbf{z}) > 0$ whenever $p(\mathbf{y}, \mathbf{z}) > 0$.

Classify Step

$$|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| = 0 \leftrightarrow \mathcal{H}_0 \text{ is true}$$

$$\begin{aligned} &2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| \\ &= D_{\mathrm{TV}}(p(\mathbf{z}, \mathbf{x}, \mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y}, \mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) \end{aligned}$$

*The errors here are the corresponding optimal Bayes classifier errors.

Mimic and Classify (Theory)

$$2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]|$$
$$= D_{\mathrm{TV}}(p(\mathbf{z},\mathbf{x},\mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y},\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

Mimic and Classify (Theory)

$$
\begin{aligned}
&2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| \\
&= D_{\mathrm{TV}}(p(\mathbf{z},\mathbf{x},\mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y},\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) \\
&\geq \int_{\mathbf{y},\mathbf{z}} \min(p(\mathbf{z})q(\mathbf{y}|\mathbf{z}), p(\mathbf{z})p(\mathbf{y}|\mathbf{z}))(1-\epsilon(\mathbf{y},\mathbf{z}))d(\mathbf{y},\mathbf{z})
\end{aligned}
$$

Mimic and Classify (Theory)

$$
\begin{aligned}
&2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| \\
&= D_{\mathrm{TV}}(p(\mathbf{z},\mathbf{x},\mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y},\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) \\
&\geq \int_{\mathbf{y},\mathbf{z}} \min(p(\mathbf{z})q(\mathbf{y}|\mathbf{z}), p(\mathbf{z})p(\mathbf{y}|\mathbf{z}))(1-\epsilon(\mathbf{y},\mathbf{z}))d(\mathbf{y},\mathbf{z})
\end{aligned}
$$

Where: $\epsilon(\mathbf{y},\mathbf{z}) = \max_{\pi \in \Pi(p(\mathbf{x}|\mathbf{z}), p(\mathbf{x}'|\mathbf{y},\mathbf{z}))} \mathbb{E}_\pi[\mathbf{1}_{\{\mathbf{x}=\mathbf{x}'\}}|\mathbf{y},\mathbf{z}]$

Conditional dependence $\leftrightarrow \epsilon(y,z) < 1$ with non-zero probability

Mimic and Classify (Theory)

$$
\begin{aligned}
&2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| \\
&= D_{\mathrm{TV}}(p(\mathbf{z},\mathbf{x},\mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y},\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) \\
&\geq \int_{\mathbf{y},\mathbf{z}} \min(p(\mathbf{z})q(\mathbf{y}|\mathbf{z}), p(\mathbf{z})p(\mathbf{y}|\mathbf{z}))(1-\epsilon(\mathbf{y},\mathbf{z}))d(\mathbf{y},\mathbf{z})
\end{aligned}
$$

Where: $\epsilon(\mathbf{y},\mathbf{z}) = \max\limits_{\pi\in\Pi(p(\mathbf{x}|\mathbf{z}),p(\mathbf{x}'|\mathbf{y},\mathbf{z}))} \mathbb{E}_\pi[\mathbf{1}_{\{\mathbf{x}=\mathbf{x}'\}}|\mathbf{y},\mathbf{z}]$

Conditional dependence $\leftrightarrow \epsilon(y,z) < 1$ with non-zero probability

Theorem 1

As long as the density function $q(\mathbf{y}|\mathbf{z}) > 0$ whenever $p(\mathbf{y},\mathbf{z}) > 0$, then conditional dependence implies that $2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| > 0$

Mimic and Classify (Theory)

Conditional independence implies $p(\mathbf{x},\mathbf{y},\mathbf{z}) = p(\mathbf{z})p(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})$.

$$D_{\mathrm{TV}}(p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) = D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

Mimic and Classify (Theory)

Conditional independence implies $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})$.

$$D_{\mathrm{TV}}(p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) = D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]|$$
$$= D_{\mathrm{TV}}(p(\mathbf{z}, \mathbf{x}, \mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y}, \mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

Mimic and Classify (Theory)

Conditional independence implies $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})$.

$$D_{\mathrm{TV}}(p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) = D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]|$$

$$= D_{\mathrm{TV}}(p(\mathbf{z}, \mathbf{x}, \mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y}, \mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$= D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

Mimic and Classify (Theory)

Conditional independence implies $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})$.

$$D_{\mathrm{TV}}(p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) = D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]|$$

$$= D_{\mathrm{TV}}(p(\mathbf{z}, \mathbf{x}, \mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y}, \mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$= D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$= D_{\mathrm{TV}}(p(\mathbf{x}, \mathbf{y}, \mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

Mimic and Classify (Theory)

Conditional independence implies $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})$.

$$D_{\mathrm{TV}}(p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})) = D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]|$$

$$= D_{\mathrm{TV}}(p(\mathbf{z}, \mathbf{x}, \mathbf{y}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z})p(\mathbf{x}|\mathbf{z})) - D_{\mathrm{TV}}(p(\mathbf{y}, \mathbf{z}), p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$= D_{\mathrm{TV}}(p(\mathbf{x}|\mathbf{z})p(\mathbf{z})p(\mathbf{y}|\mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

$$= D_{\mathrm{TV}}(p(\mathbf{x}, \mathbf{y}, \mathbf{z}), p(\mathbf{x}|\mathbf{z})p(\mathbf{z})q(\mathbf{y}|\mathbf{z}))$$

Theorem 2

Conditional independence implies that $2|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| = 0$

Mimic and Classify (Theory)

Combining Theorem 1 and Theorem 2

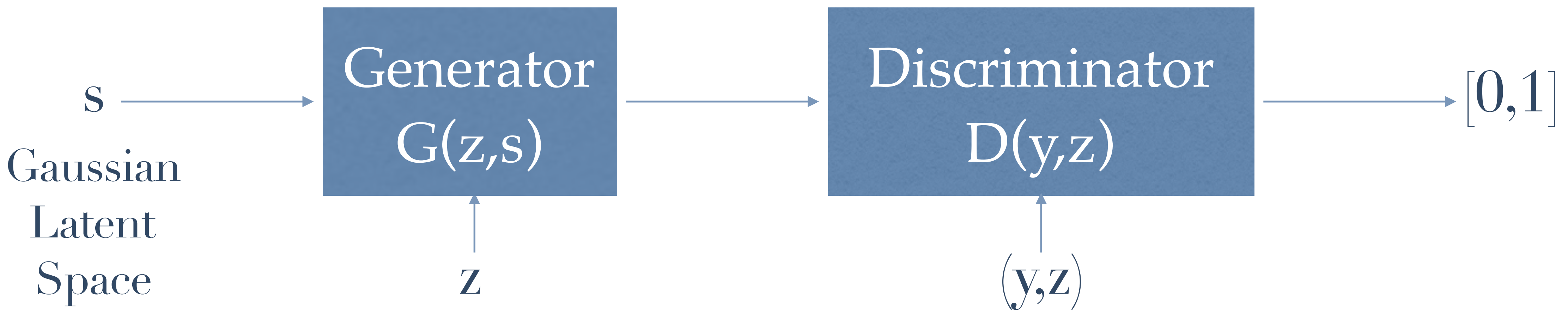
Theorem 3

As long as the density function $q(y|z) > 0$ when $p(y, z) > 0$

$|\mathbf{E}_D[\mathcal{E}_{xyz}] - \mathbf{E}_D[\mathcal{E}_{yz}]| = 0 \leftrightarrow \mathcal{H}_0$ is true

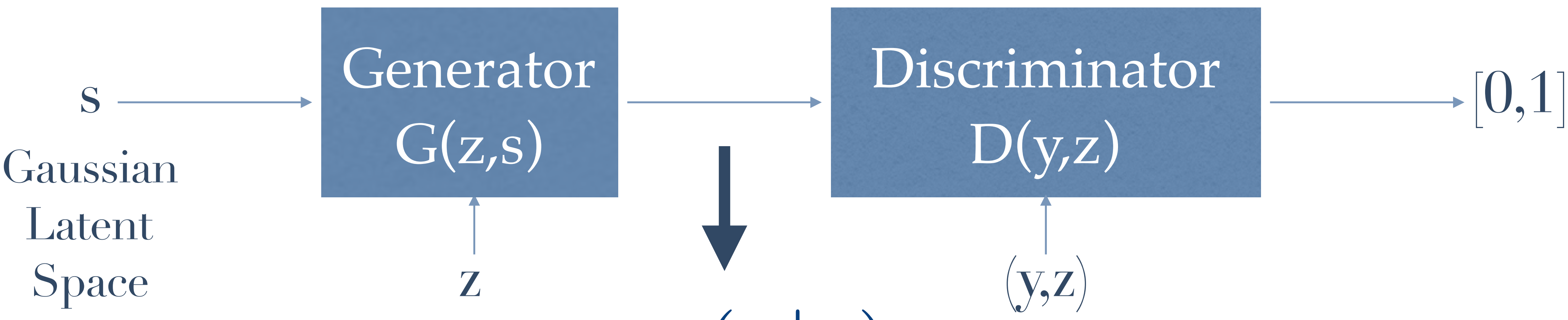
Deep Learning based MIMIC Functions

MIMIFY - CGAN



Deep Learning based MIMIC Functions

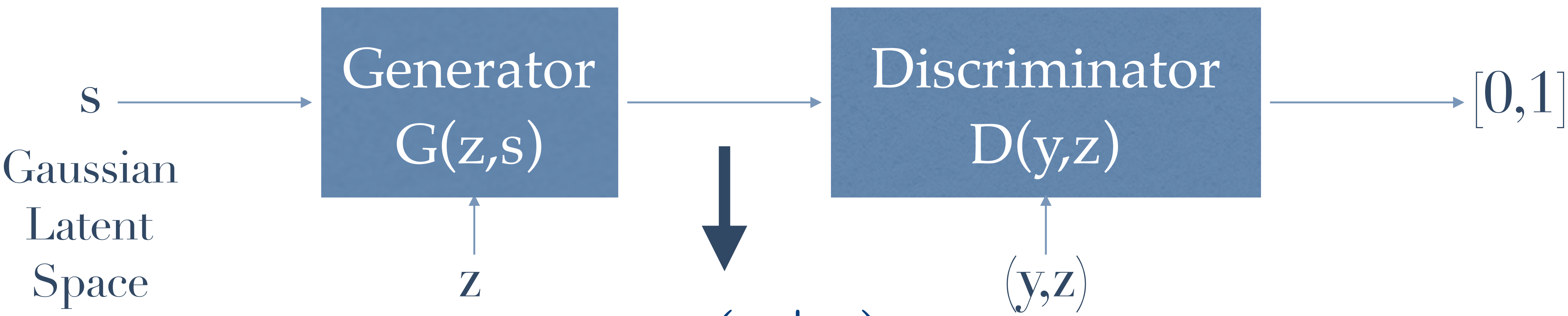
MIMIFY - CGAN



$$\sim q(y|z)$$

Deep Learning based MIMIC Functions

MIMIFY - CGAN



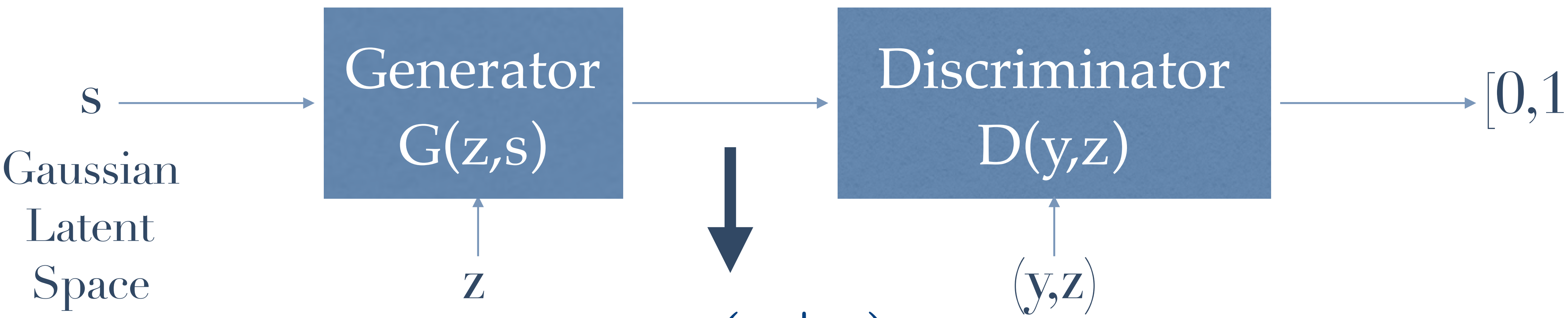
$$\sim q(y|z)$$

MIMIFY - REG

Regress to estimate $r(z) = \mathbf{E}[Y|Z = z]$

Deep Learning based MIMIC Functions

MIMIFY - CGAN



$$\sim q(y|z)$$

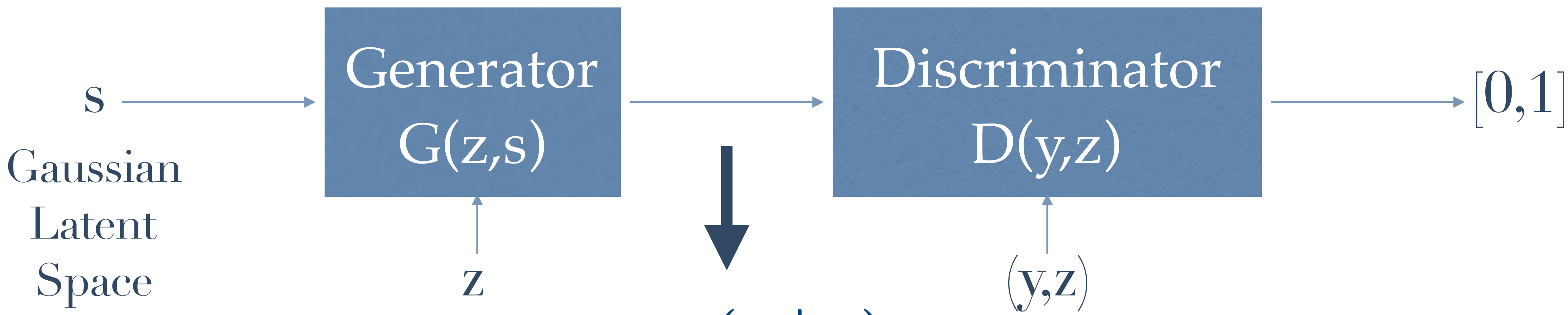
MIMIFY - REG

$$\text{Regress to estimate } r(z) = \mathbf{E}[Y|Z=z]$$

$$\hat{y} = r(z) + \text{ Gaussian Noise } \quad \sim q(y|z)$$

Deep Learning based MIMIC Functions

MIMIFY - CGAN



$$\sim q(y|z)$$

MIMIFY - REG

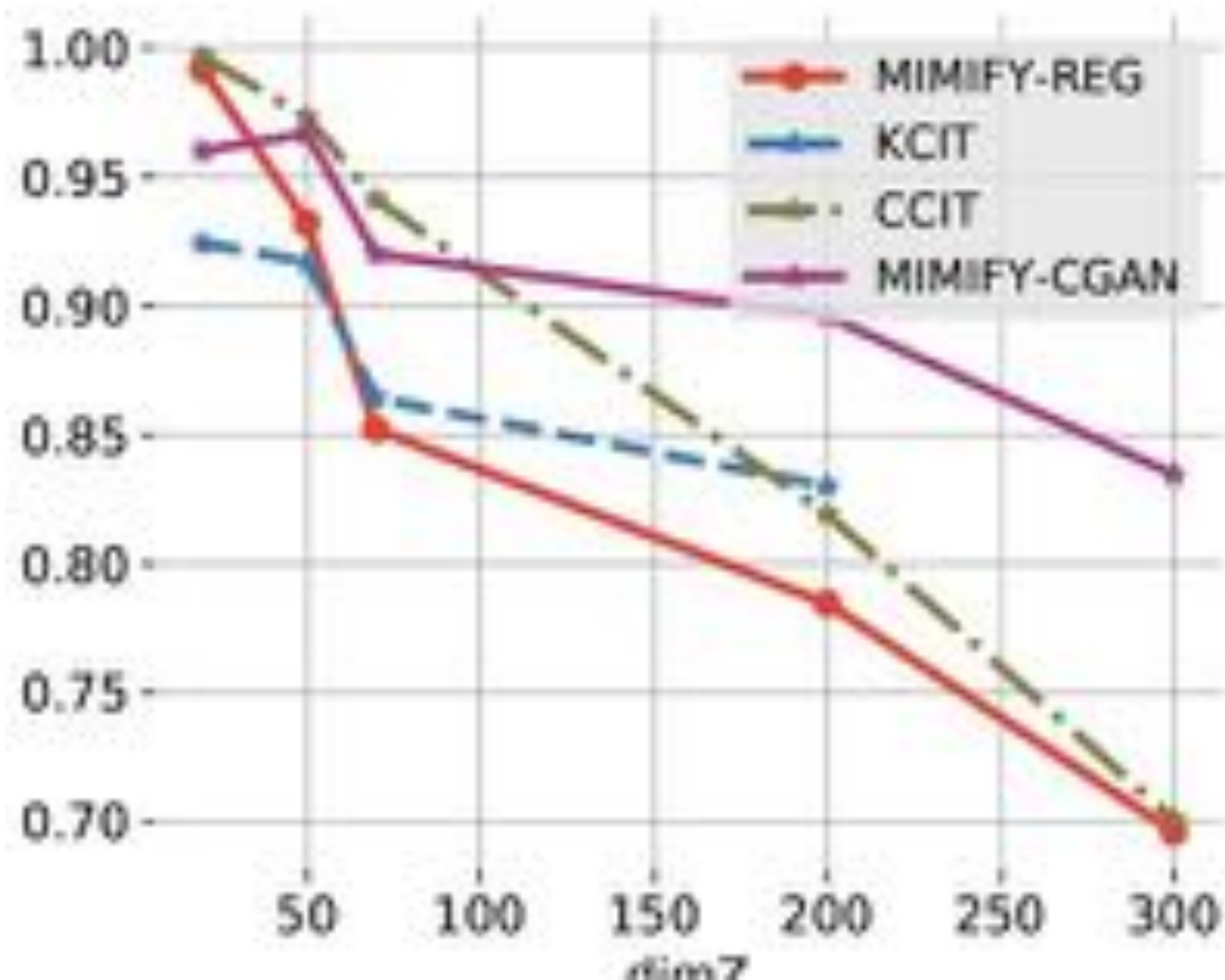
Regress to estimate $r(z) = \mathbf{E}[Y|Z = z]$

$\hat{y} = r(z) +$ Gaussian Noise $\sim q(y|z)$

(or, laplacian noise)

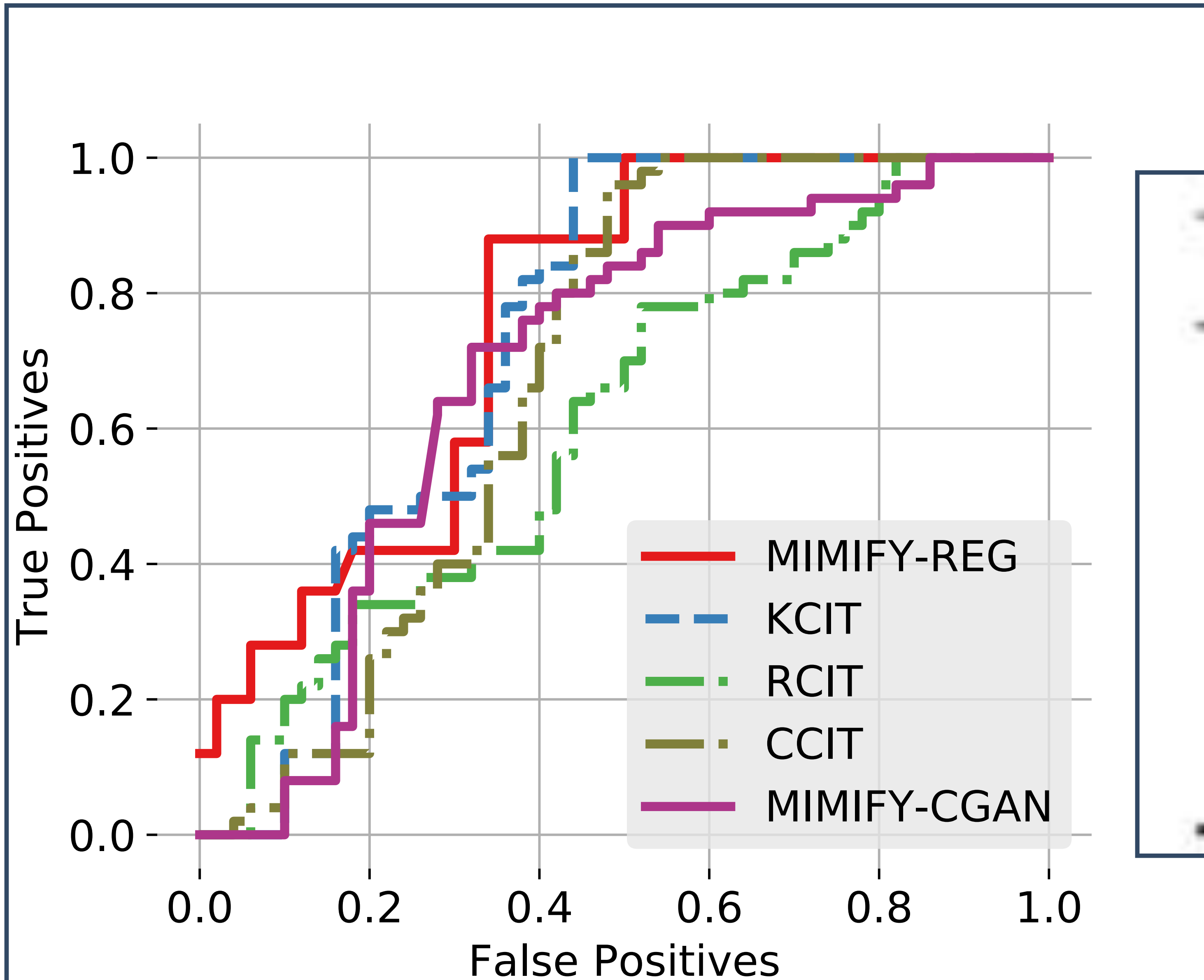
Experiments

Post-Nonlinear Noise Synthetic Experiments: AUROC



Experiments

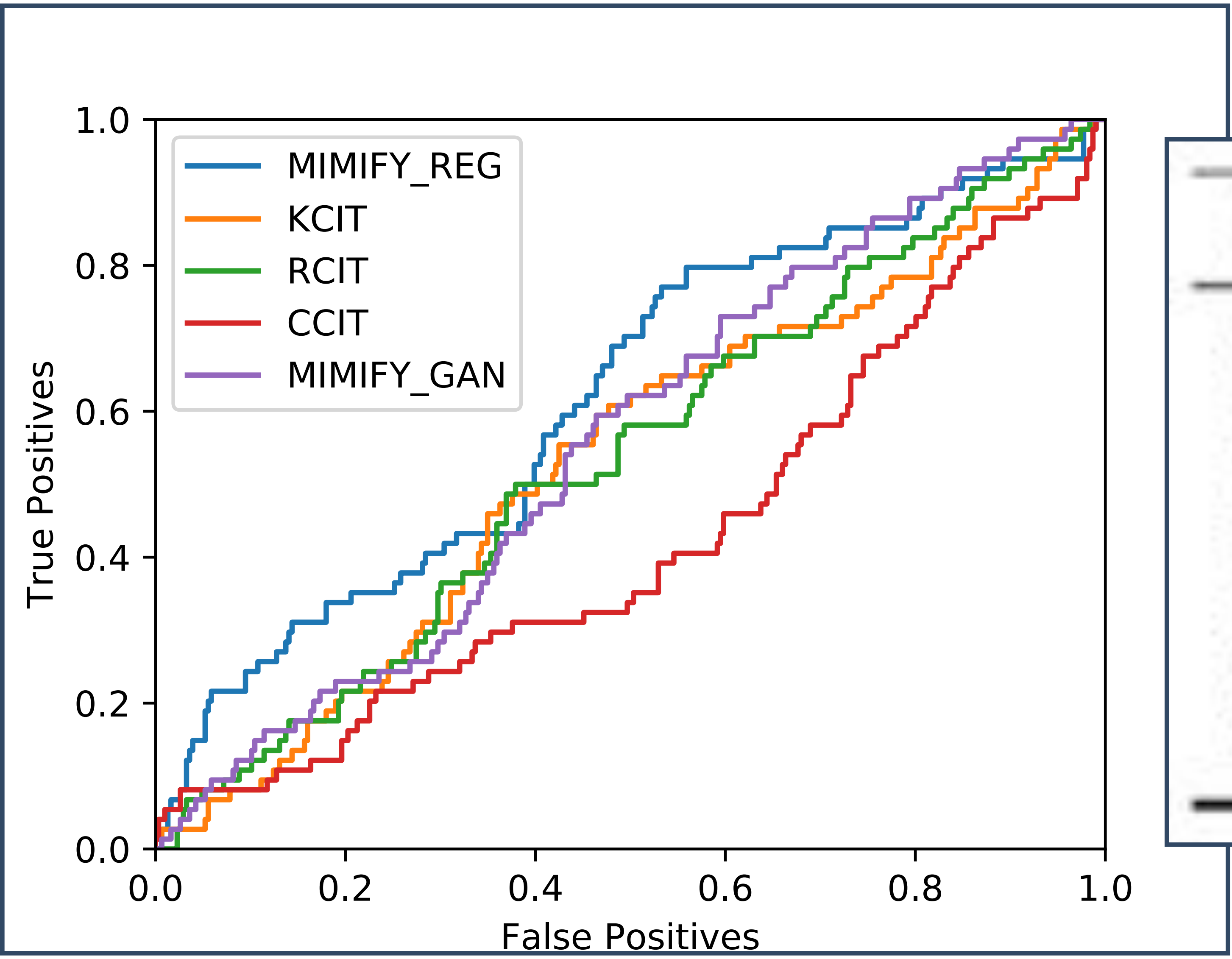
Flow-cytometry Data



Algo.	ROC-AUC
MIMIFY-REG	0.7638
KCIT	0.7328
MIMIFY-GAN	0.6891
CCIT	0.6816
RCIT	0.6135

Experiments

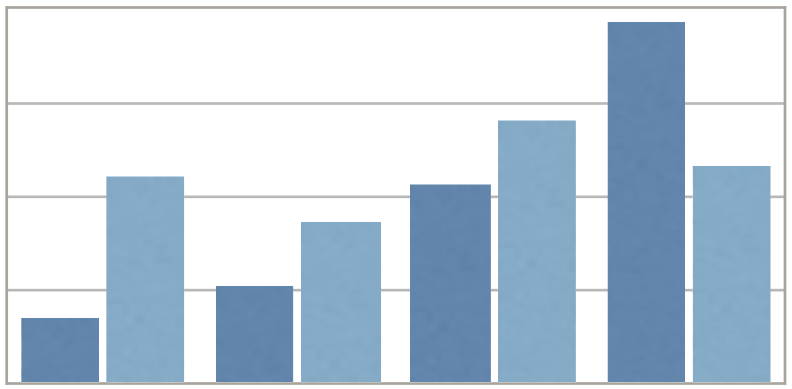
Gene Regulatory Network Inference (DREAM)



Algo.	ROC-AUC
MIMIFY-REG	0.61645
MIMIFY-GAN	0.55679
RCIT	0.53511
KCIT	0.53175
CCIT	0.42439

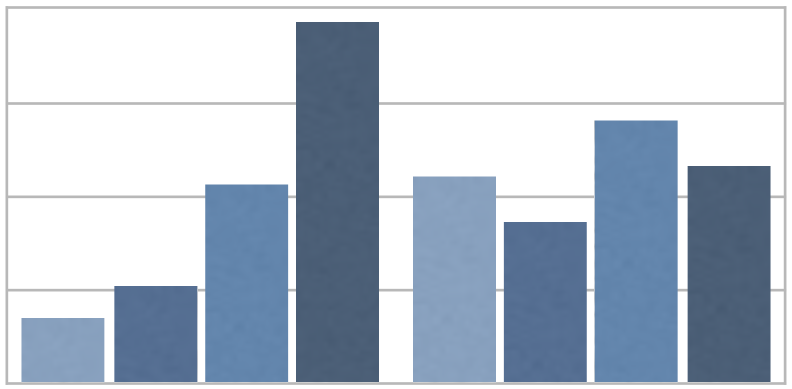
Estimating Information Measures

Estimating Kullback-Leibler Distance



P

n samples

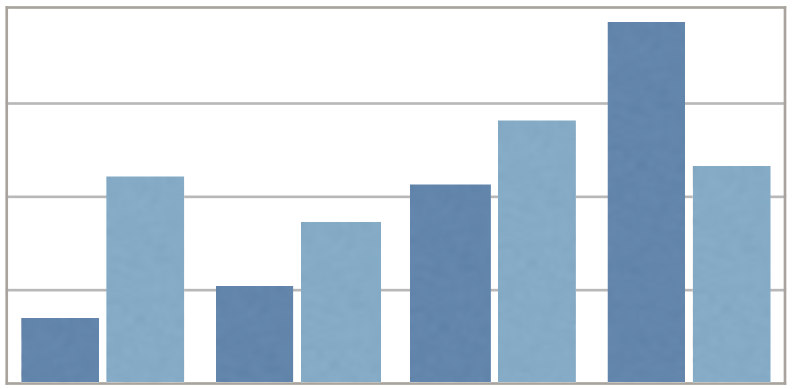


Q

n samples

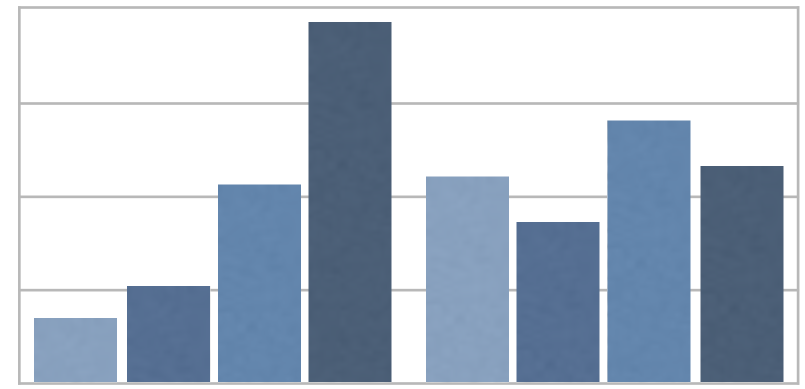
Estimate $D_{KL}(P \parallel Q)$?

Estimating Kullback-Leibler Distance



P

n samples



Q

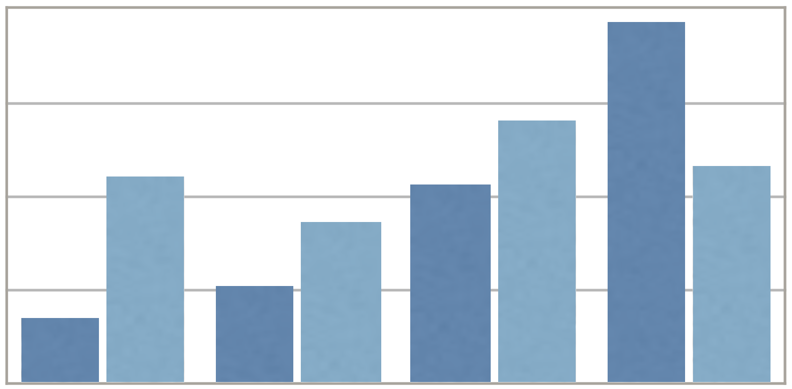
n samples

Estimate $D_{KL}(P \,\|\, Q)$?

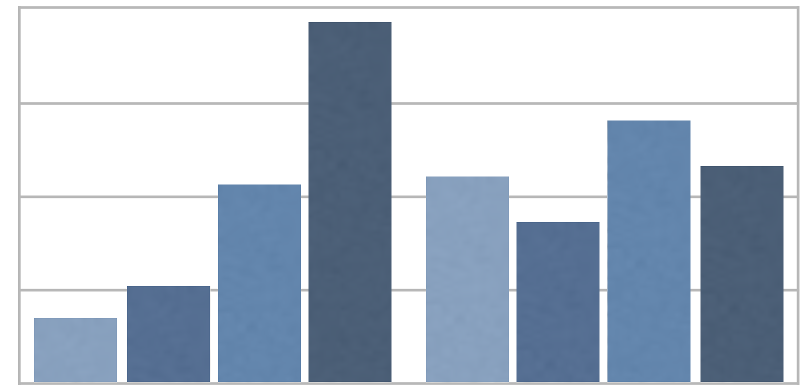
P and Q can be arbitrary.

Search beyond Traditional Density Estimation Methods

Neural Network Approximation

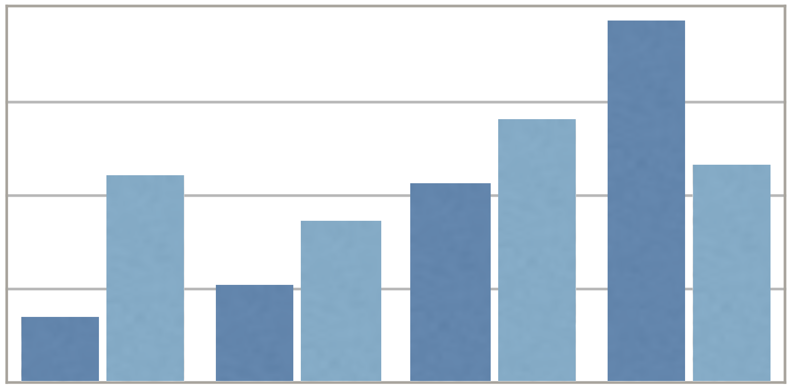


n samples $\sim$ P

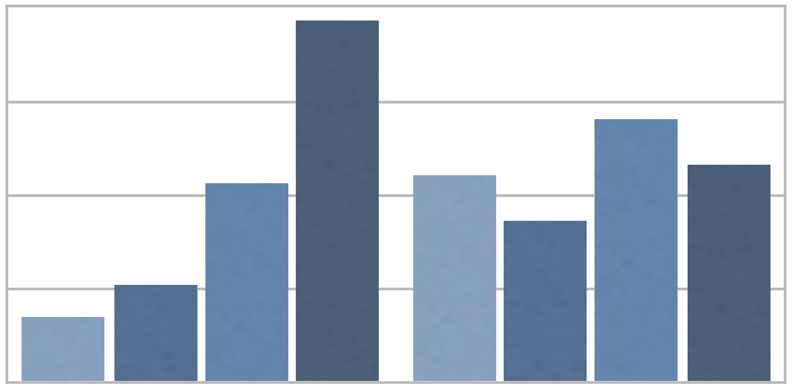


n samples $\sim$ Q

Neural Network Approximation



n samples ~ P

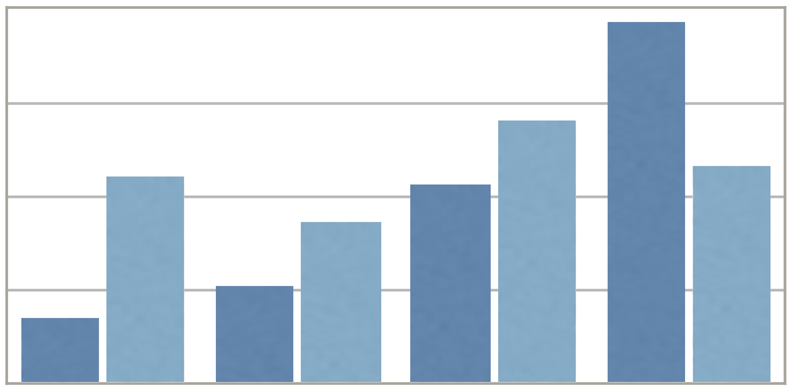


n samples ~ Q

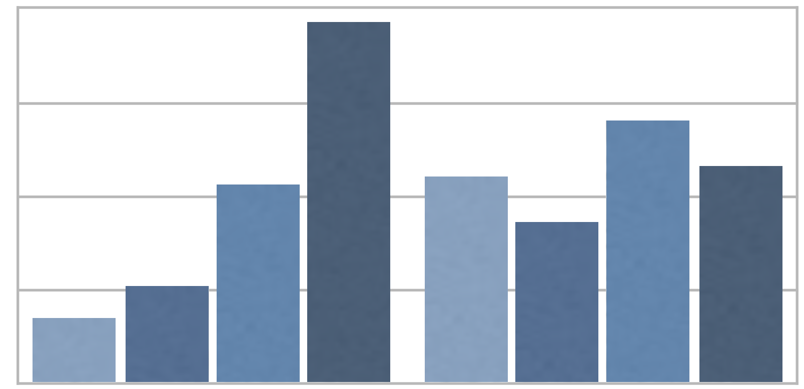
Donsker-Varadhan Dual Representation:

$$D_{KL}(P \parallel Q) = \sup_T \mathbf{E}_P[T] - \log(\mathbf{E}_Q[e^T])$$

Neural Network Approximation



n samples $\sim$ P



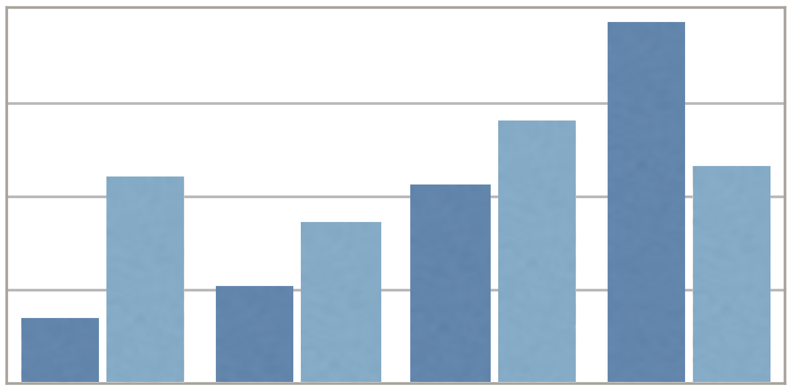
n samples $\sim$ Q

Donsker-Varadhan Dual Representation:

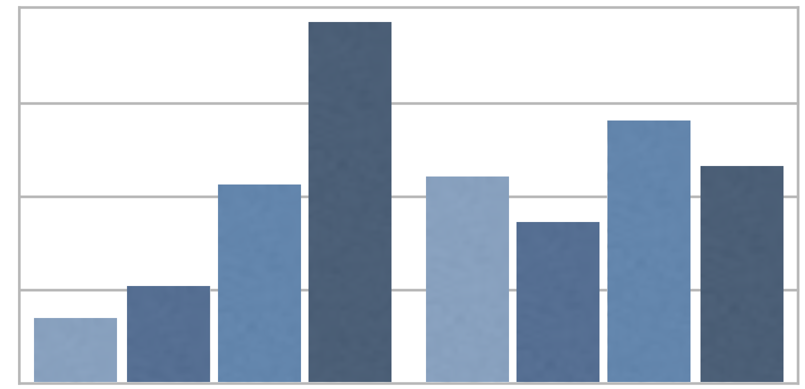
$$D_{KL}(P \parallel Q) = \sup_T \mathbf{E}_P[T] - \log(\mathbf{E}_Q[e^T])$$

- $T \leftarrow$ Rich NN class
- $\mathbf{E} \leftarrow$ Sample Averages
- $\sup_T \leftarrow$ Obtained via Stochastic Gradient search

Mutual Information Neural Estimation (MINE)



n samples $\sim$ P



n samples $\sim$ Q

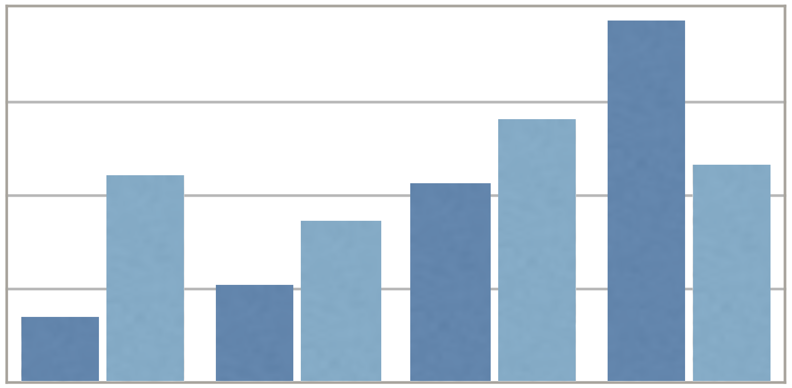
Donsker-Varadhan Dual Representation:

$$D_{KL}(P \parallel Q) = \sup_T \mathbf{E}_P[T] - \log(\mathbf{E}_Q[e^T])$$

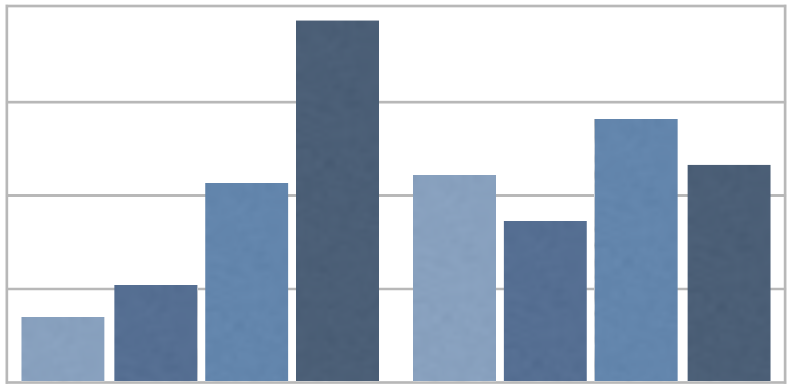
$$I(X;Y) = D_{KL}(\mathbf{P}_{XY} \parallel \mathbf{P}_X \mathbf{P}_Y)$$

*Benghazi et al, **MINE : Mutual Information Neural Estimation**, *ICML 2018*

Mutual Information Neural Estimation (MINE)



n samples $\sim$ P



n samples $\sim$ Q

Donsker-Varadhan Dual Representation:

$$D_{KL}(P \parallel Q) = \sup_T \mathbf{E}_P[T] - \log(\mathbf{E}_Q[e^T])$$

$$I(X;Y) = D_{KL}(\mathbf{P}_{XY} \parallel \mathbf{P}_X \mathbf{P}_Y)$$

Generated via Permutation

*Benghazi et al, **MINE : Mutual Information Neural Estimation**, *ICML 2018*

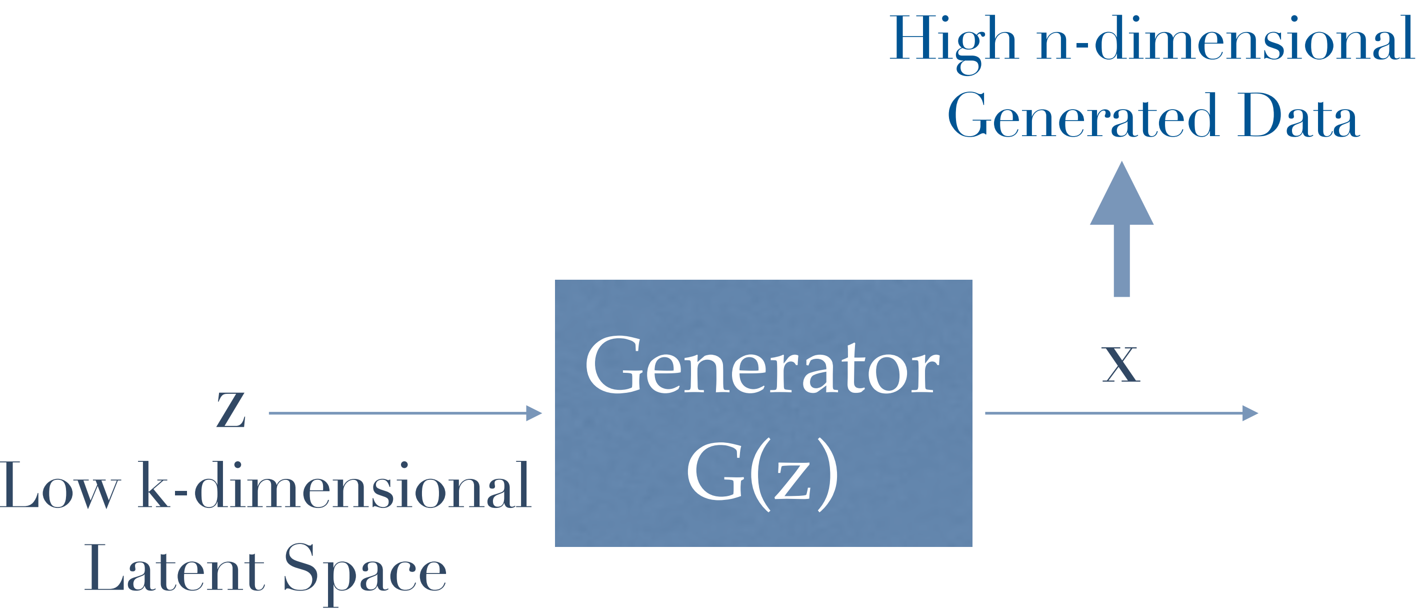
Compressed Sensing

Generative Model and Linear Measurements

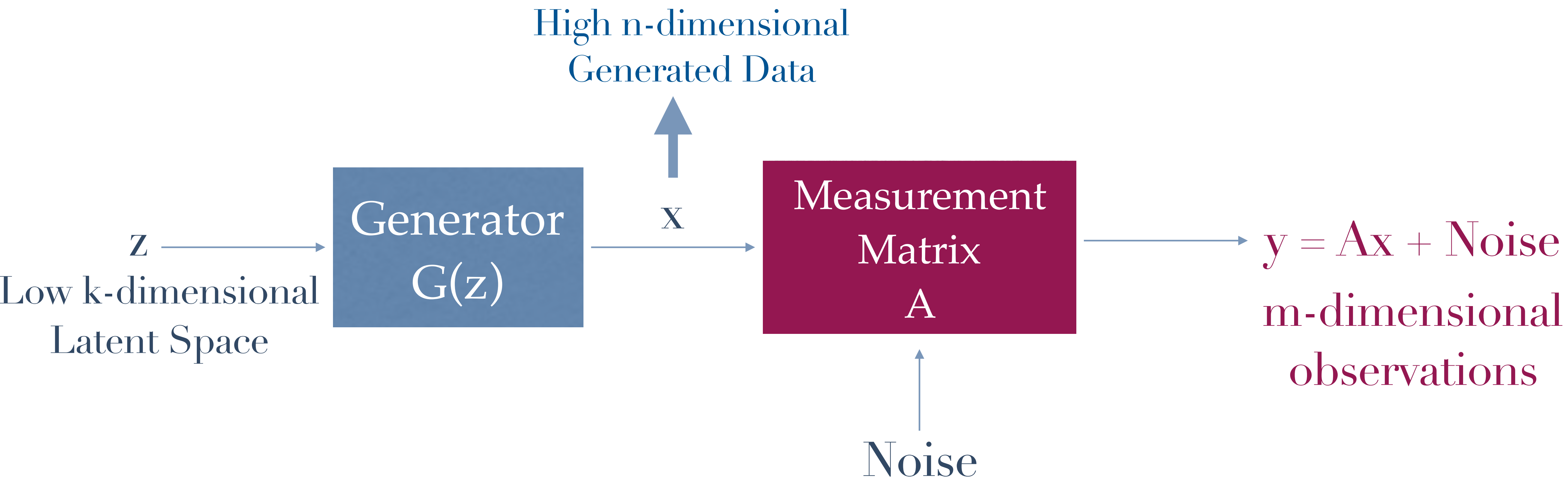
z

Low k-dimensional
Latent Space

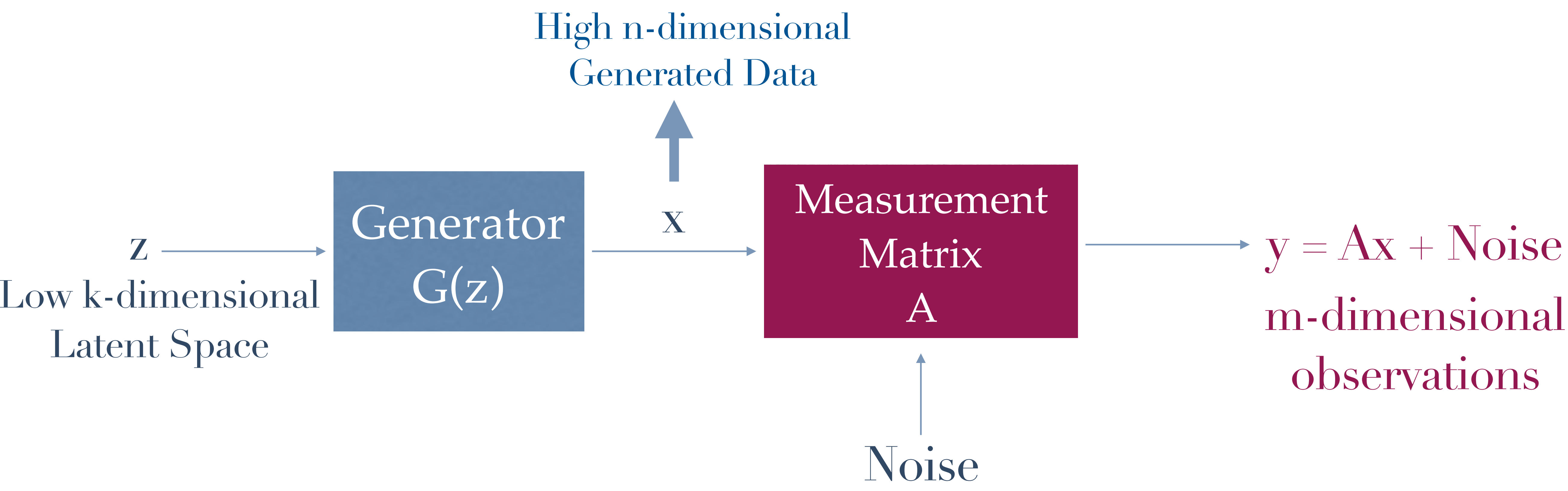
Generative Model and Linear Measurements



Generative Model and Linear Measurements

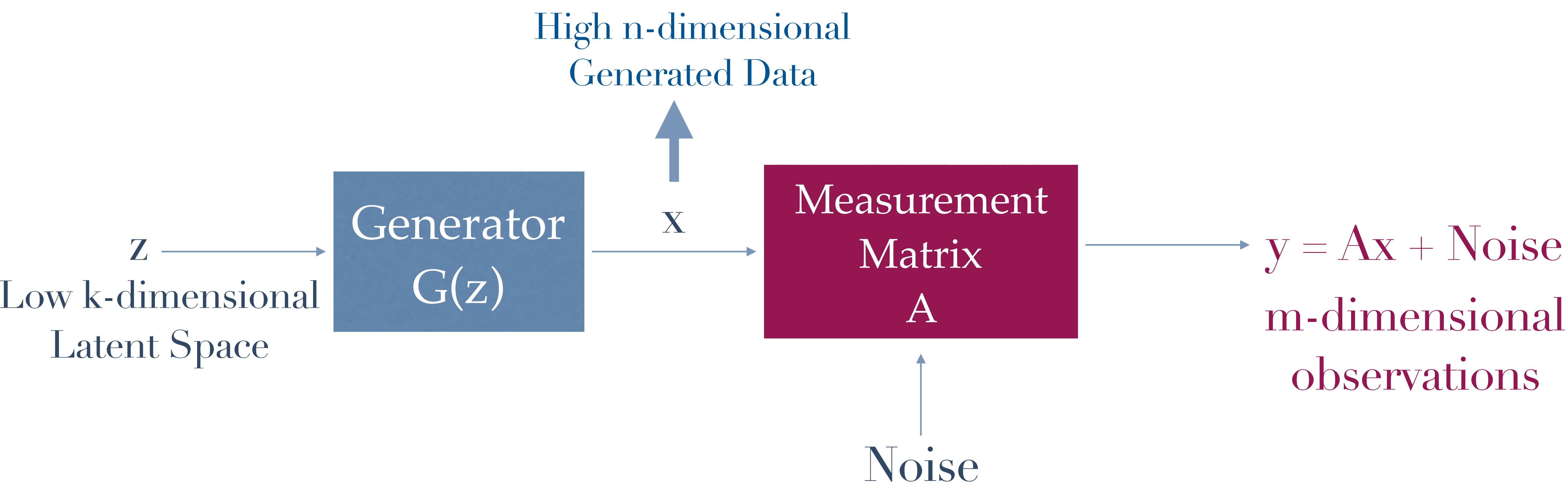


Generative Model and Linear Measurements



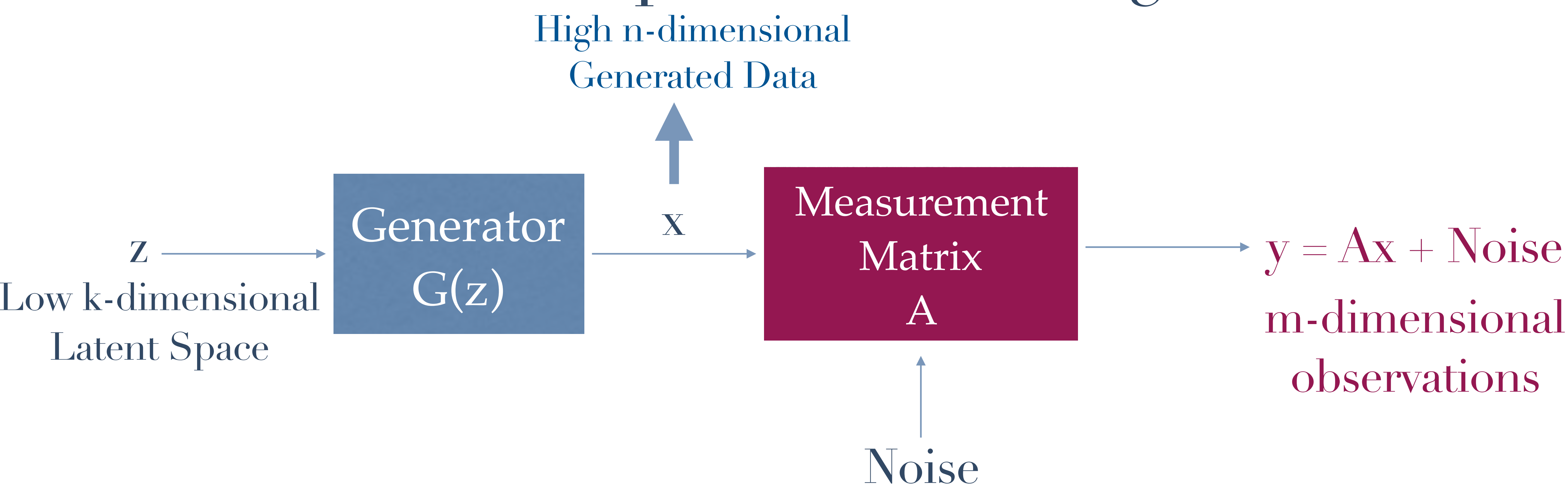
Given y : Guess x?

Generative Model and Linear Measurements



How large is m (#measurements)?

Compressed Sensing



$A =$ scaled Gaussian Random Matrix, $G = d$-layer NN then, $m = O(kd \log n)$ suffice.

*Bora et al, **Compressed Sensing using Generative Models**, *ICML 2017*

*Yeh et al, **Semantic Image Inpainting with Deep Generative Models**, *CVPR 2017*

*Bora et al, **AmbientGAN: Generative models from lossy measurements**, *ICLR 2018*

Open Problems

- Statistical property testing and estimation problems
  - Beyond DTV: Distance measure estimation using classifier.
  - Time-series data (Directed information estimation and testing).
- Information bottleneck and deep learning
  - Relationship hotly disputed. Need strong MI estimators!
- Conditional mutual information estimation
  - Plays vital role in controlling bias or privacy
    - I(Salary ; Race | Performance) small
- Rely on GAN based generative models
  - Does not work well in small sample regime
  - Need for Unified framework

Part 2A.
Applications of (Information) Theory to Generative Adversarial Networks

Sewoong Oh

University of Illinois at Urbana-Champaign

Organization: This Tutorial

Part-1: Deep learning for information theory

1a. Deep learning for communication

1b. Deep learning for statistical inference

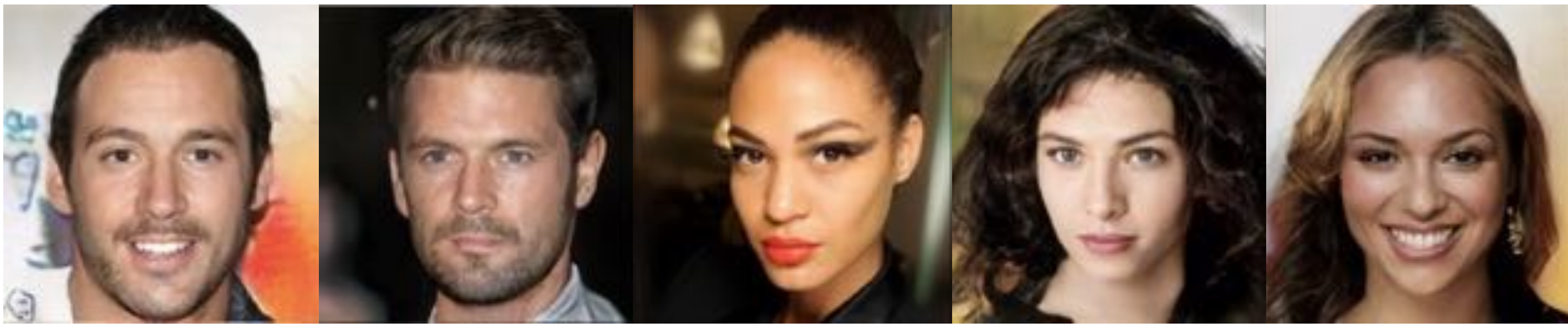
Part-2: Information theory for deep learning

2a. Theory for GAN

2b. Learning Gated Neural Networks

Neural network generative models

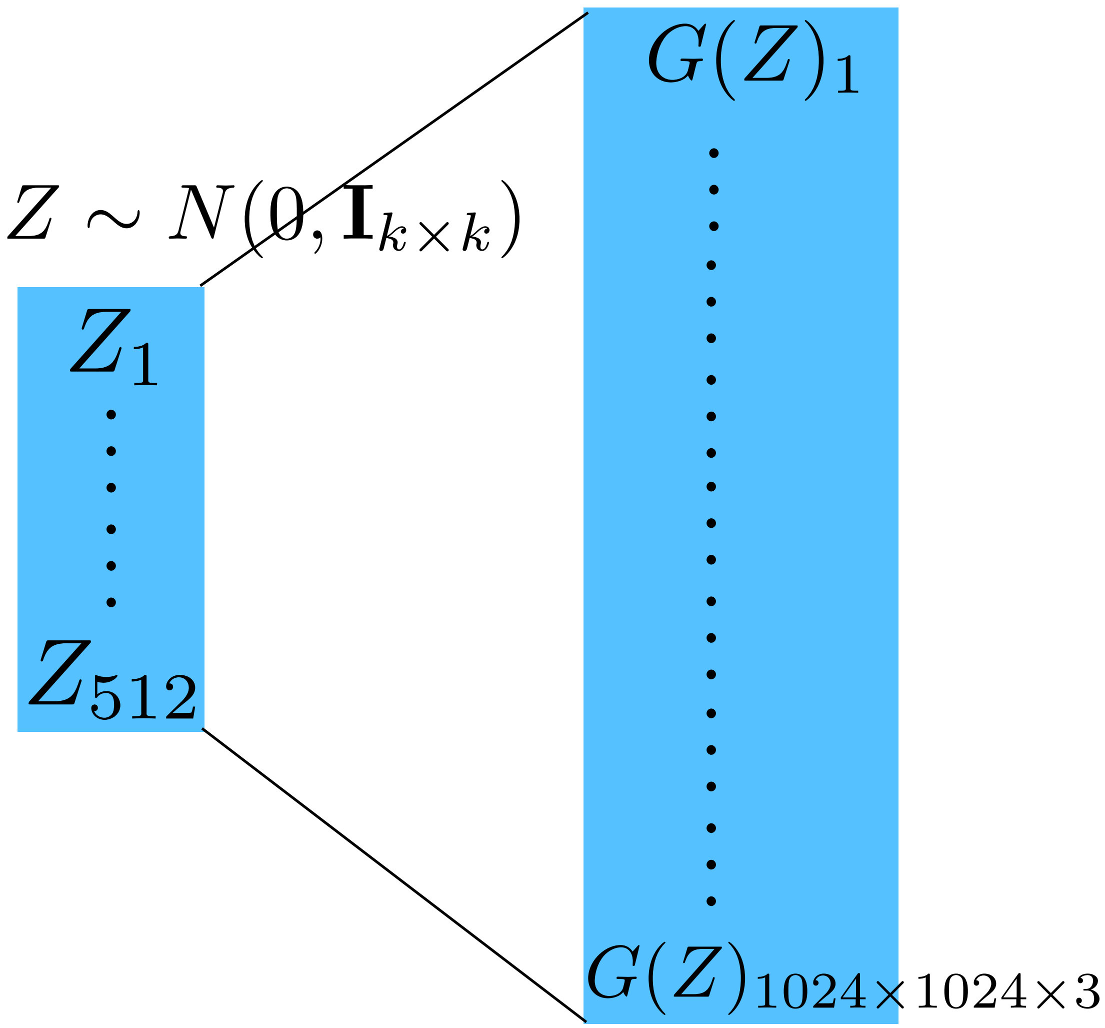
- How do we model the distribution of complex data in high-dimensions?



- Parametric models (e.g. mixture of Gaussians) fail on complex data

- Non-parametric models (e.g. KDE, Nearest Neighbor) fail in high dimensions

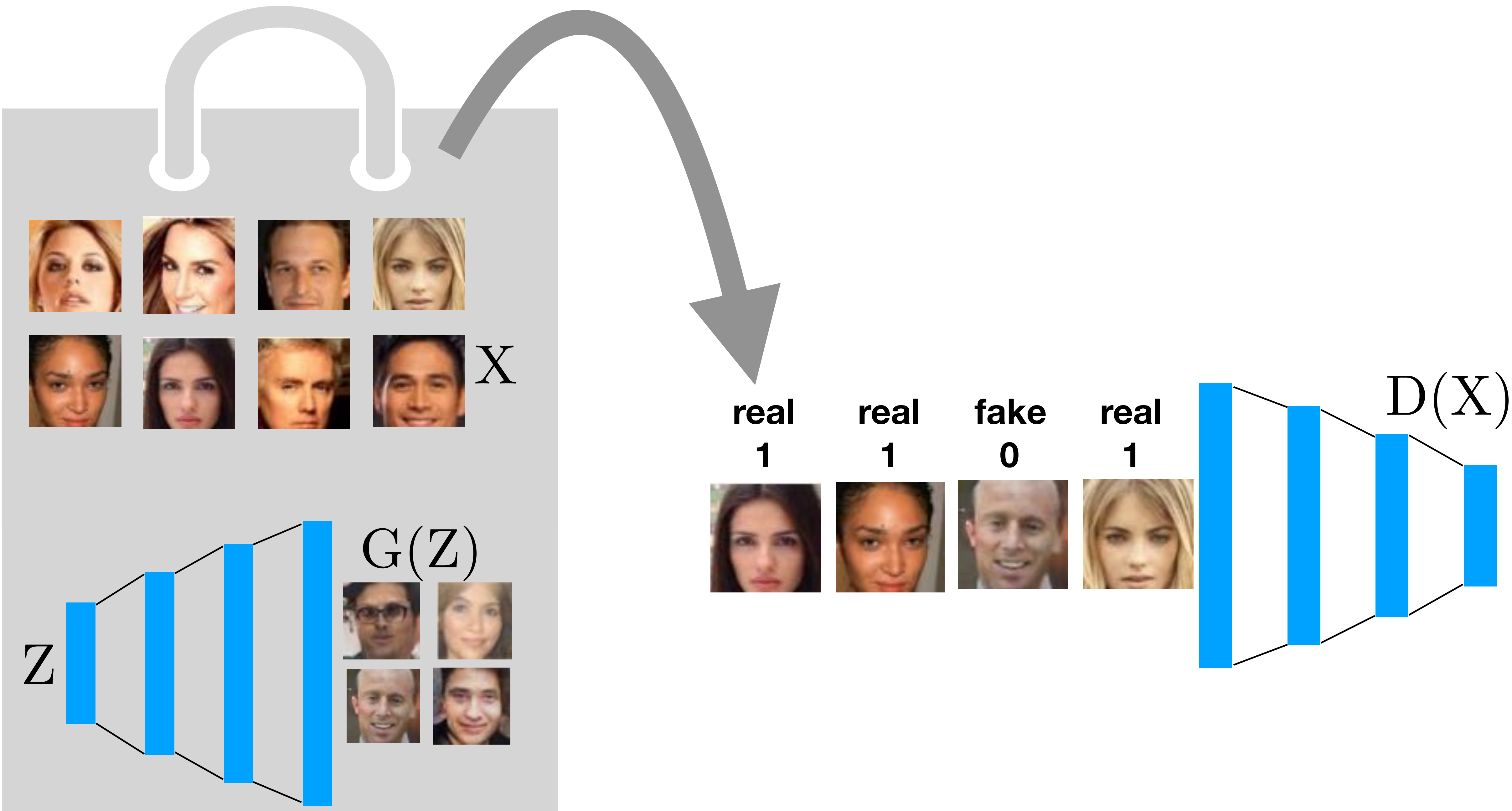
Neural network generative models



$G(Z) \in \mathbb{R}^{1024 \times 1024 \times 3}$

- A generative model takes a random vector Z and produces samples G(Z)
- The neural network weights can be trained by gradient descent

Generative Adversarial Network



$$\min_G \max_D V(G, D)$$

["Generative adversarial nets",Goodfellow et al., 2014]

Generative Adversarial Network

- GAN loss choices
  - Cross-Entropy loss

$$\min_G \max_D \mathbb{E}_{P_{\text{real}}}[\log(D(X))] + \mathbb{E}_{Q_G}[\log(1 - D(X))]$$

$$D^*(X) = \frac{P_{\text{real}}(X)}{P_{\text{real}}(X) + Q_G(X)}$$

$$\min_G \; 2\, D_{\text{JS}}(P_{\text{real}} \| Q_G) - \log 4$$

$$D_{\text{JS}}(P\|Q) \;=\; \frac{1}{2} D_{\text{KL}}\Big(P\|\frac{P+Q}{2}\Big) + \frac{1}{2} D_{\text{KL}}\Big(Q\|\frac{P+Q}{2}\Big)$$

[“Generative adversarial nets”,Goodfellow et al., 2014]

Generative Adversarial Network

- GAN loss choices
  - 0-1 loss

$$\min_{G} \quad \max_{D} \quad \mathbb{E}_{P_{\text{real}}}[D(X)] - \mathbb{E}_{Q_G}[D(X)]$$

$$D^*(X) \quad = \quad \mathbb{I}\{P_{\text{real}}(X) > Q_G(X)\}$$

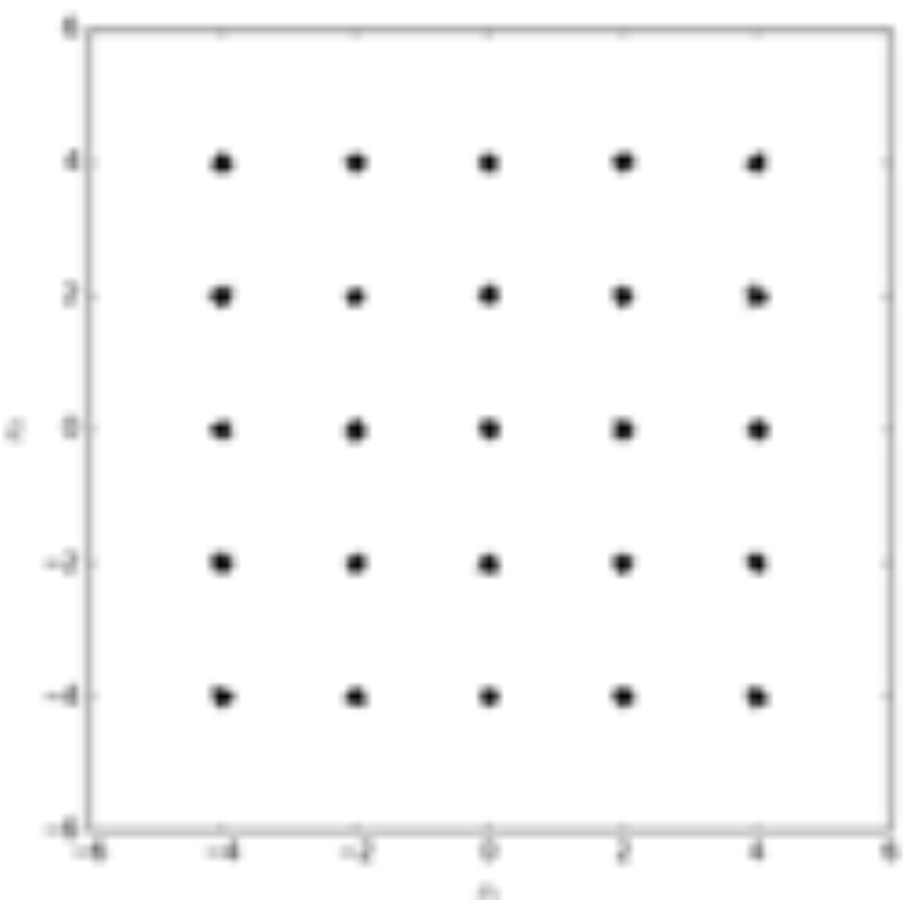
$$\min_{G} \quad d_{\text{TV}}(P_{\text{real}}, Q_G)$$

  - Other popular choices: f-divergence, Wasserstein distance

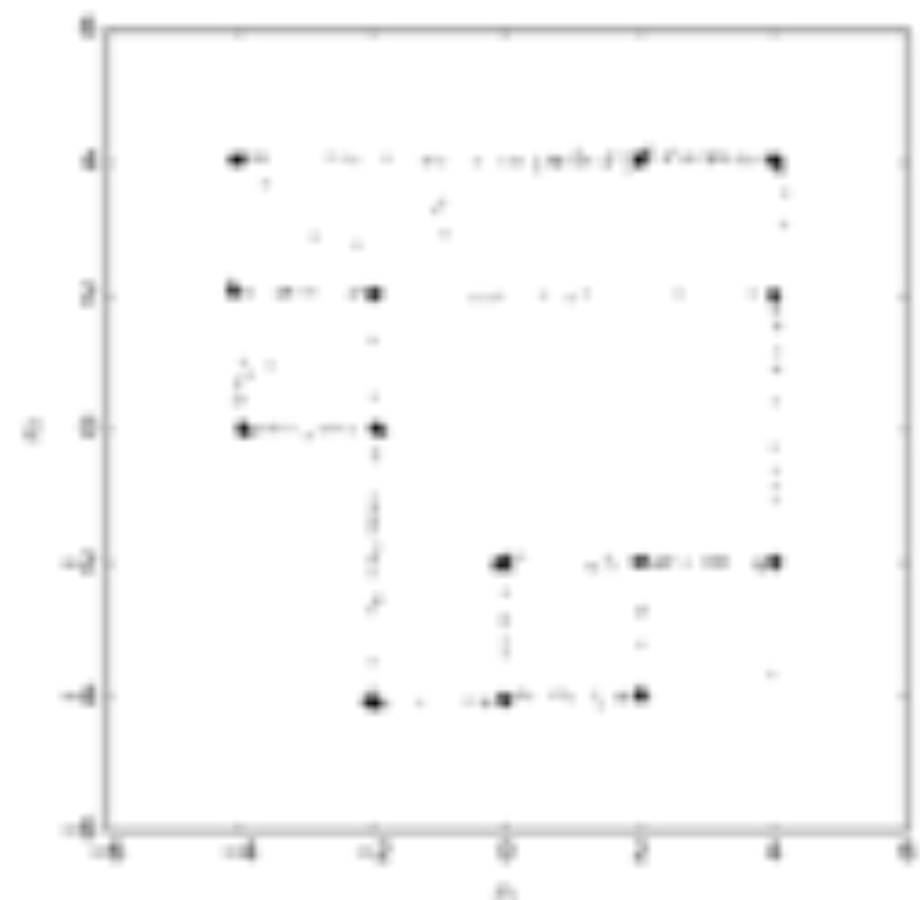
[“Wasserstein GAN”, Arjovsky,Chintala,Bottou,2017]
[“f-gan”Nowozin,Cseke,Tomioka,2016]

Mode Collapse is a major challenge in GAN

- **Mode Collapse** collectively refers to the lack of diversity in the generated samples



target distribution
mixture of 25 Gaussians in 2D

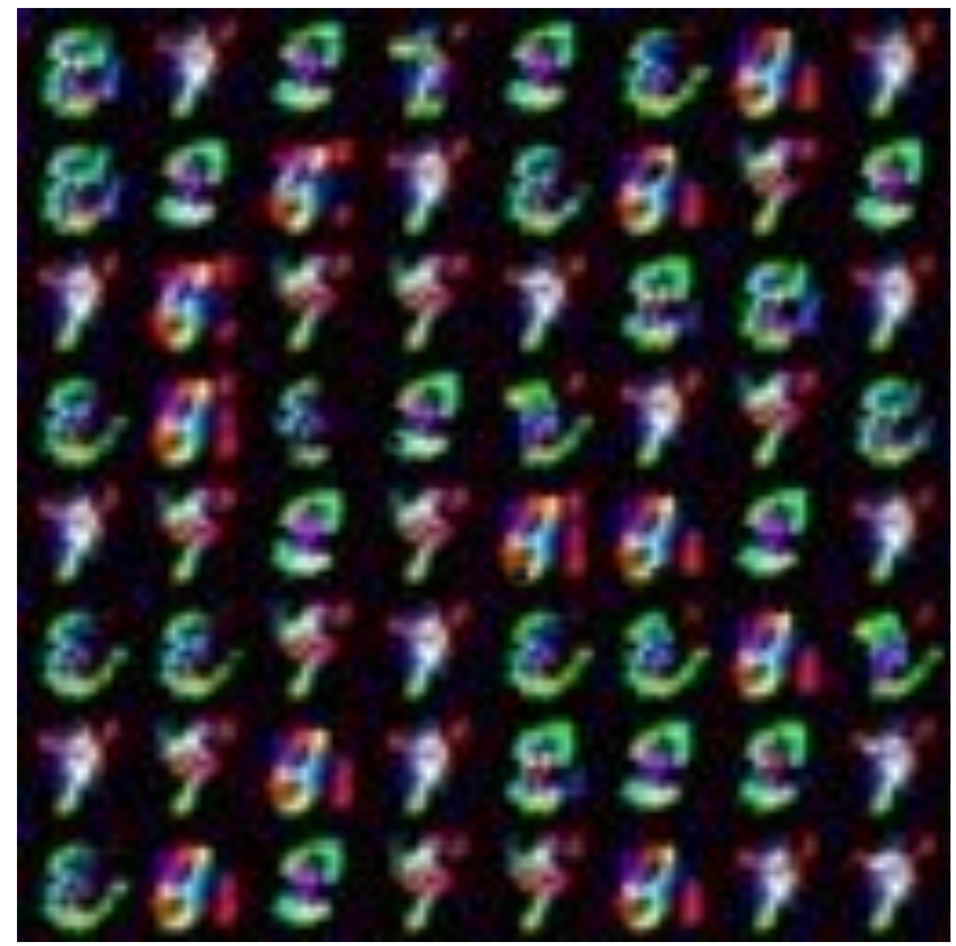


	Modes (Max 25)
GAN	17.3

Mode Collapse is a major challenge in GAN

- **Mode Collapse** collectively refers to the lack of diversity in the generated samples

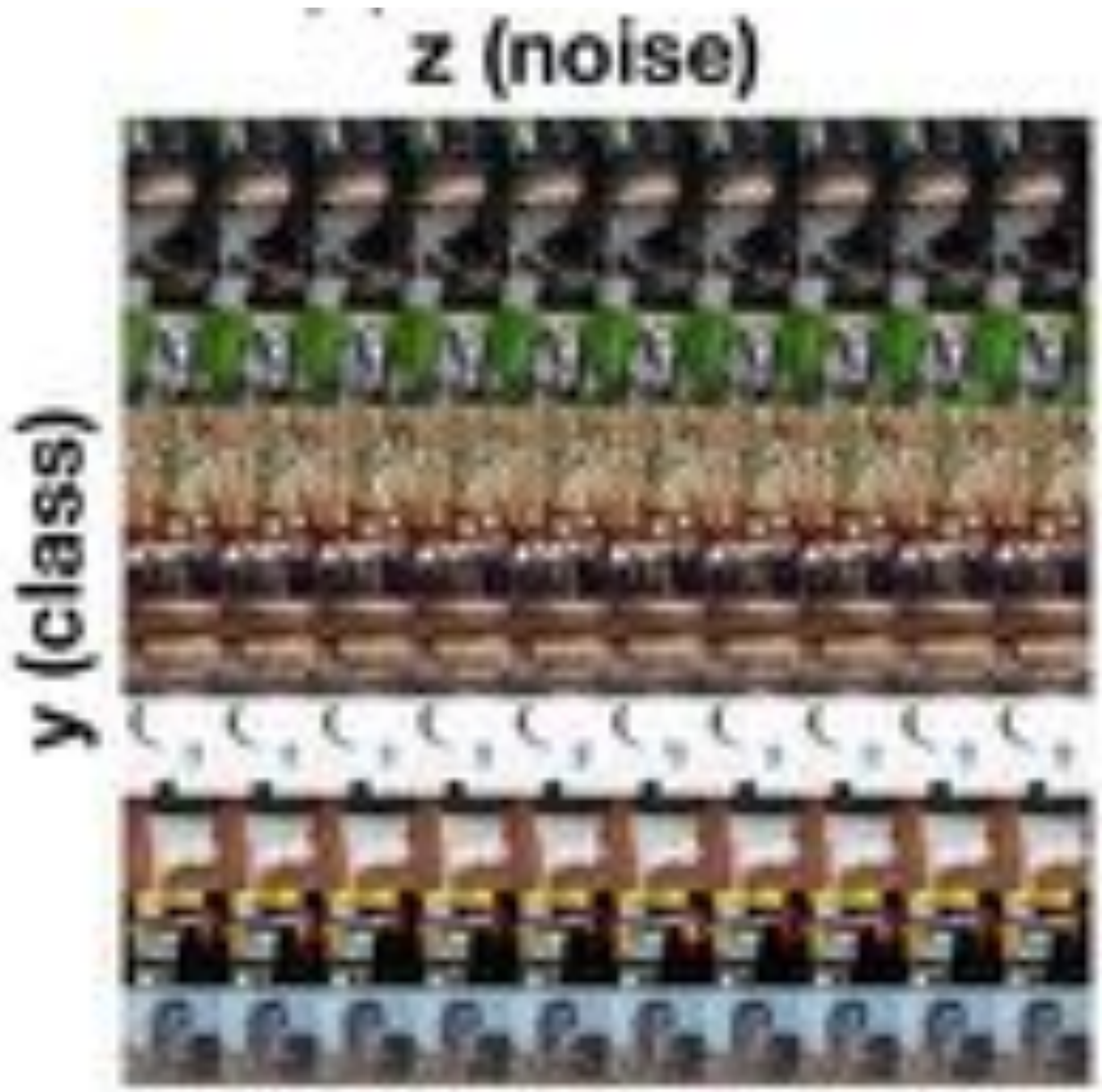
target distribution
Stacked MNIST



	Modes (Max 1000)
DCGAN	99.0

Mode Collapse is prevalent in real applications

- Heuristics tailored for each task (or dataset) don't generalize to new tasks



["Conditional image synthesis with auxiliary classifier GANs", Odena,Olah,Shlens, 2016]
["GANs with projection discriminator", Miyato,Koyama, 2018]

Mode Collapse is prevalent in real applications

- Heuristics provide varying levels of improvement, but **Mode Collapse** is a fundamental challenge
  - "A man in a orange jacket with sunglasses and a hat ski down a hill."

["Generating interpretable images with controllable structure", Reed et al., 2016]

(Detection) theoretical understanding of Mode Collapse

- Through the lens of **binary hypothesis testing**, we provide new formal definition of Mode Collapse

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \quad \text{and} \quad Q(S) \leq \varepsilon \ .$$

- The 2-D region representation
  - allows formal comparison of strengths of Mode Collapse
  - read off all divergences
  - intuition on how to understand adversarial training
  - new architecture for GAN
  - new proof technique to prove our main results

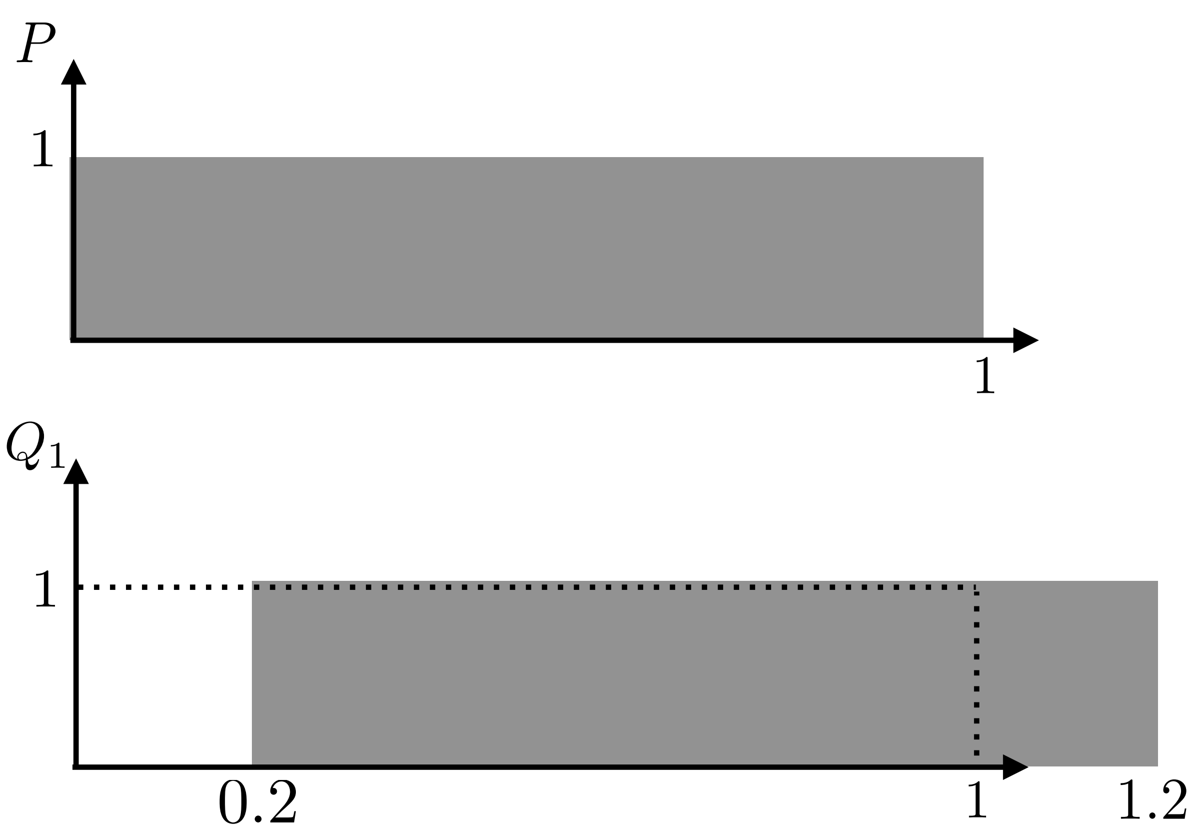
["PacGAN: the power oft samples in generative adversarial networks",Lin,Khetan,Oh,Fanti, 2017]

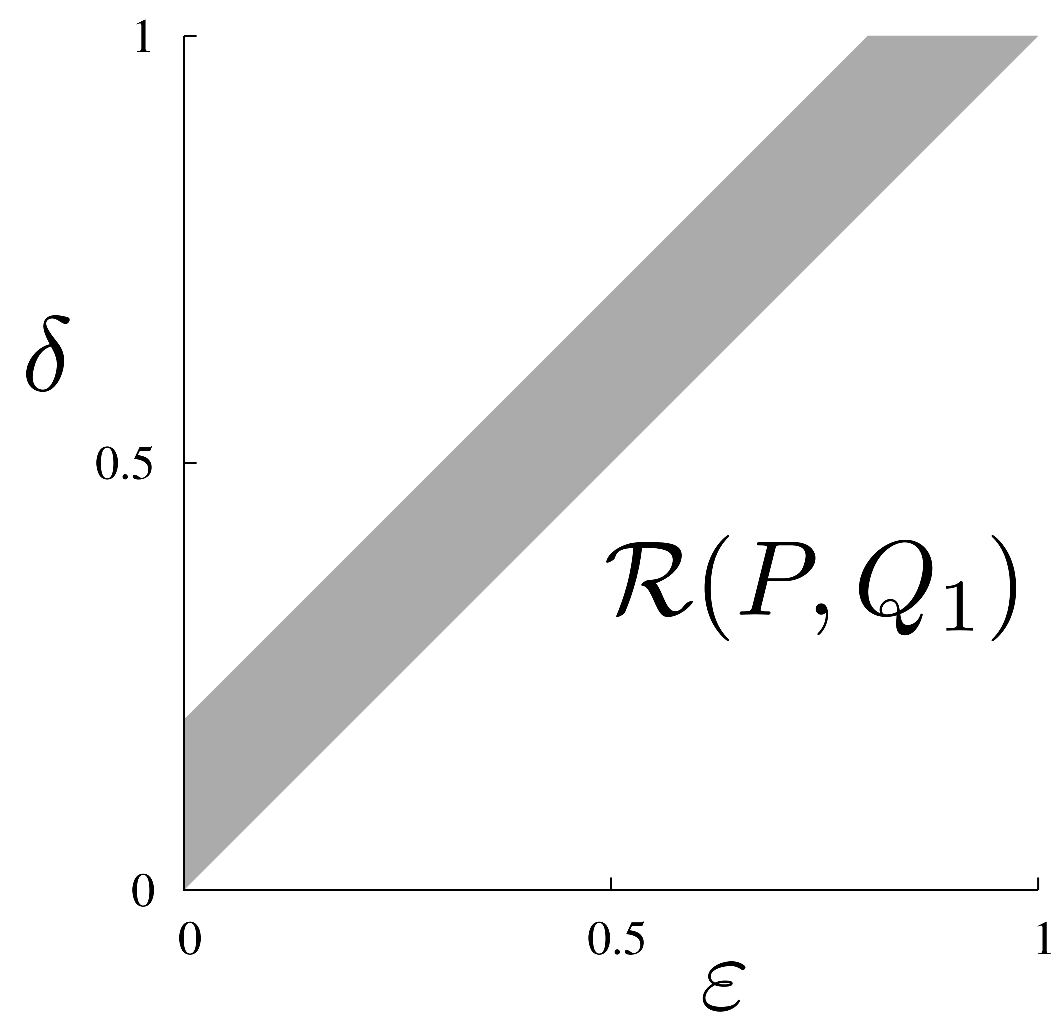
Mode Collapse region

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \;, \qquad \text{and} \qquad Q(S) \leq \varepsilon \;.$$



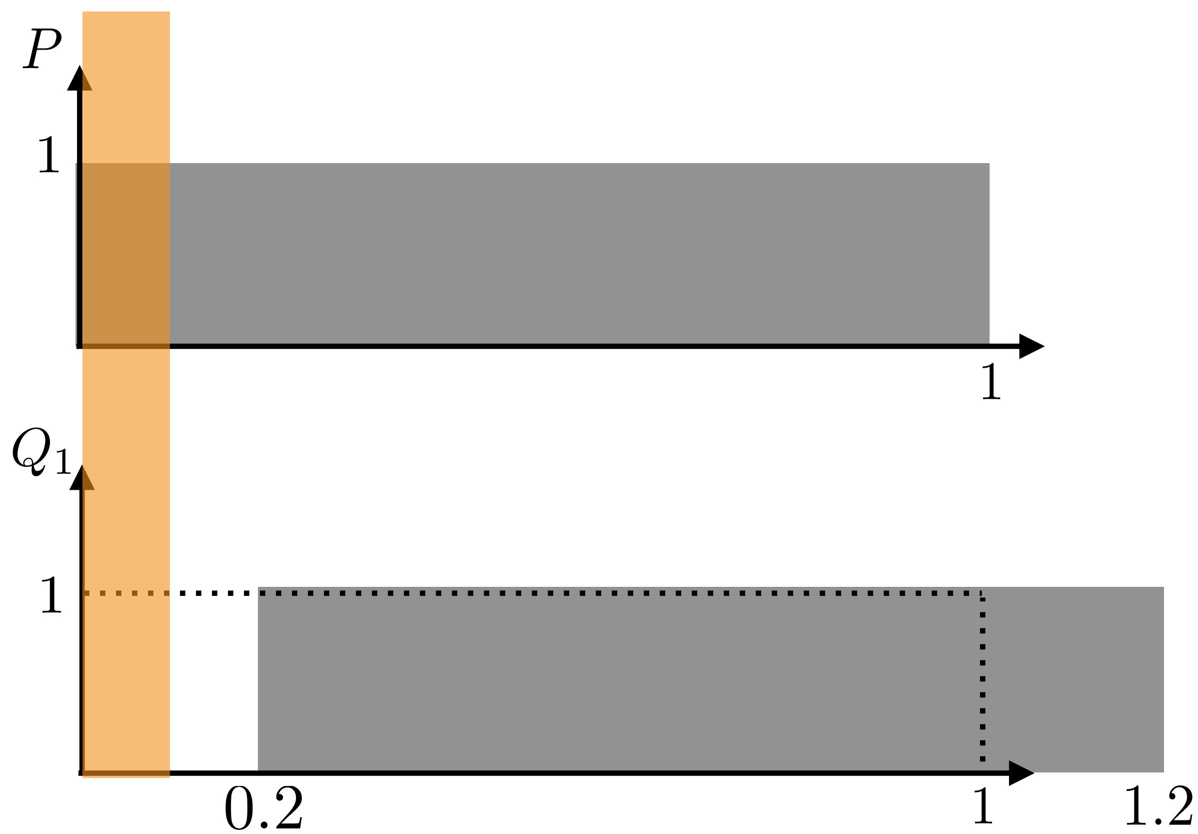


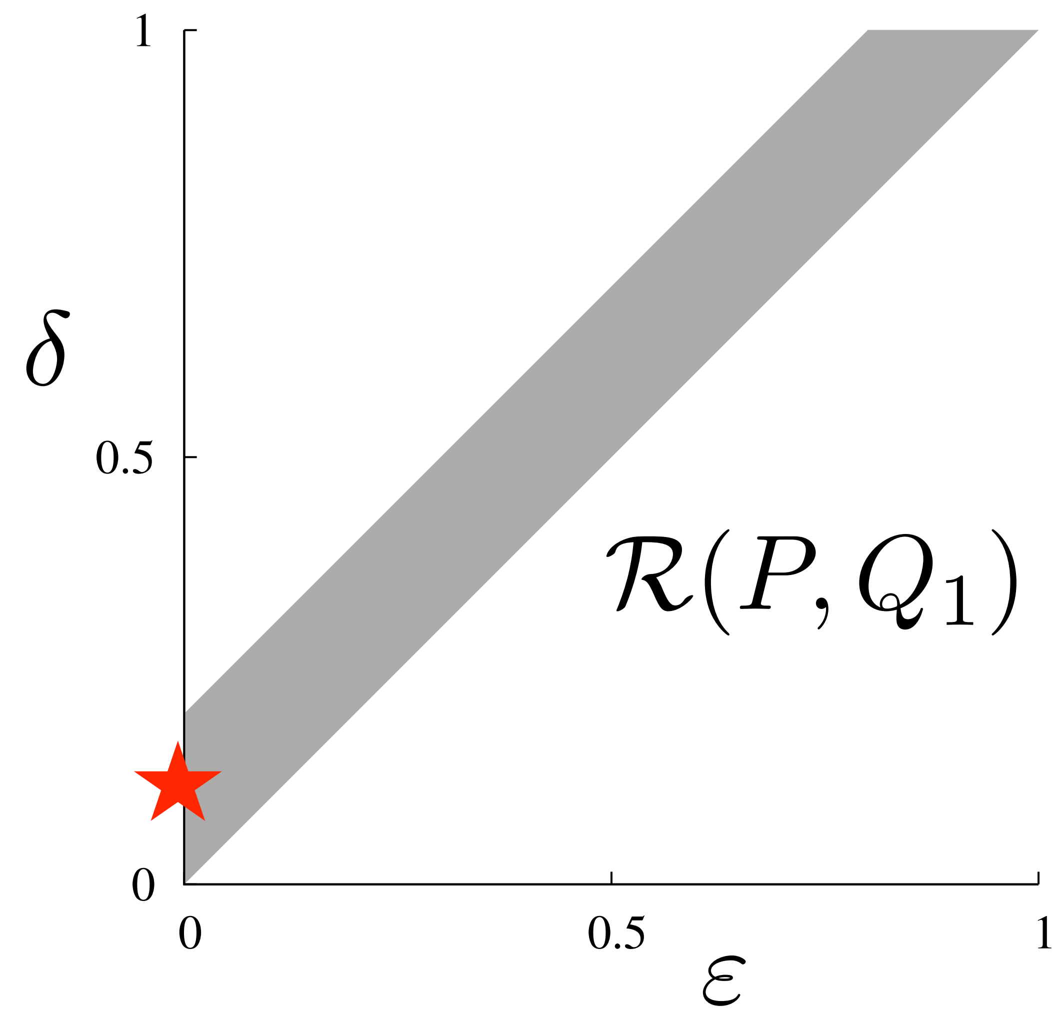
Mode Collapse region

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \qquad \text{and} \qquad Q(S) \leq \varepsilon \ .$$



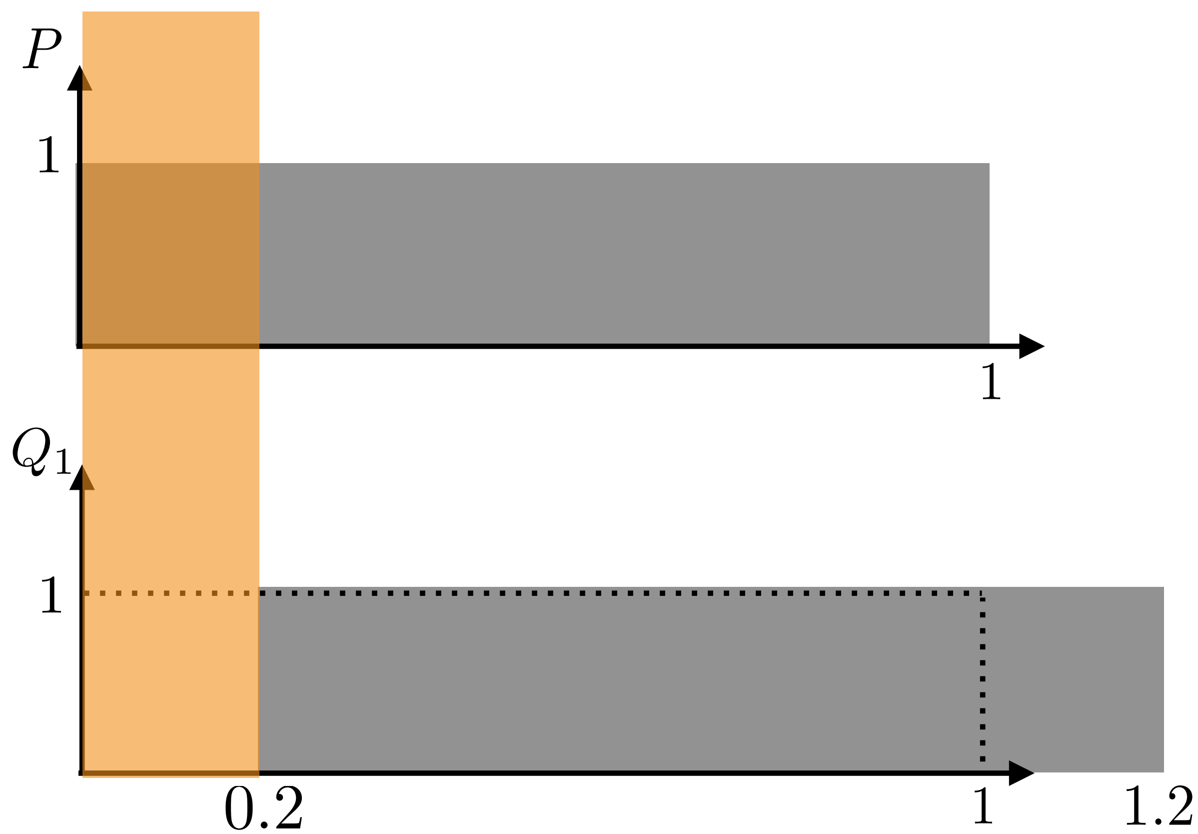


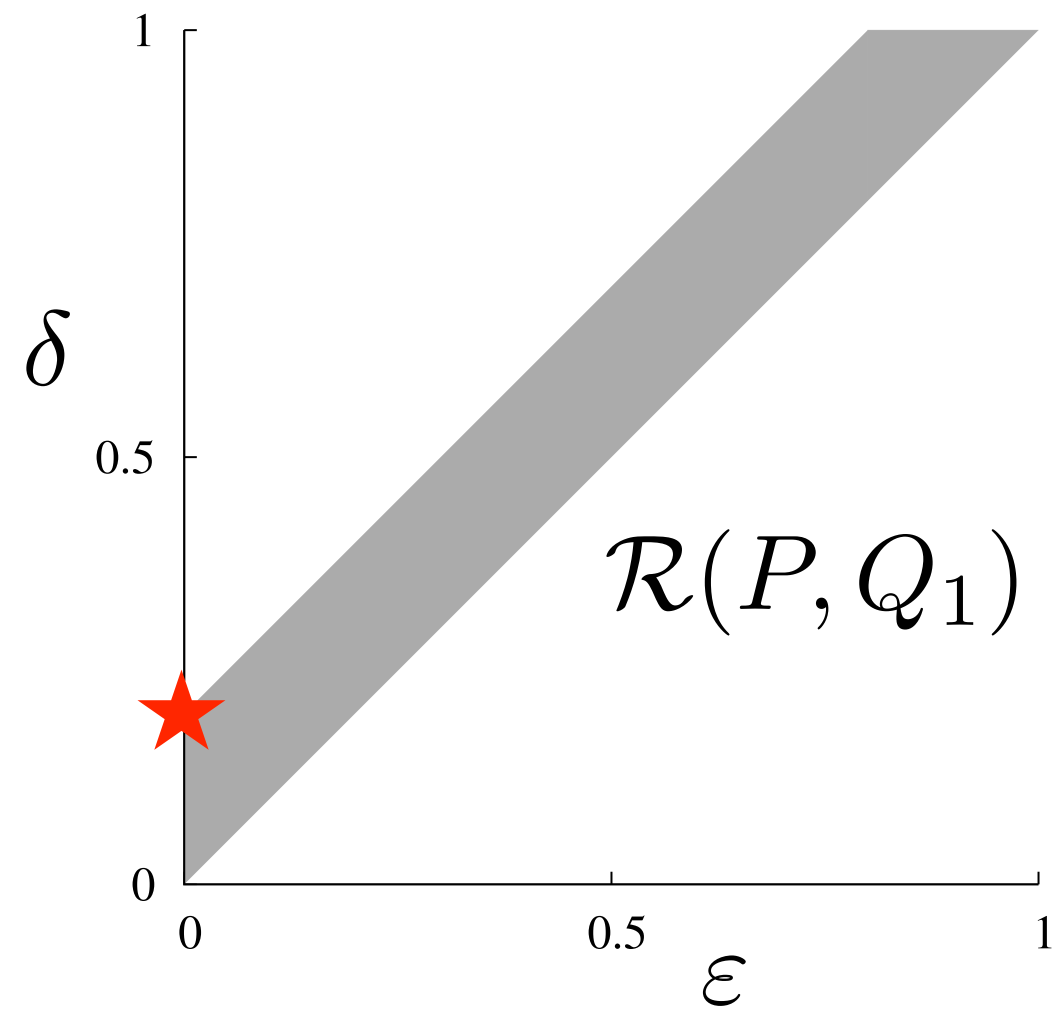
Mode Collapse region

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \quad \text{and} \quad Q(S) \leq \varepsilon \ .$$



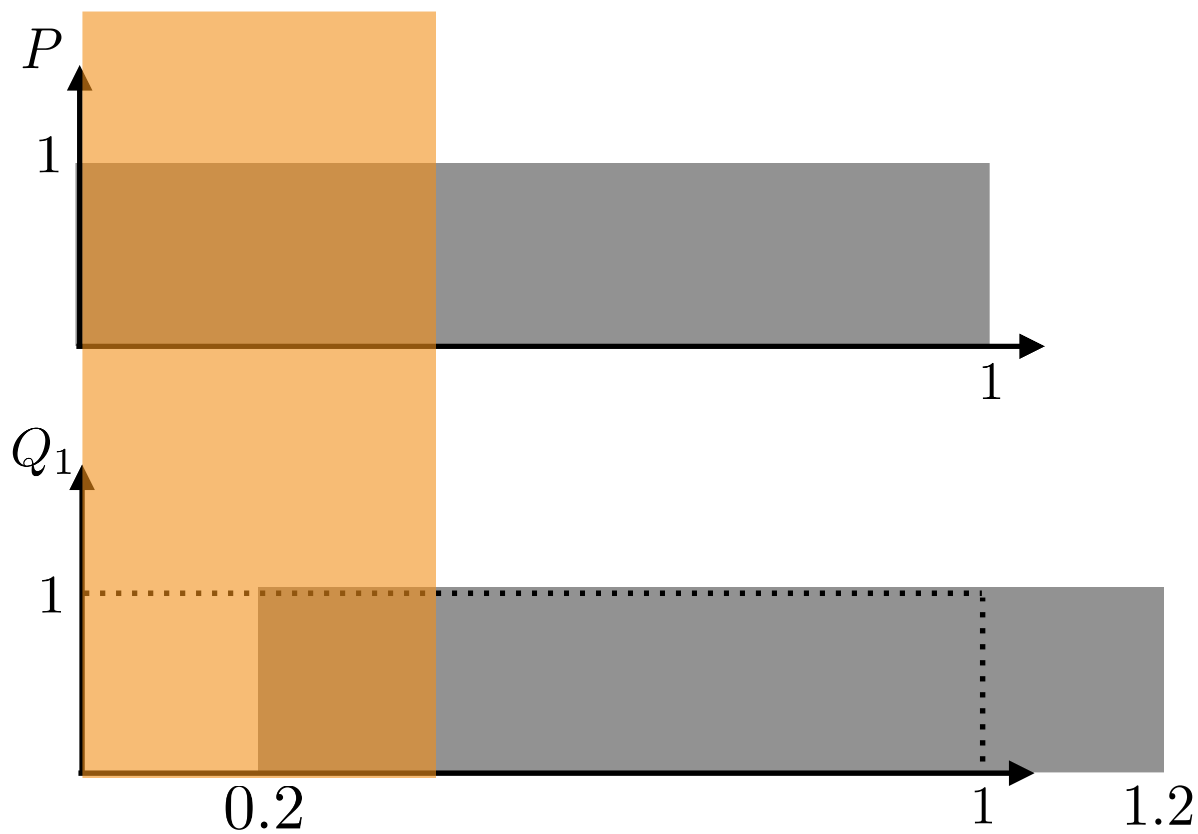


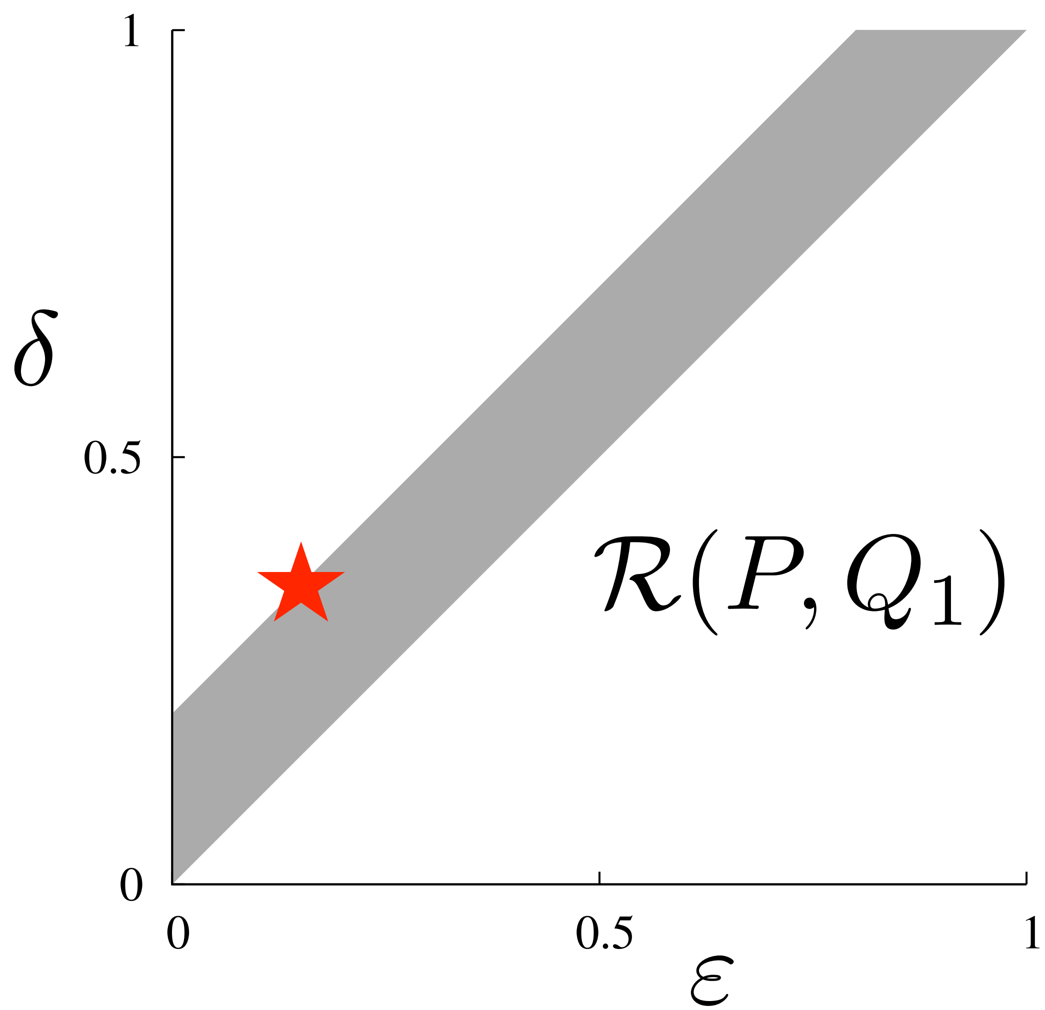
Mode Collapse region

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \qquad \text{and} \qquad Q(S) \leq \varepsilon \ .$$



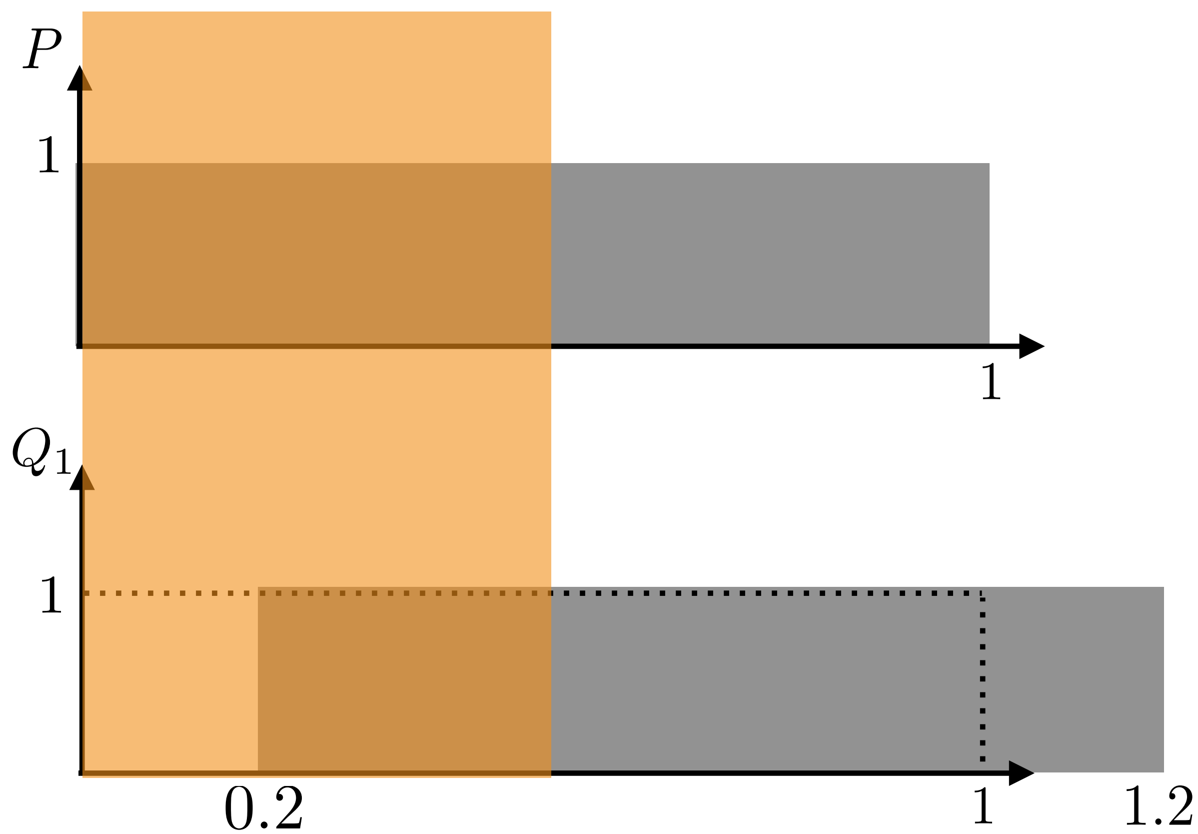


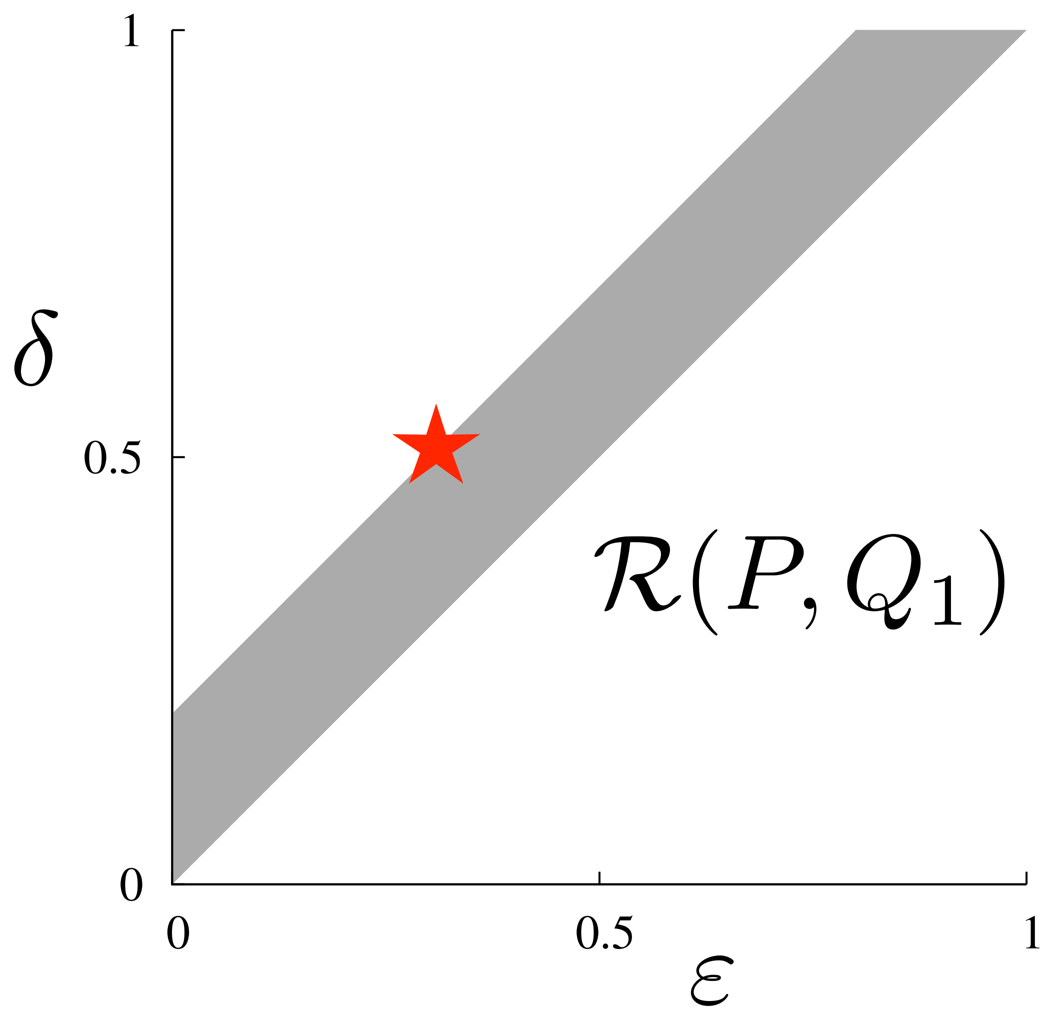
Mode Collapse region

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \quad \text{and} \quad Q(S) \leq \varepsilon \ .$$



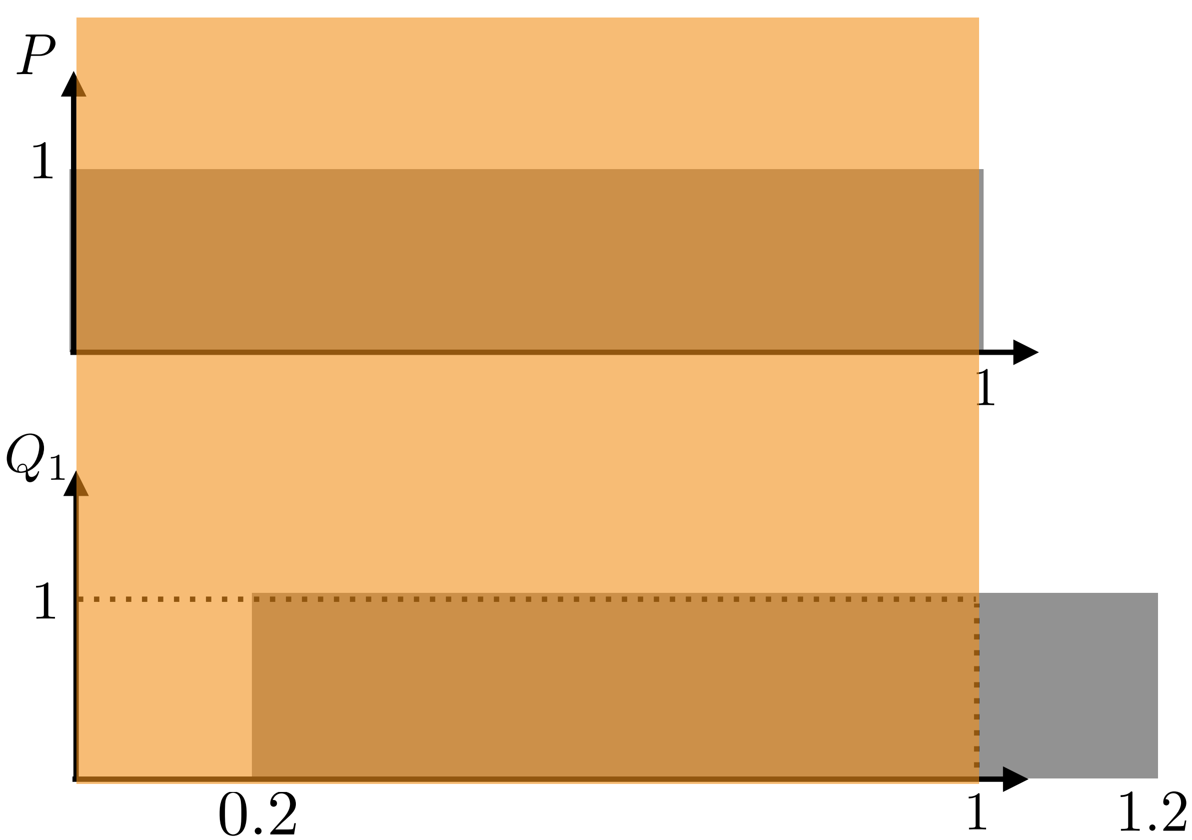


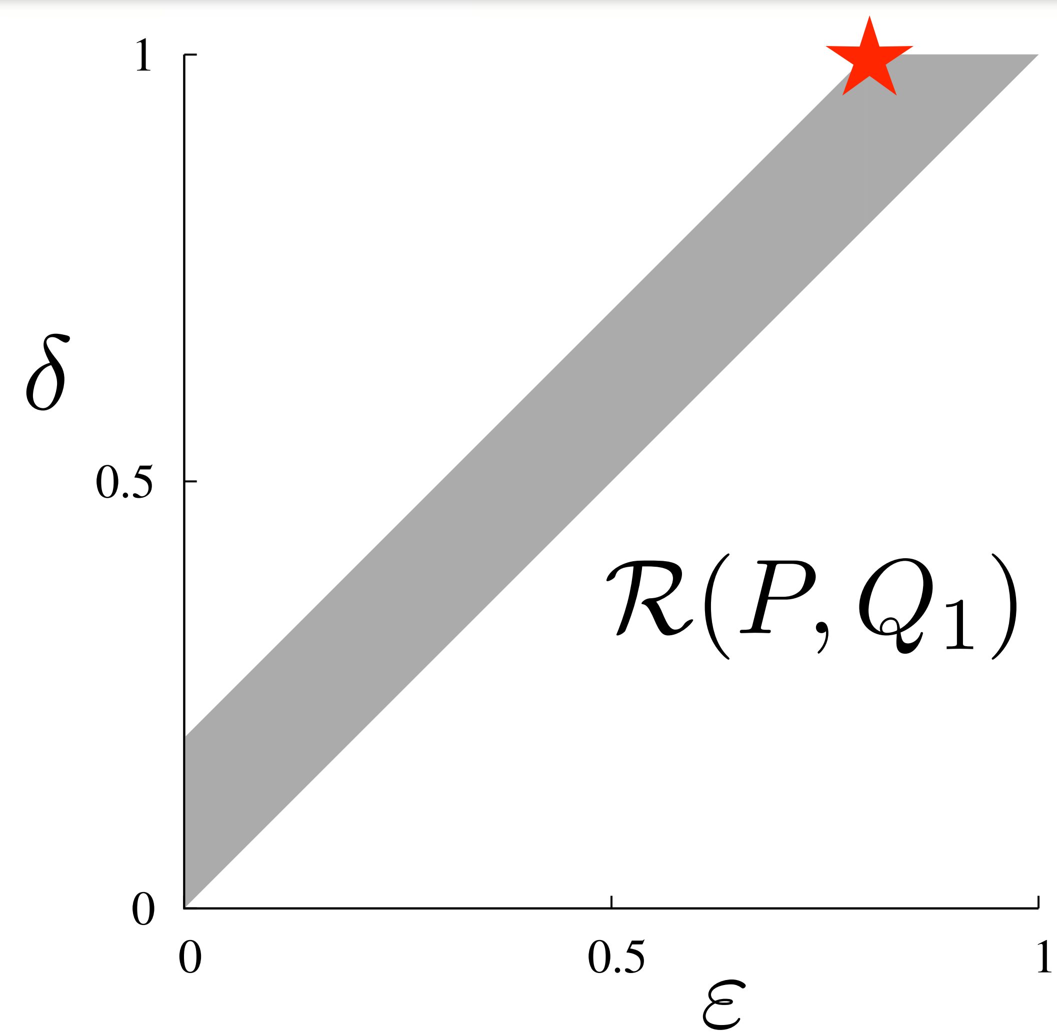
Mode Collapse region

Definition [mode collapse region]

We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \quad \text{and} \quad Q(S) \leq \varepsilon \ .$$



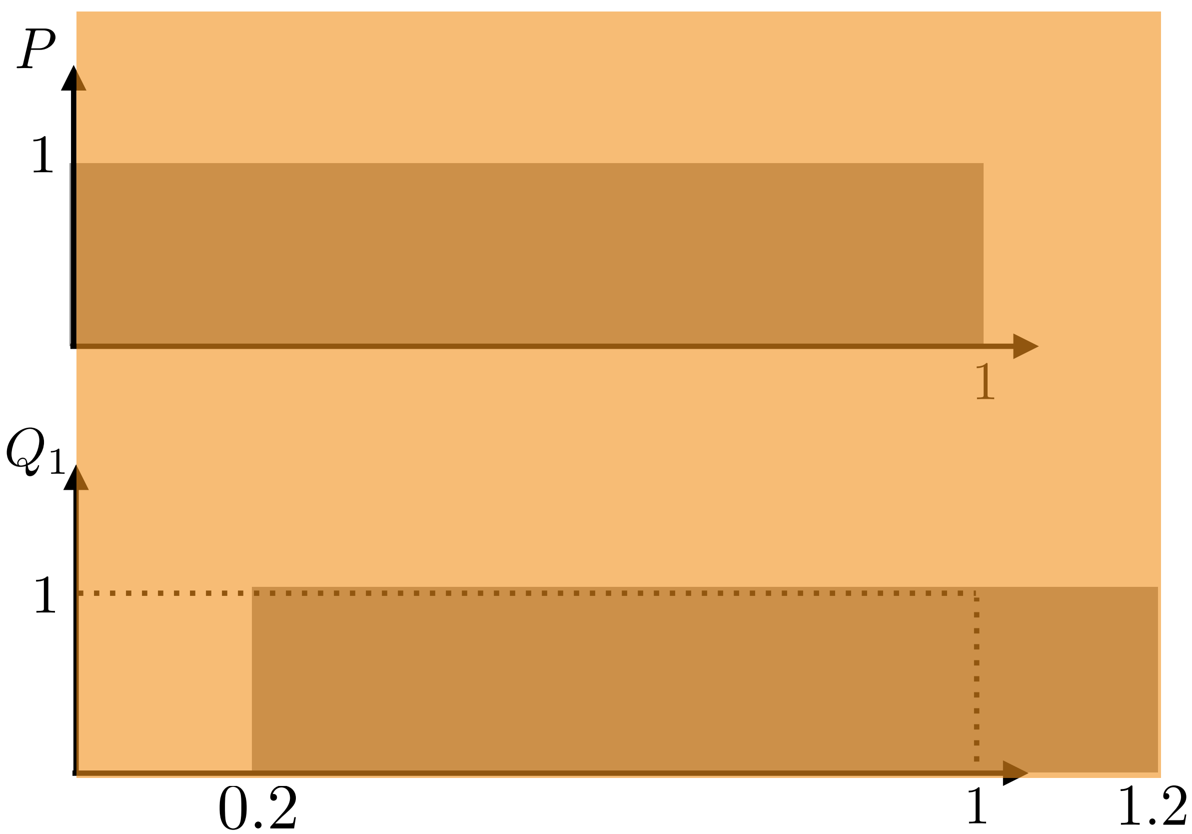


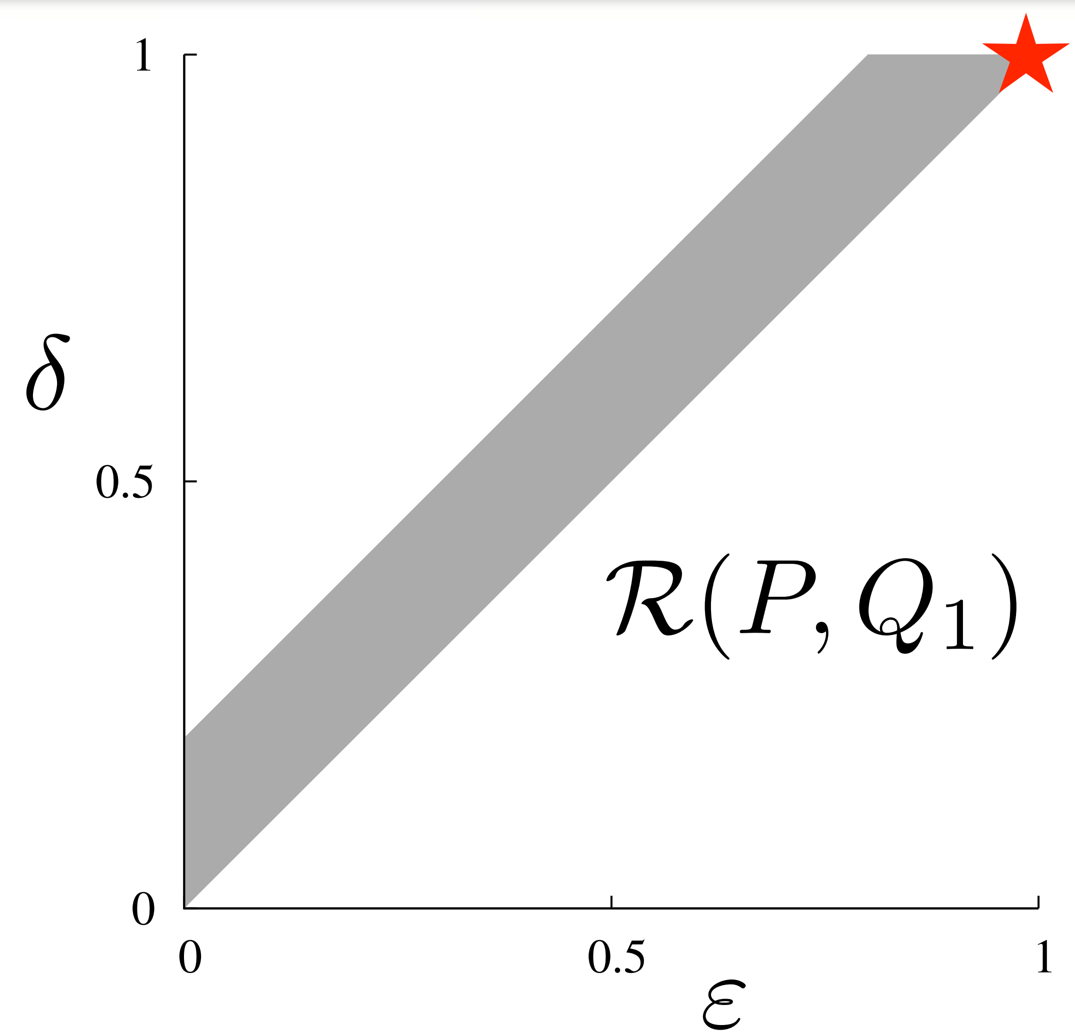
Mode Collapse region

Definition [mode collapse region]

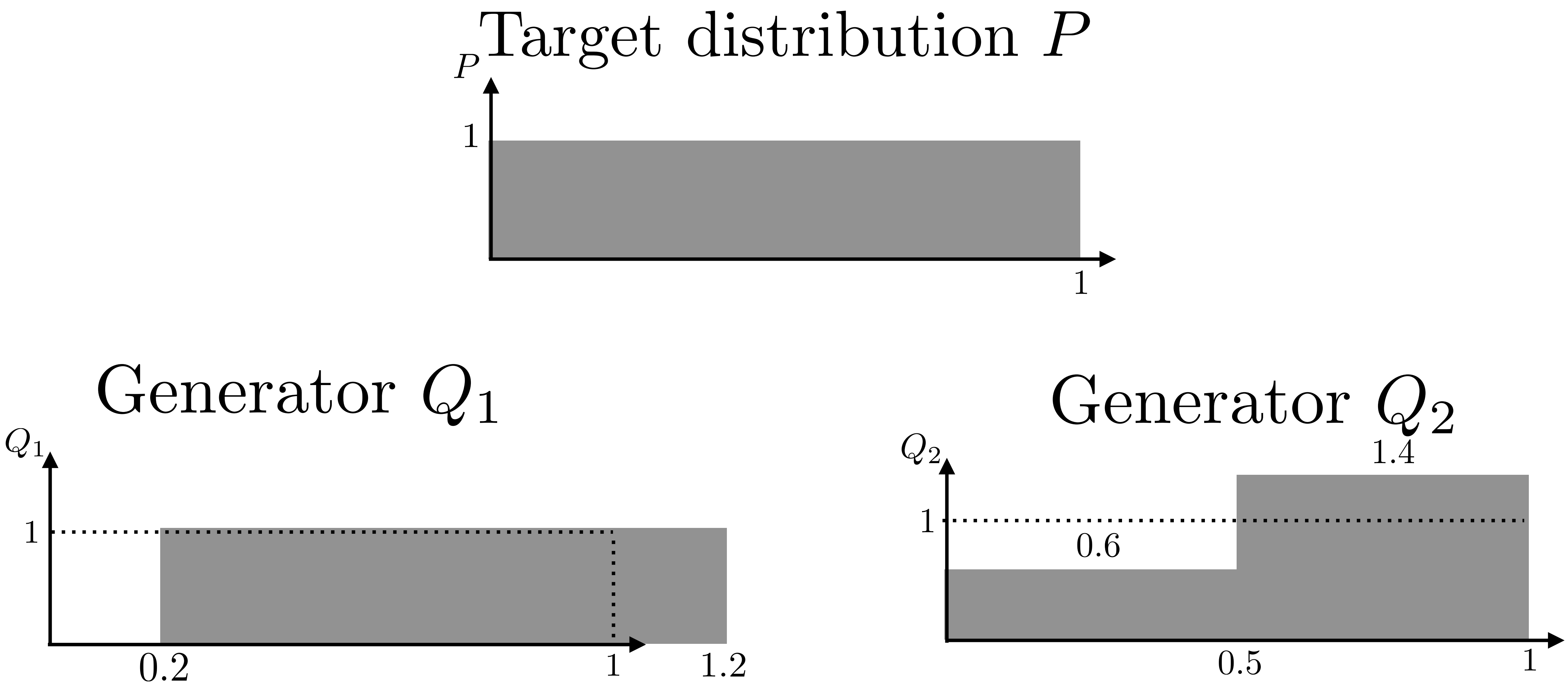
We say a pair (P, Q) of a target distribution P and a generator distribution Q has (ε, δ)-**mode collapse** if there exists a set S such that

$$P(S) \geq \delta \ , \qquad \text{and} \qquad Q(S) \leq \varepsilon \ .$$





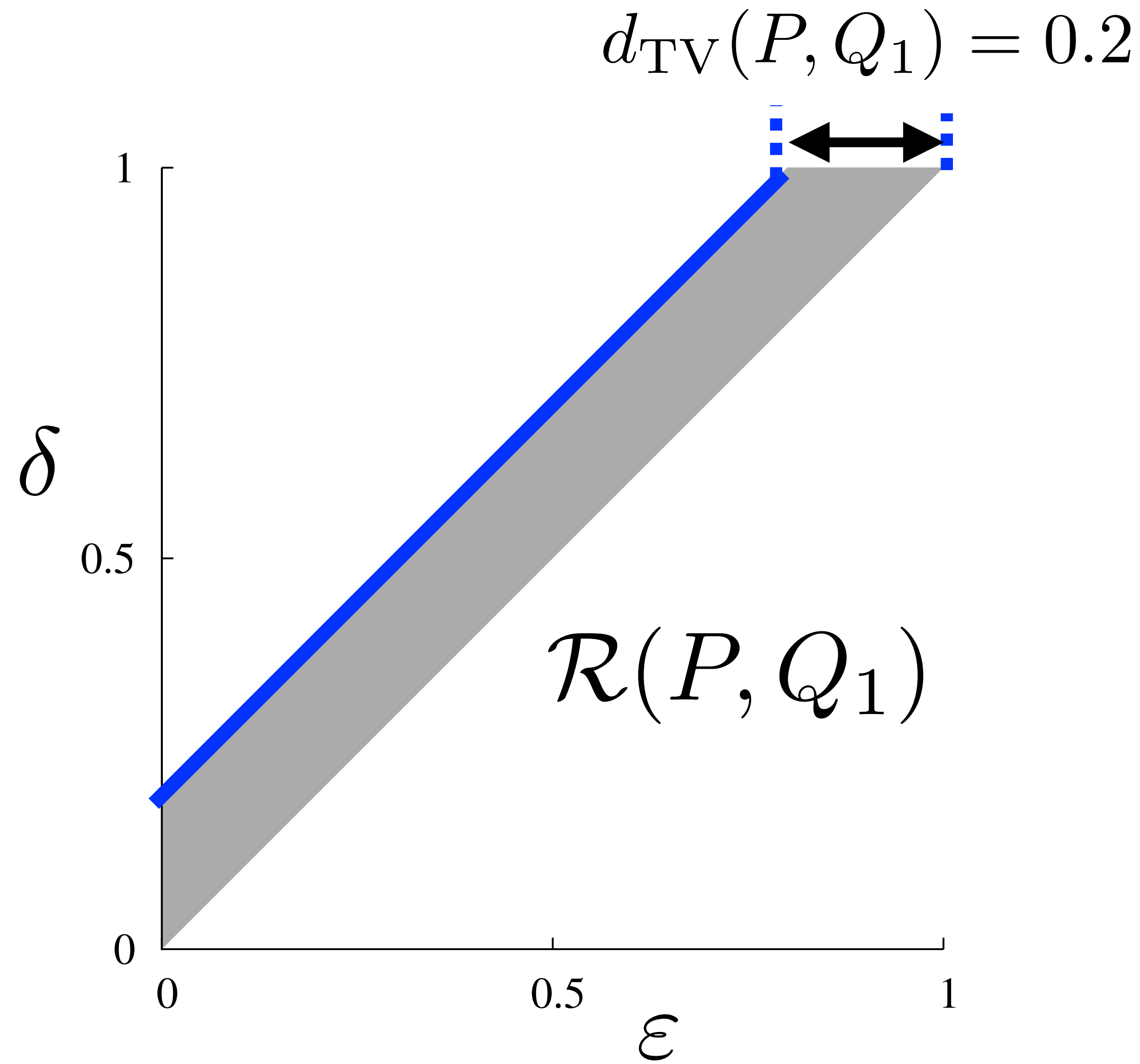
(Detection) theoretical understanding of Mode Collapse

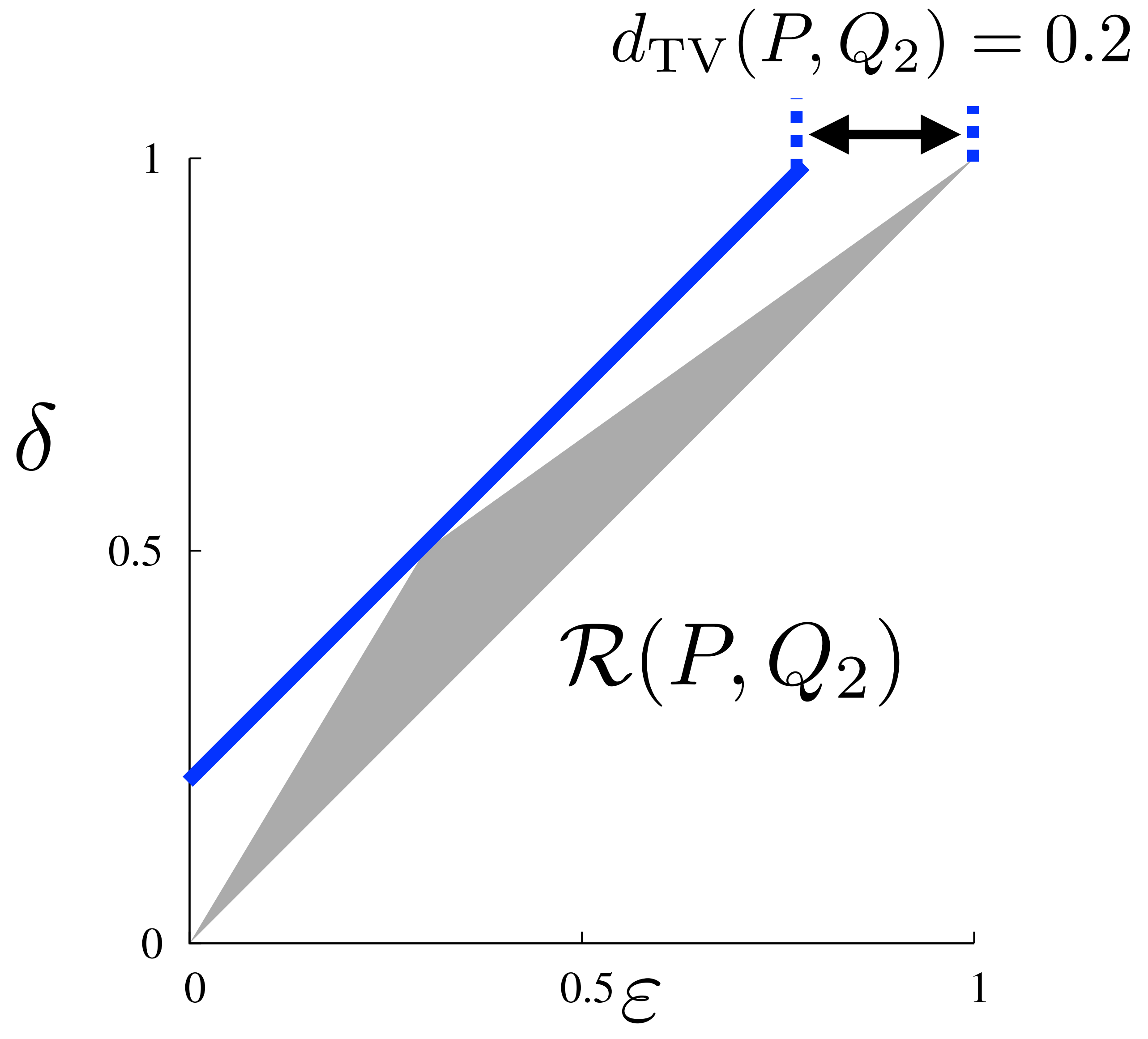


$d_{\mathrm{TV}}(P, Q_1) = 0.2$	Loss	$d_{\mathrm{TV}}(P, Q_2) = 0.2$
Strong	Mode Collapse	Weak

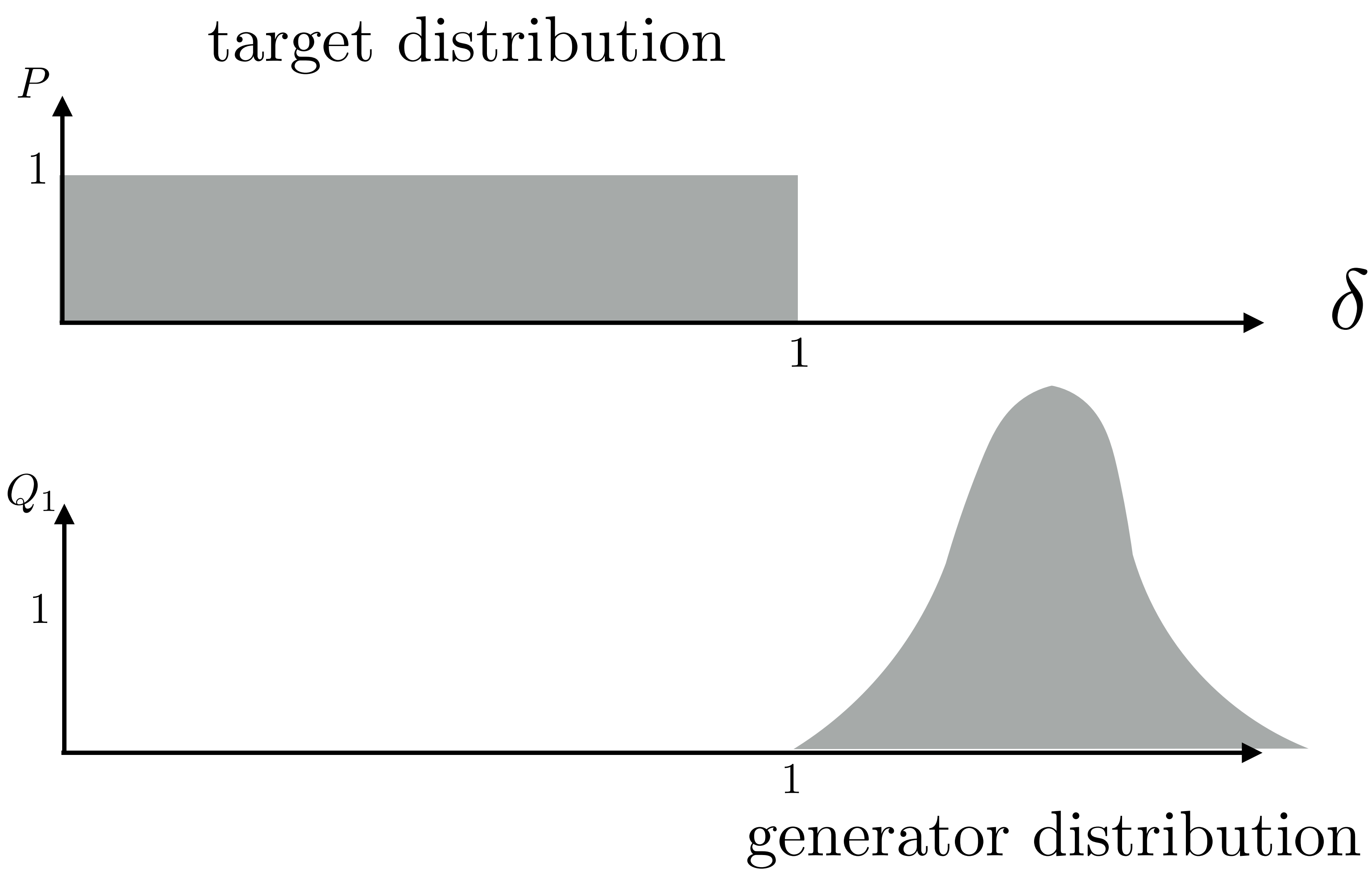
Mode Collapse region

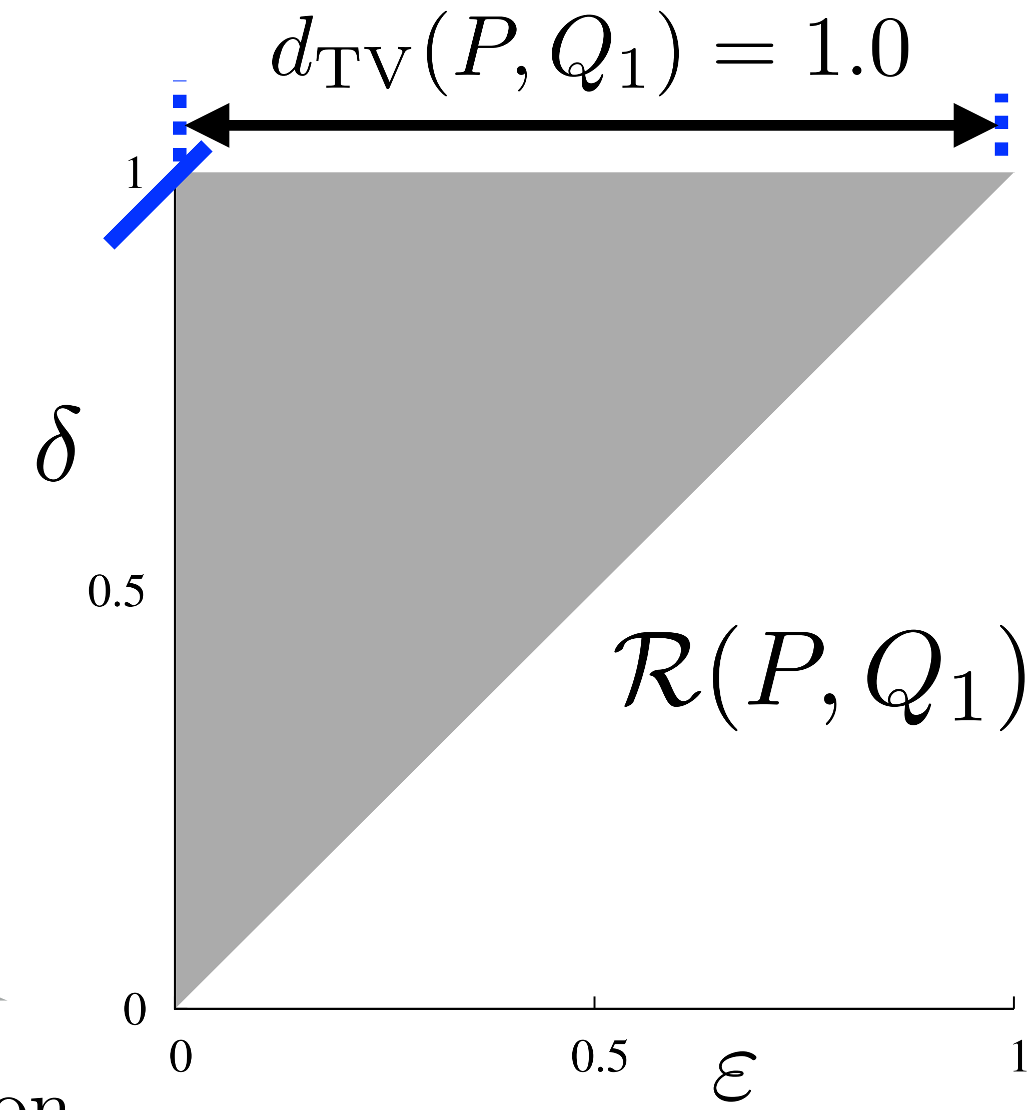
- The 2-D region representation
  - allows formal comparison of strengths of Mode Collapse
  - Read off all divergences





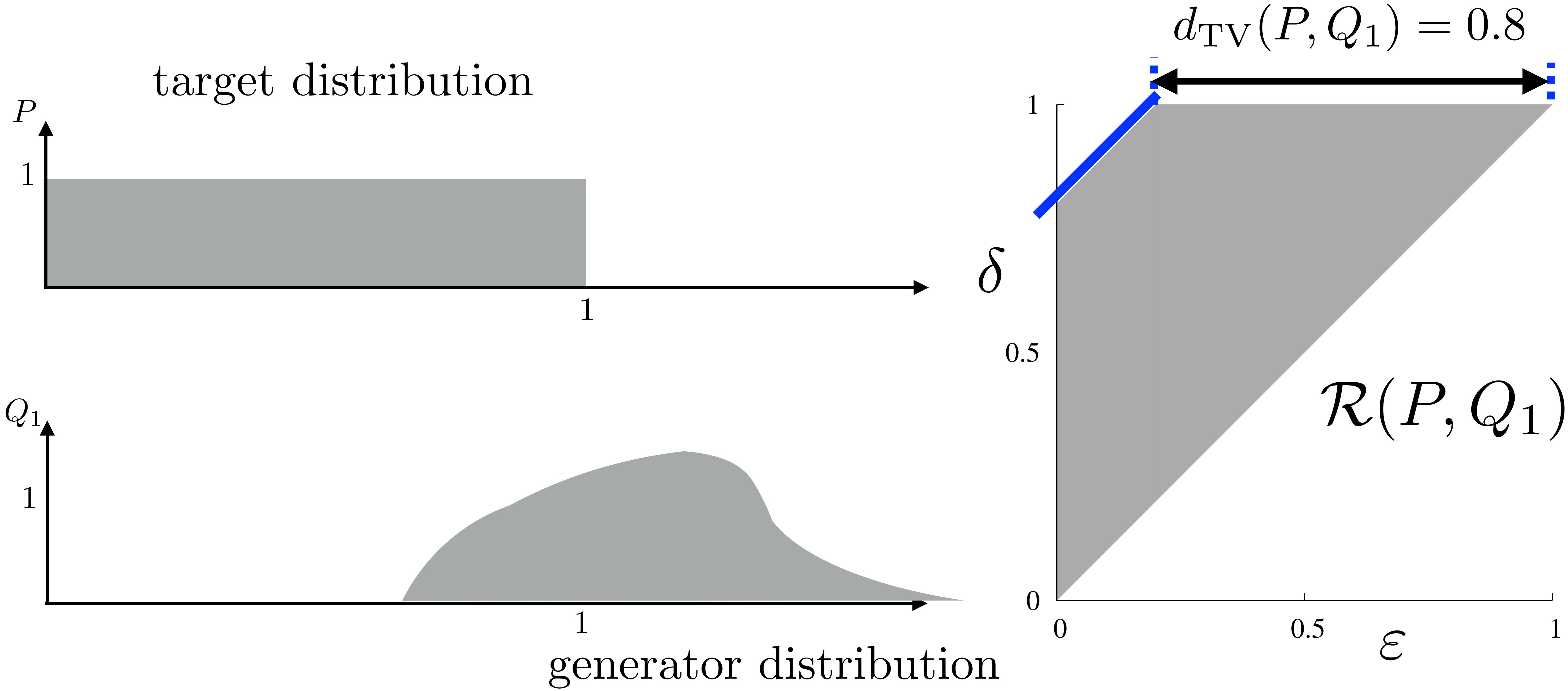
Alternate view of GAN training via Mode Collapse region





corresponding
Mode Collapse region

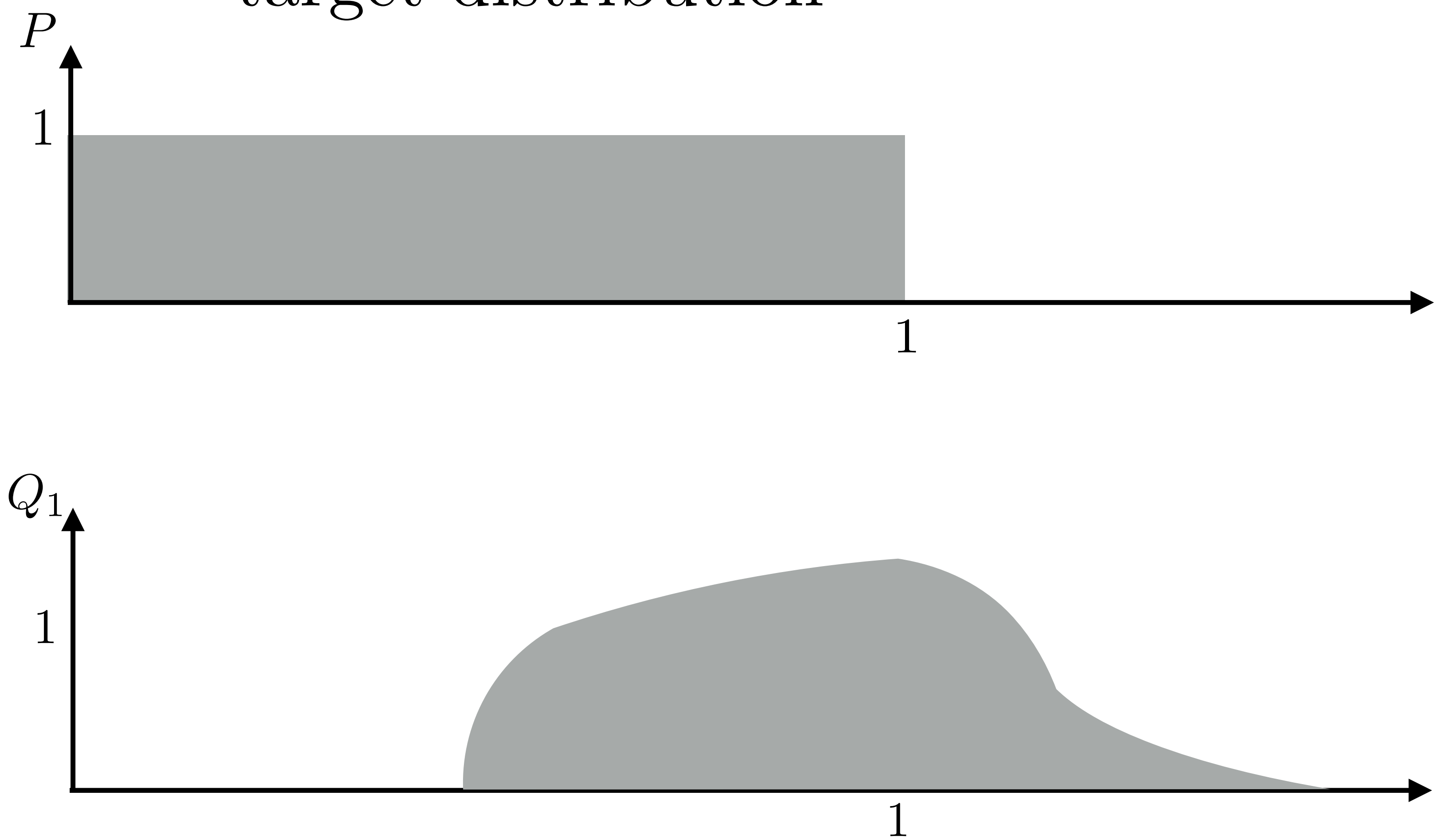
GAN training via Mode Collapse region



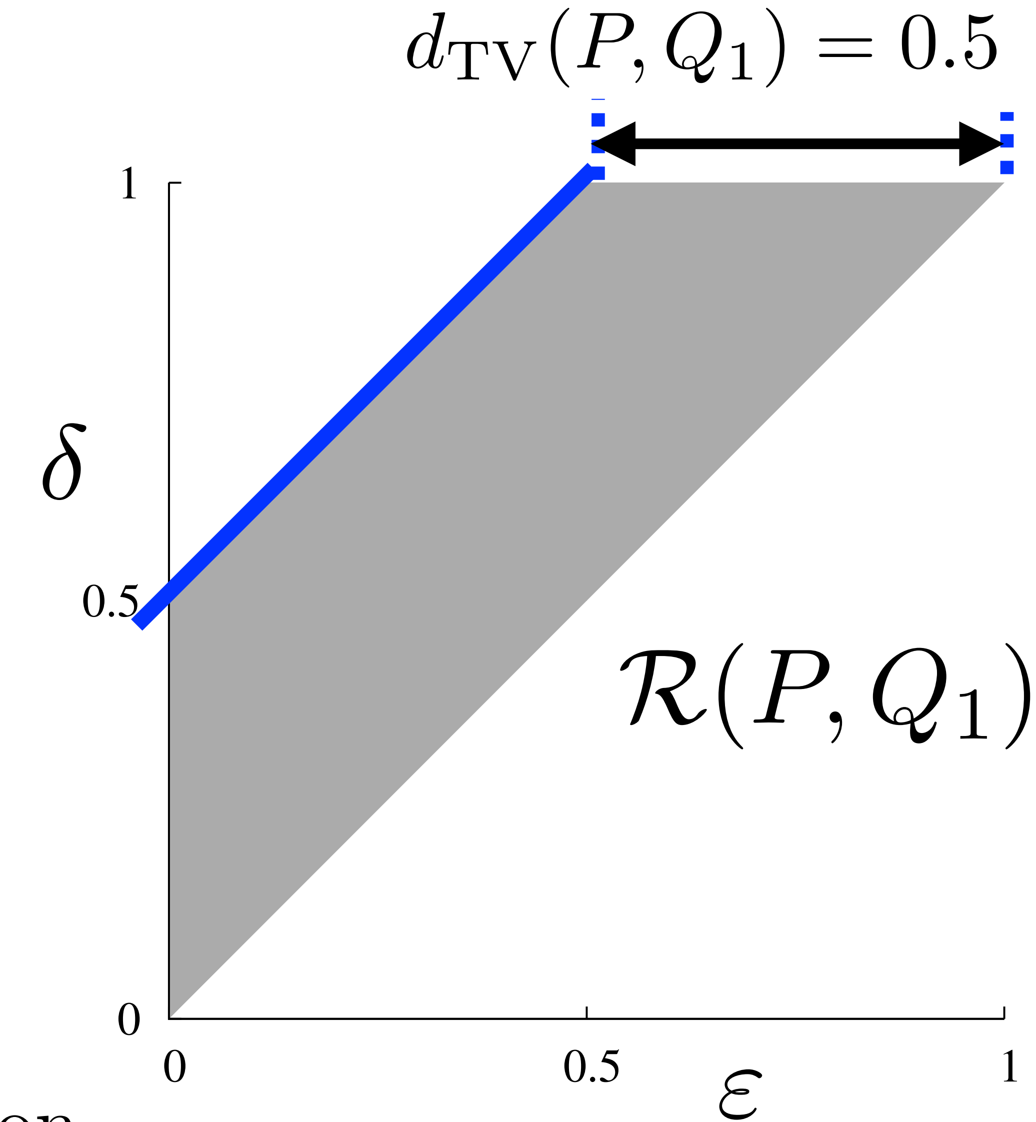
corresponding
Mode Collapse region

GAN training via Mode Collapse region

target distribution

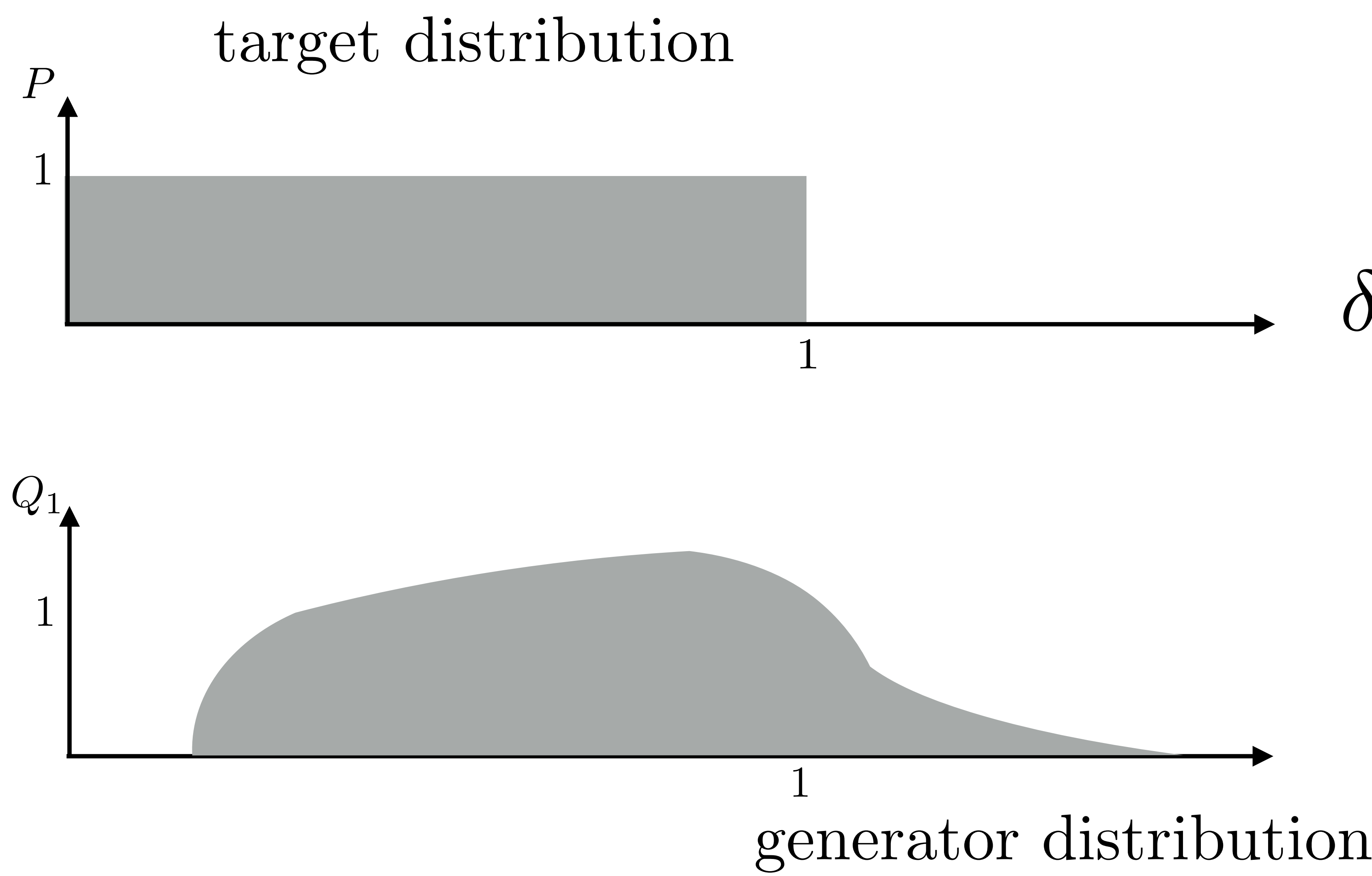


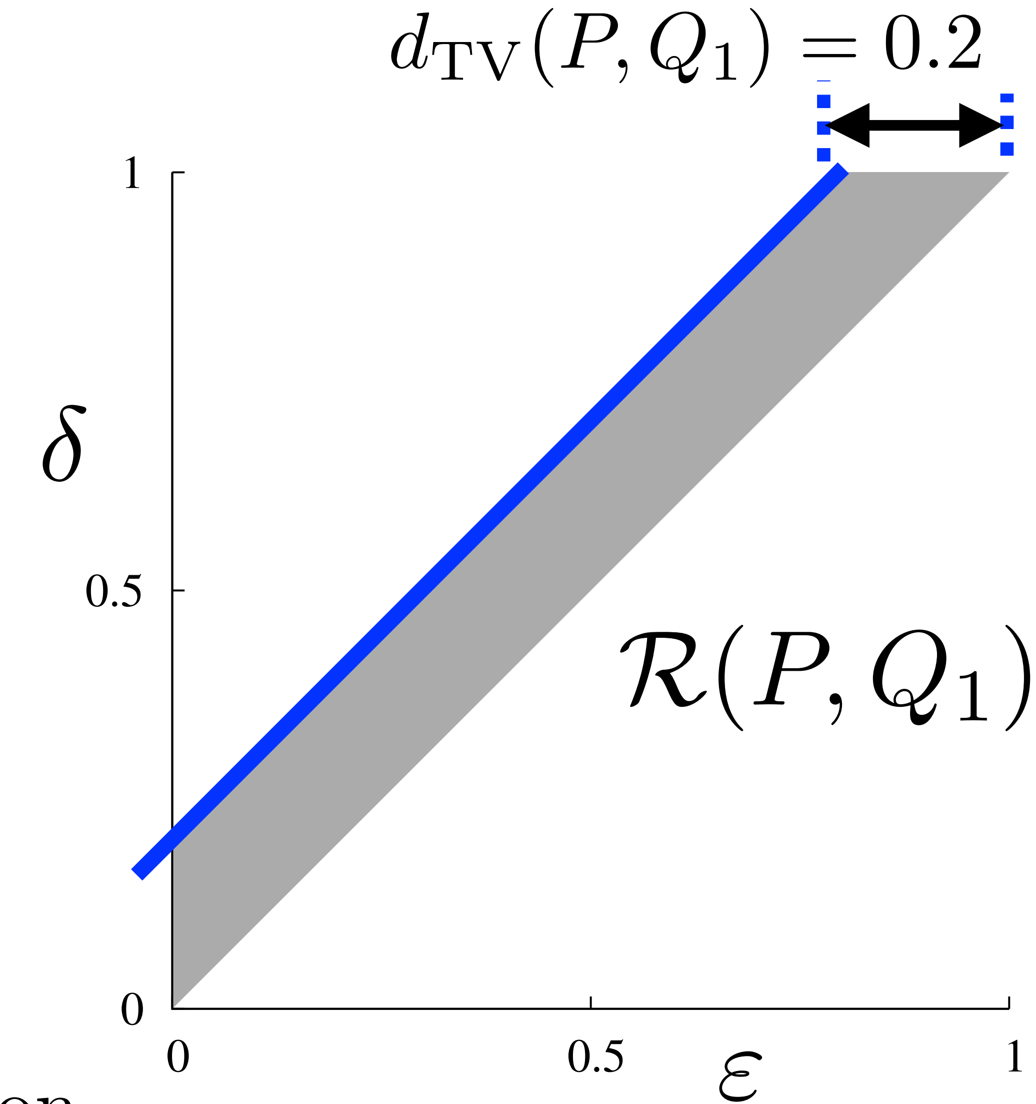
generator distribution



corresponding
Mode Collapse region

GAN training via Mode Collapse region

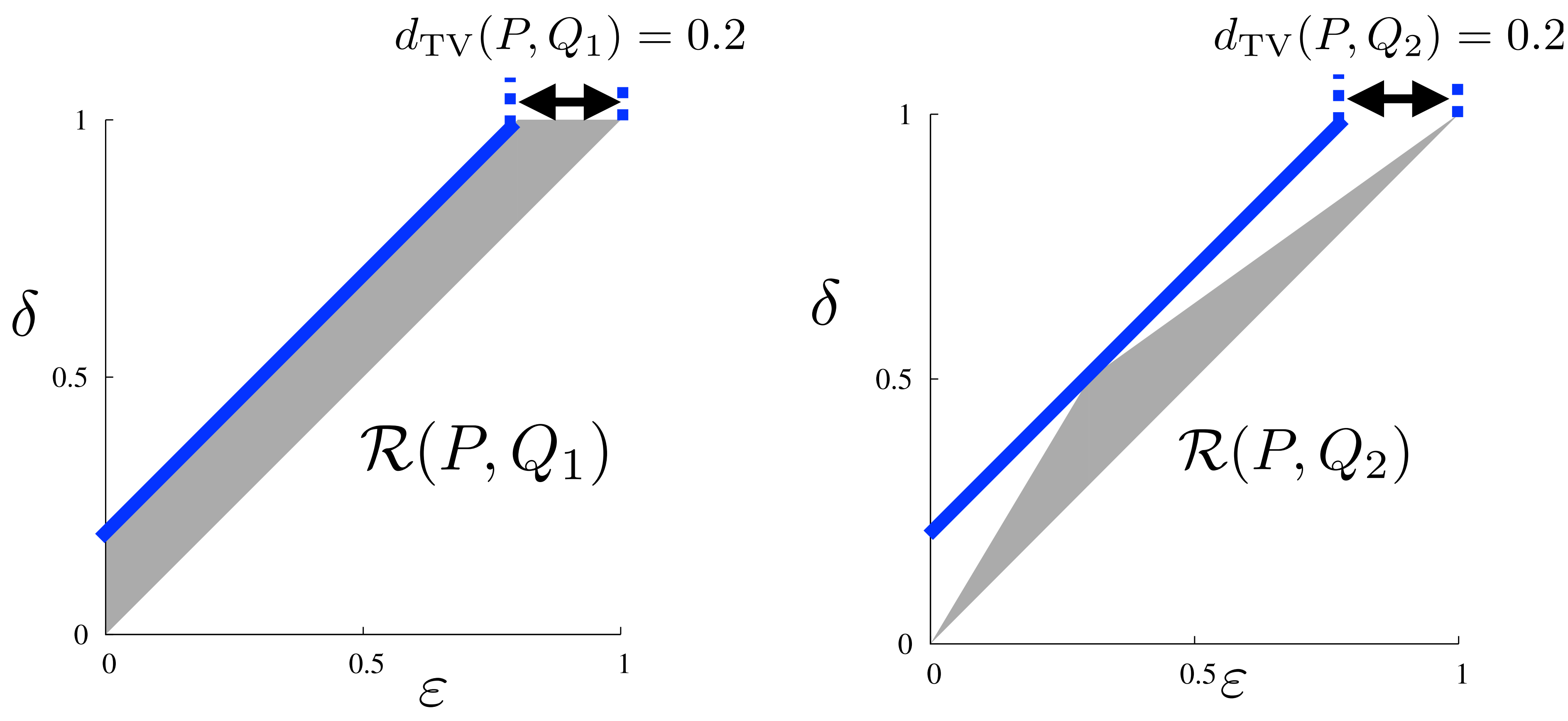




corresponding
Mode Collapse region

Main challenge

- Varying degrees of Mode Collapses are indistinguishable from the standard choices of losses



- Goal: how do we design new (family of) losses that naturally penalizes Mode Collapse?

Lifting the loss to the product distributions

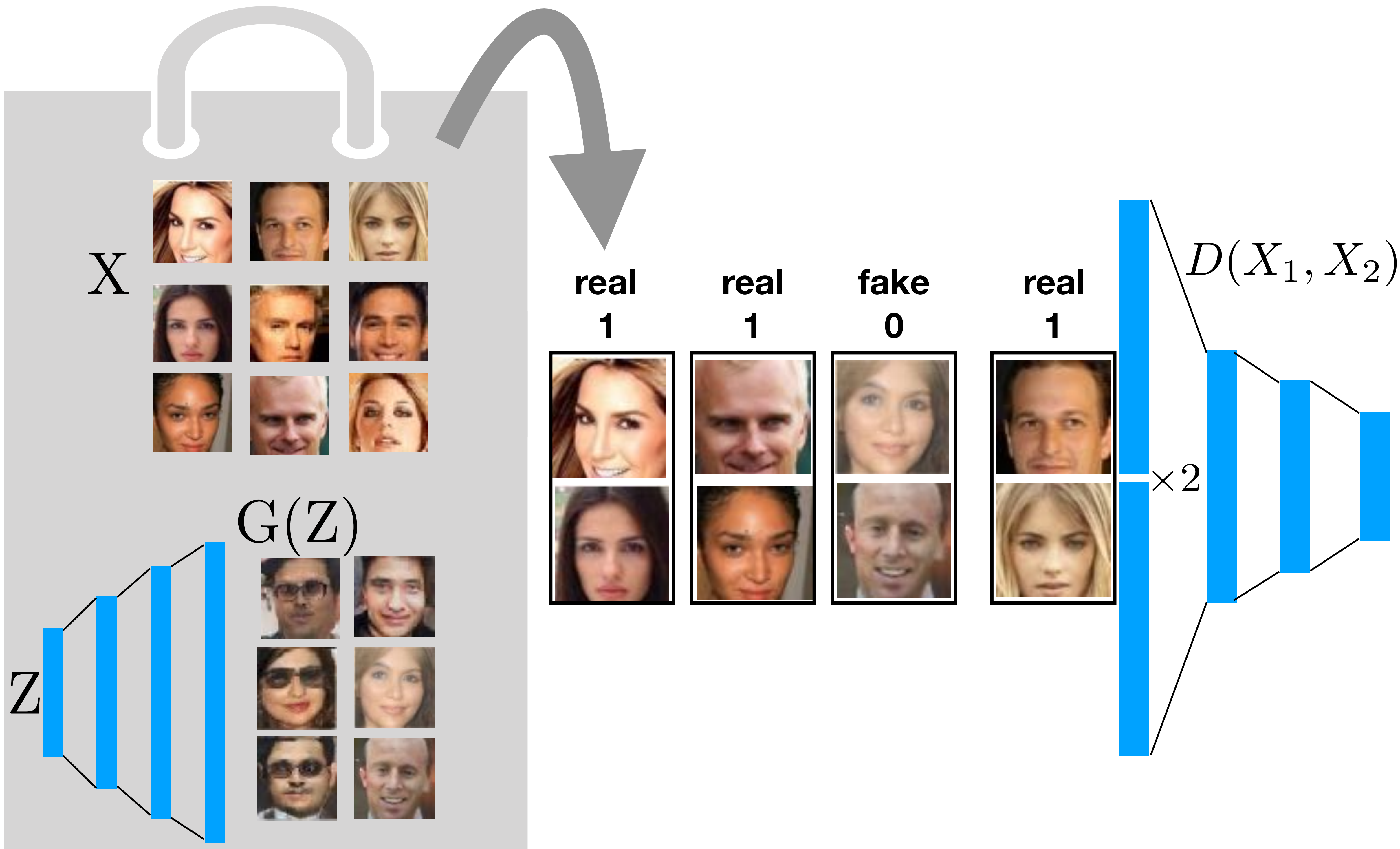
- Mathematical intuitions from
  - Comparisons of experiments [Blackwell1953]
  - (reverse) Data-processing inequality
  - Differential Privacy [KairouzOhViswanath2017]

$D_{\mathrm{TV}}(P^m, Q^m)$ naturally penalizes mode collapse

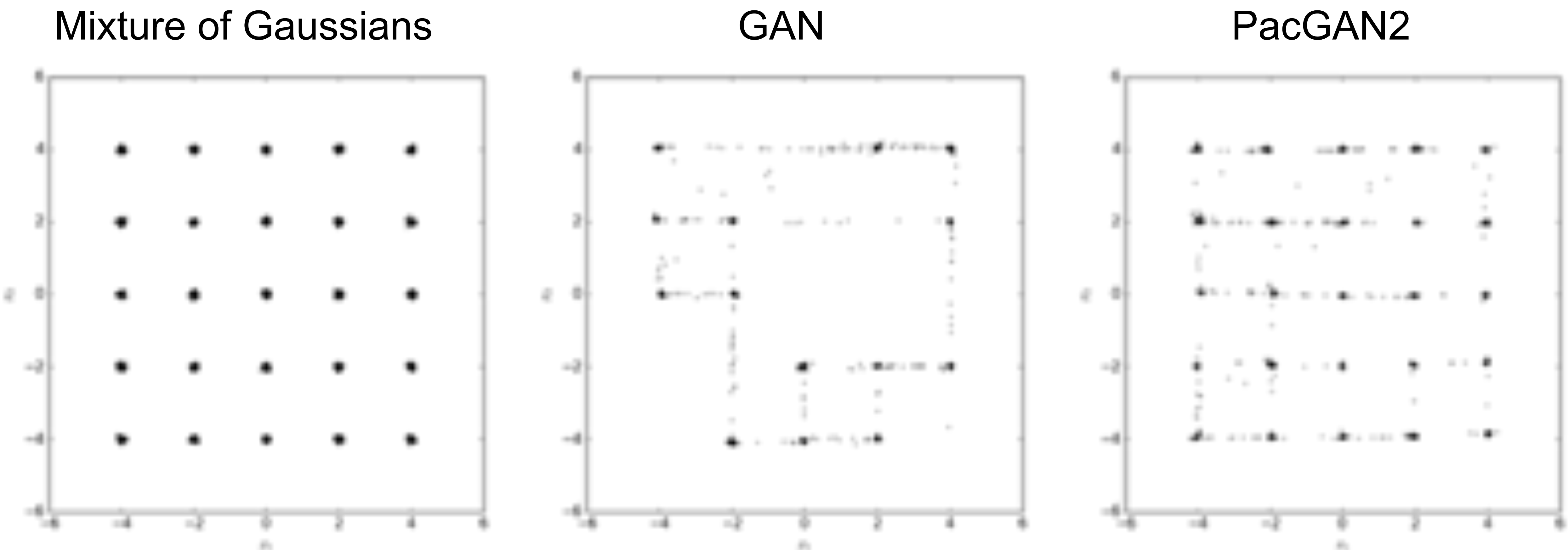
["Equivalent comparisons of experiments",Blackwell,1953]
["The composition theorem for differential privacy", Kairouz,Oh,Viswanath,2017]

PacGAN: principled approach to Mode Collapse

- Discriminator needs to sample from the product distribution



Benchmark test

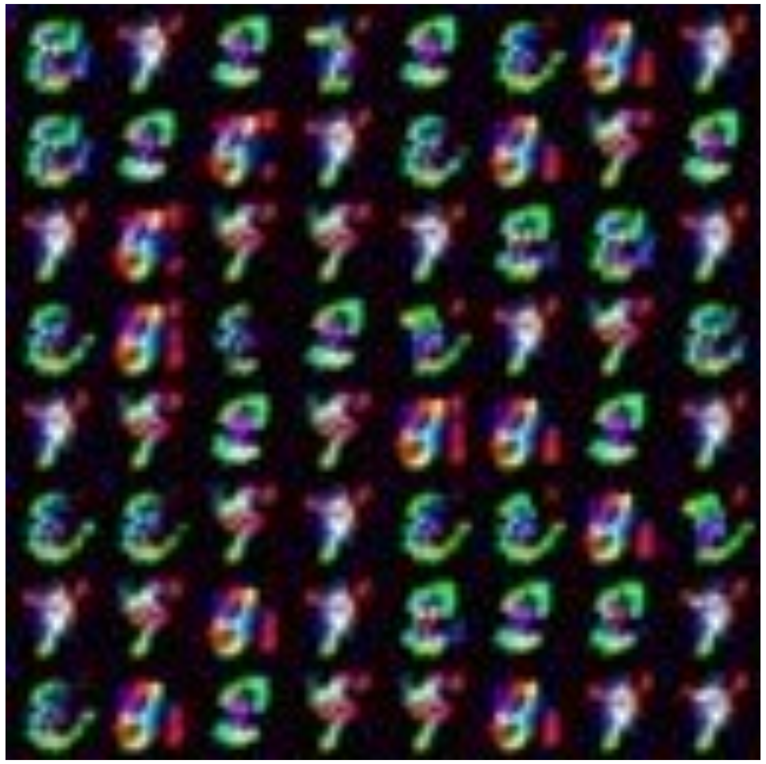


	Modes (Max 25)
GAN	17.3
PacGAN2	23.8
PacGAN3	24.6
PacGAN4	24.8

Benchmark tests

Stacked MNIST

DCGAN



PacDCGAN2

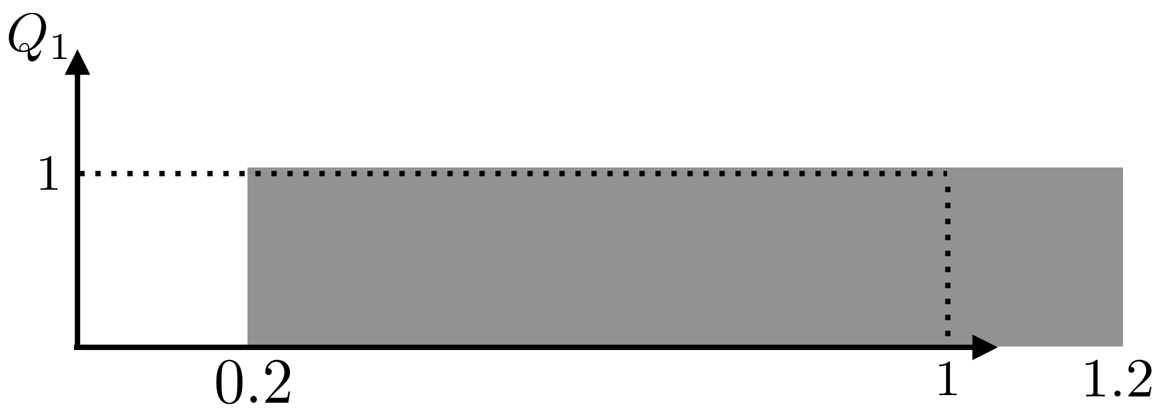
	Modes (Max 1000)
DCGAN	99.0
ALI	16.0
Unrolled GAN	48.7
VEEGAN	150.0
PacDCGAN2	1000.0
PacDCGAN3	1000.0
PacDCGAN4	1000.0

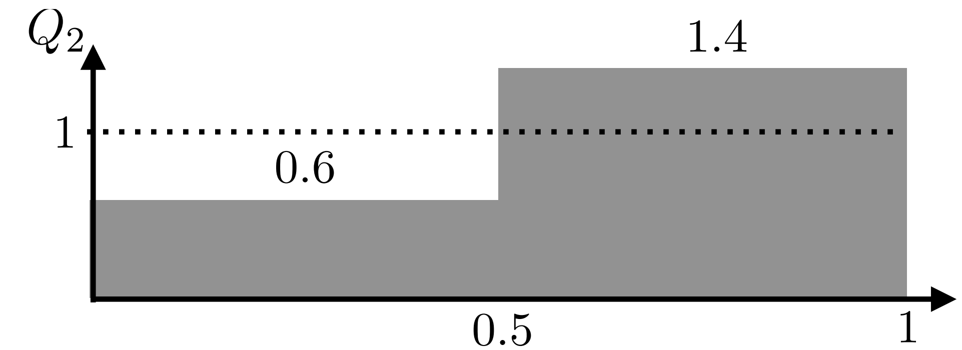
[“VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning”, Srivastava,Valkov,Russell,Gutmann,Sutton, 2017]

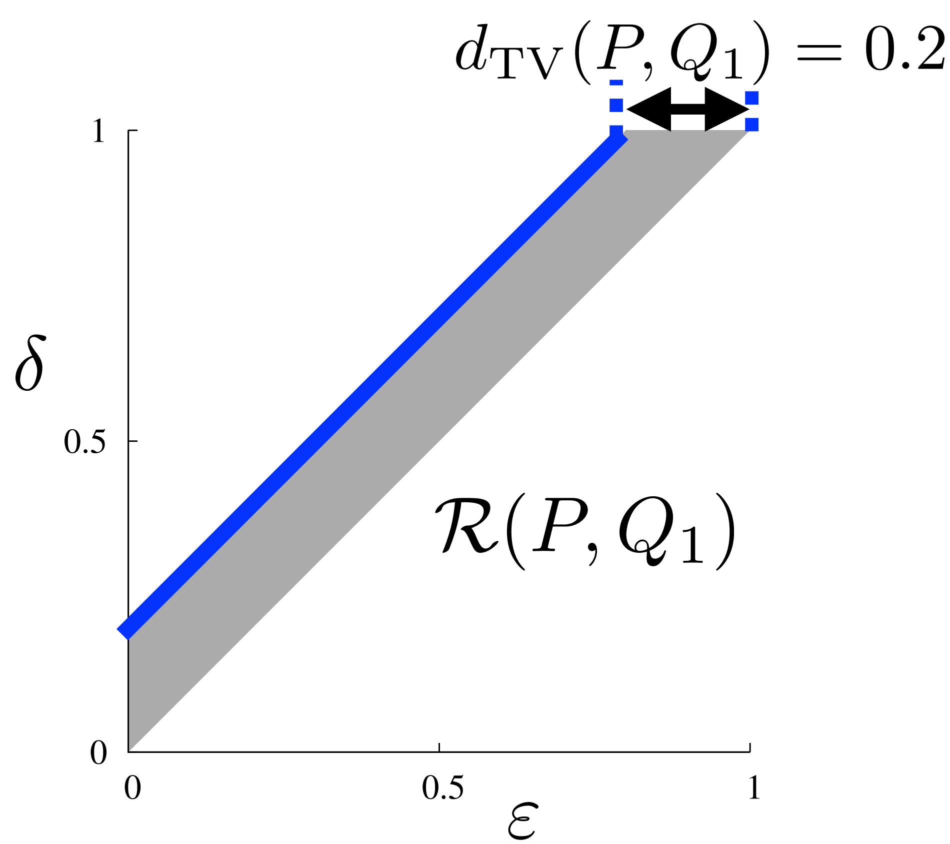
We can “measure” Mode Collapse via lifting

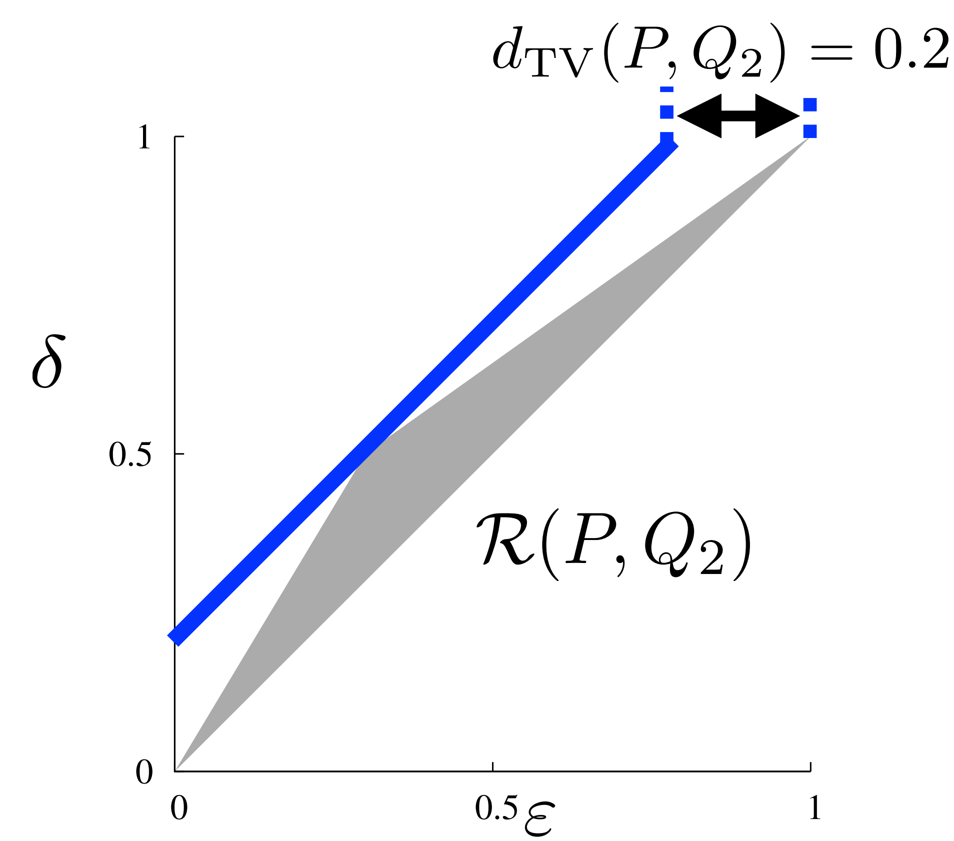
(reverse) data processing inequality [KairouzOhViswanath17]

$$\text{If } \mathcal{R}(P, Q_1) \supseteq \mathcal{R}(P, Q_2), \text{ then } \mathcal{R}(P^m, Q_1^m) \supseteq \mathcal{R}(P^m, Q_2^m)$$

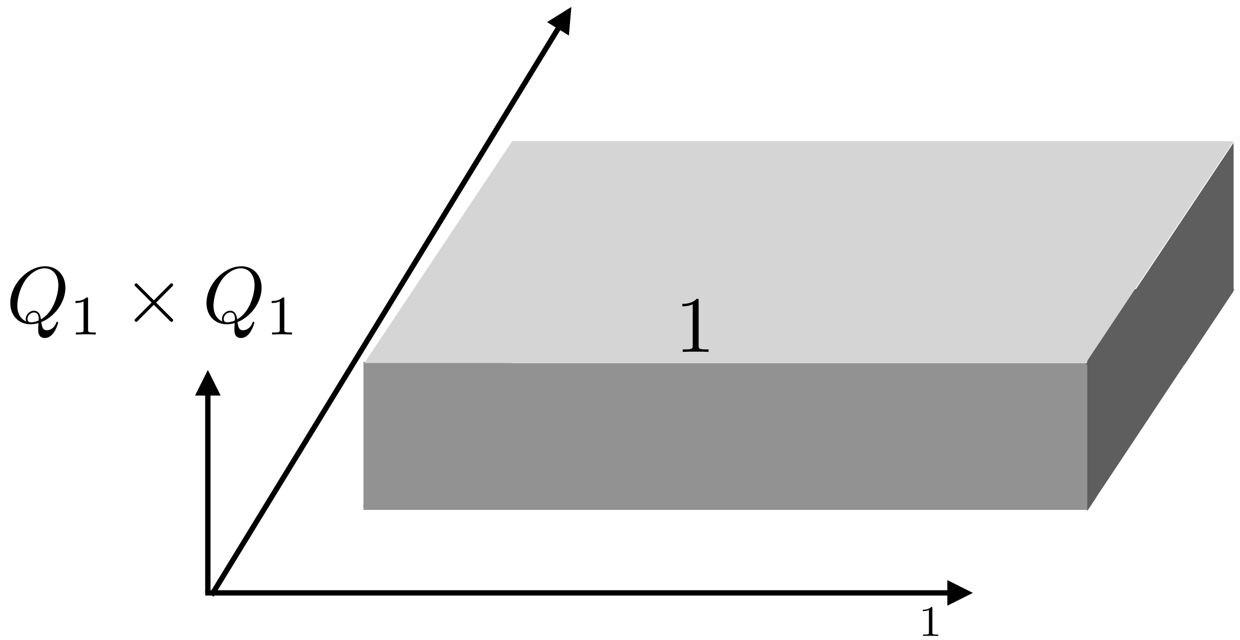


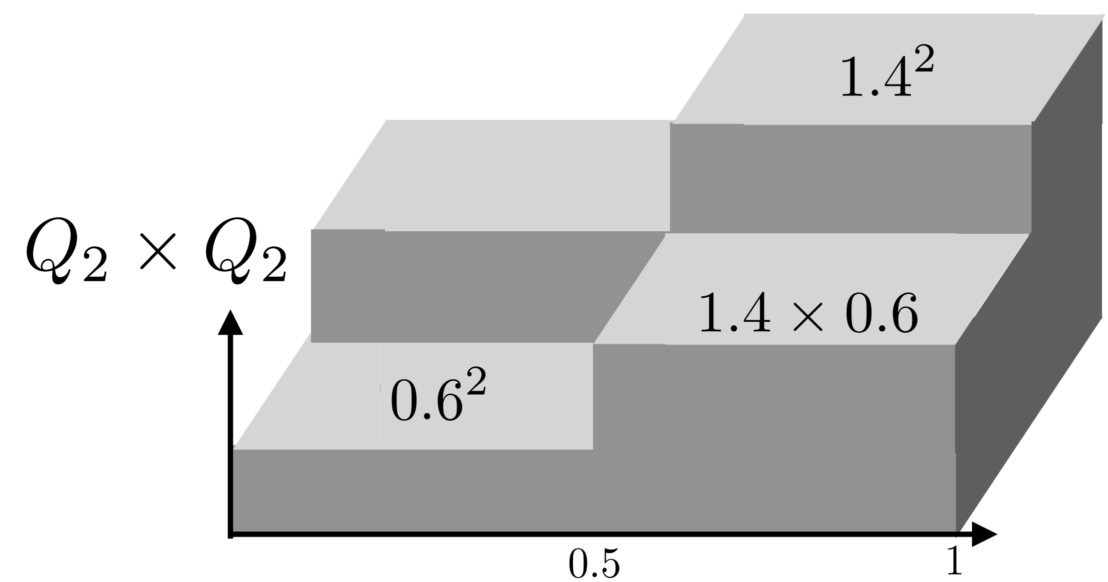




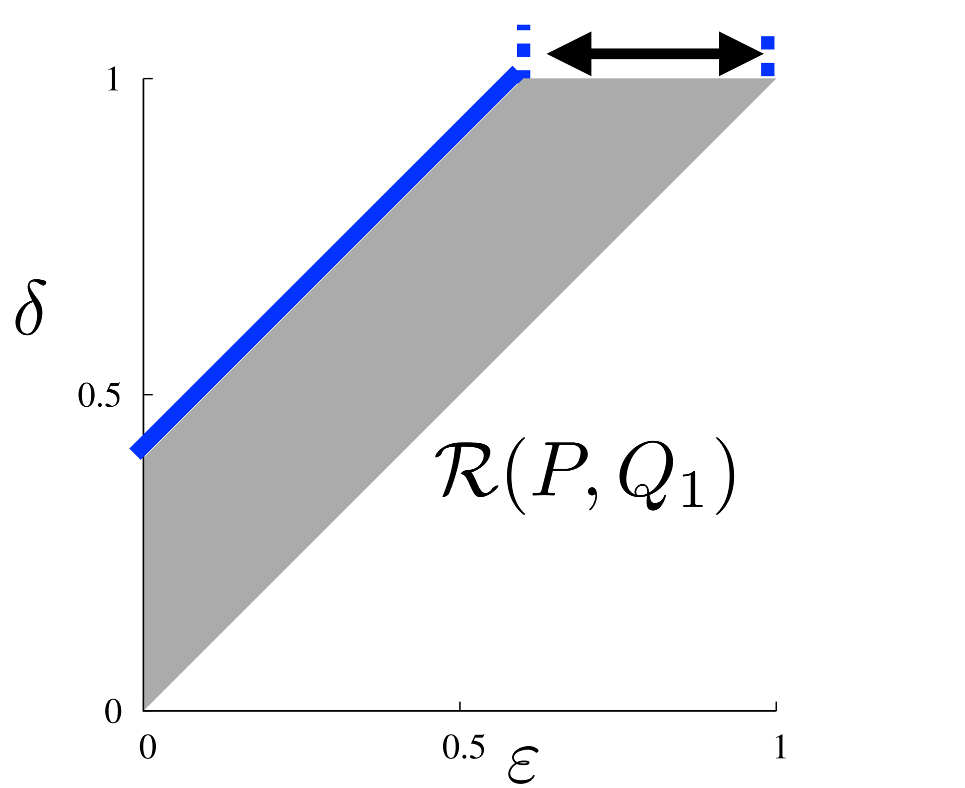


(reverse) data-processing inequality

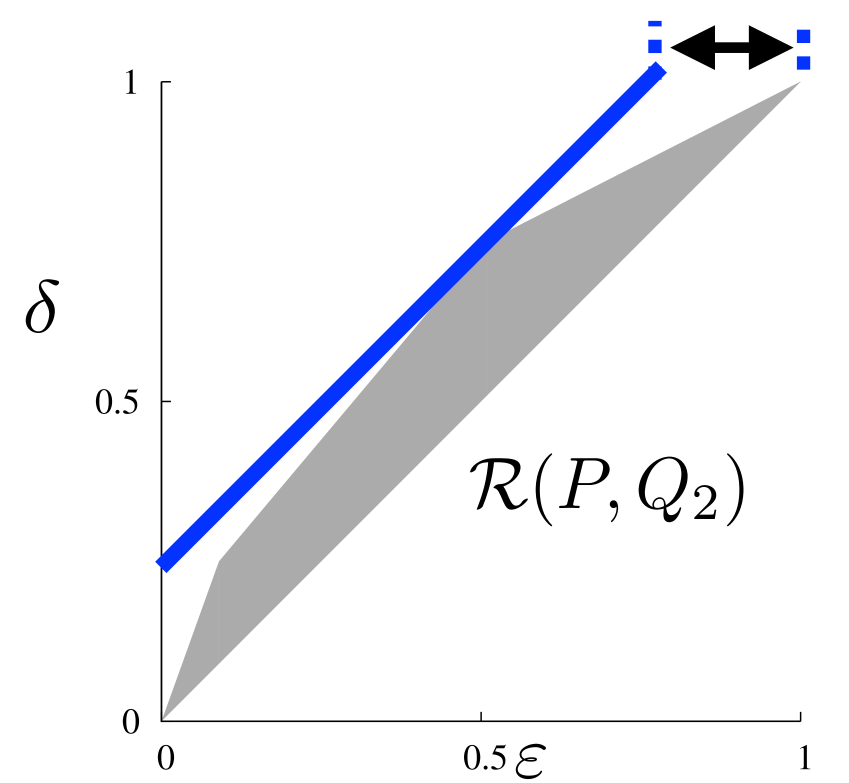




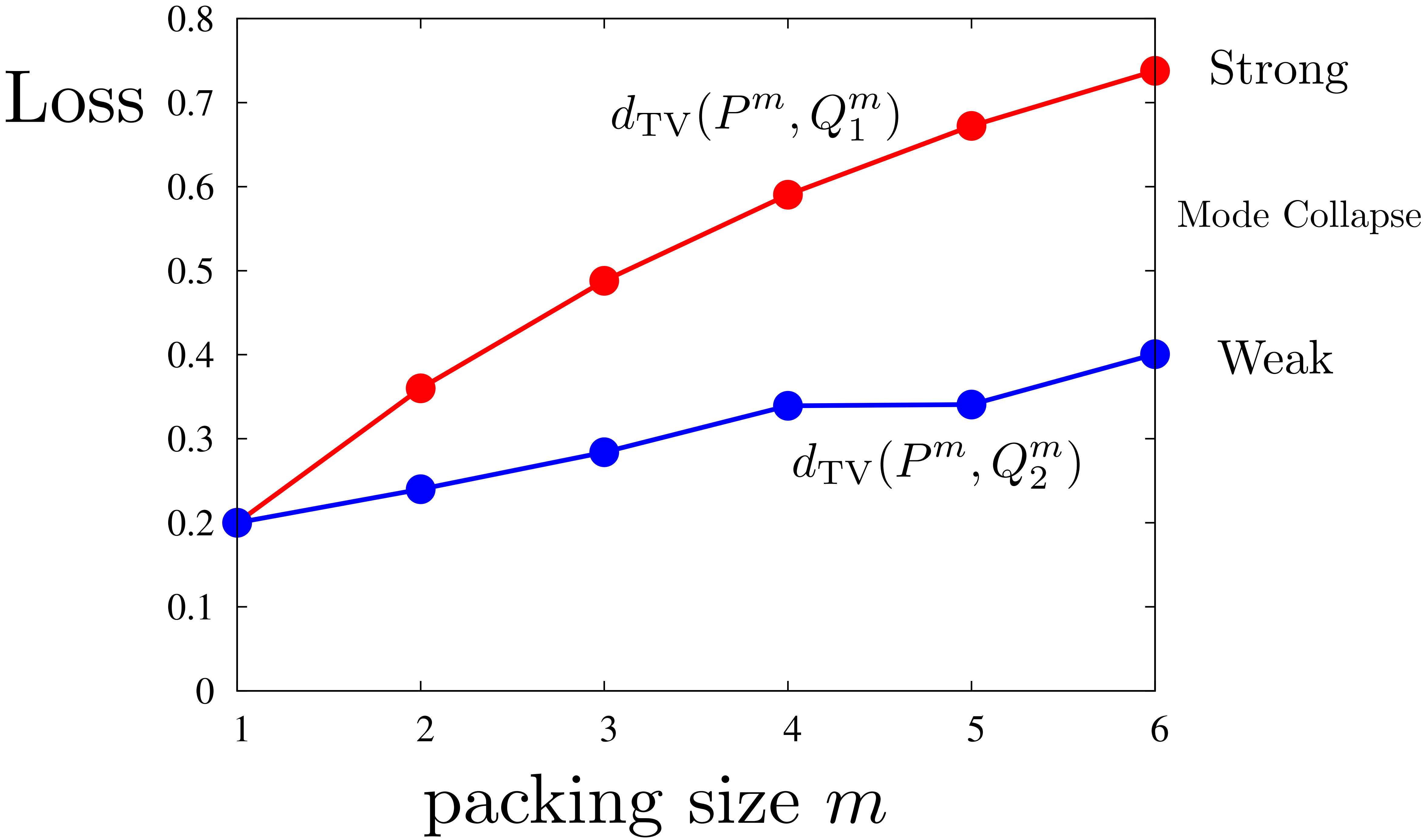
$$d_{\mathrm{TV}}(P^2, Q_1^2) = 0.36$$



$$d_{\mathrm{TV}}(P^2, Q_2^2) = 0.24$$



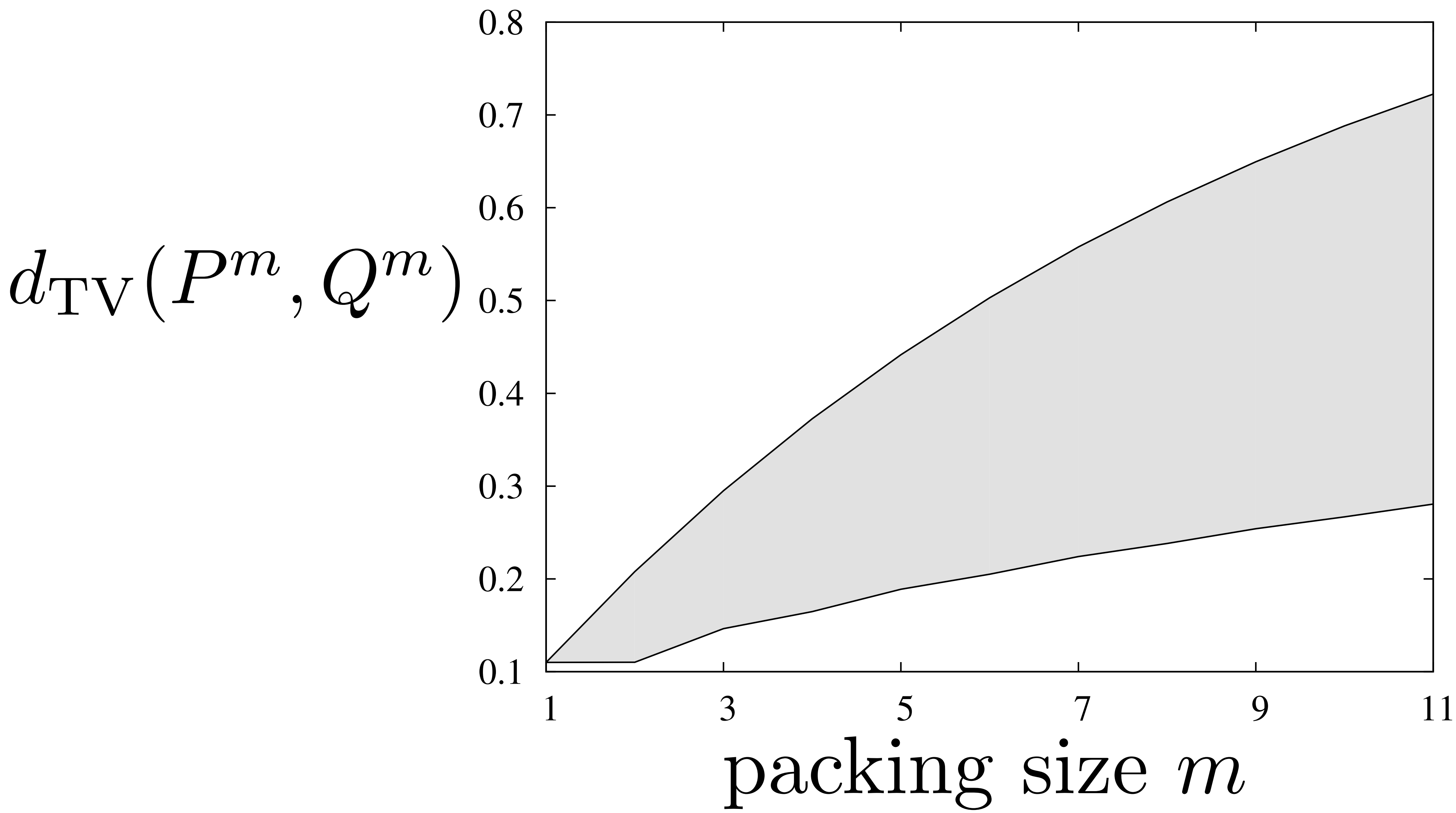
Lifting naturally penalizes Mode Collapse



Analysis of lifted TV

- Evolution of TV distances

$$\max_{P,Q} / \min_{P,Q} \quad d_{\mathrm{TV}}(P^m, Q^m)$$
$$\text{subject to} \quad d_{\mathrm{TV}}(P, Q) = \tau$$



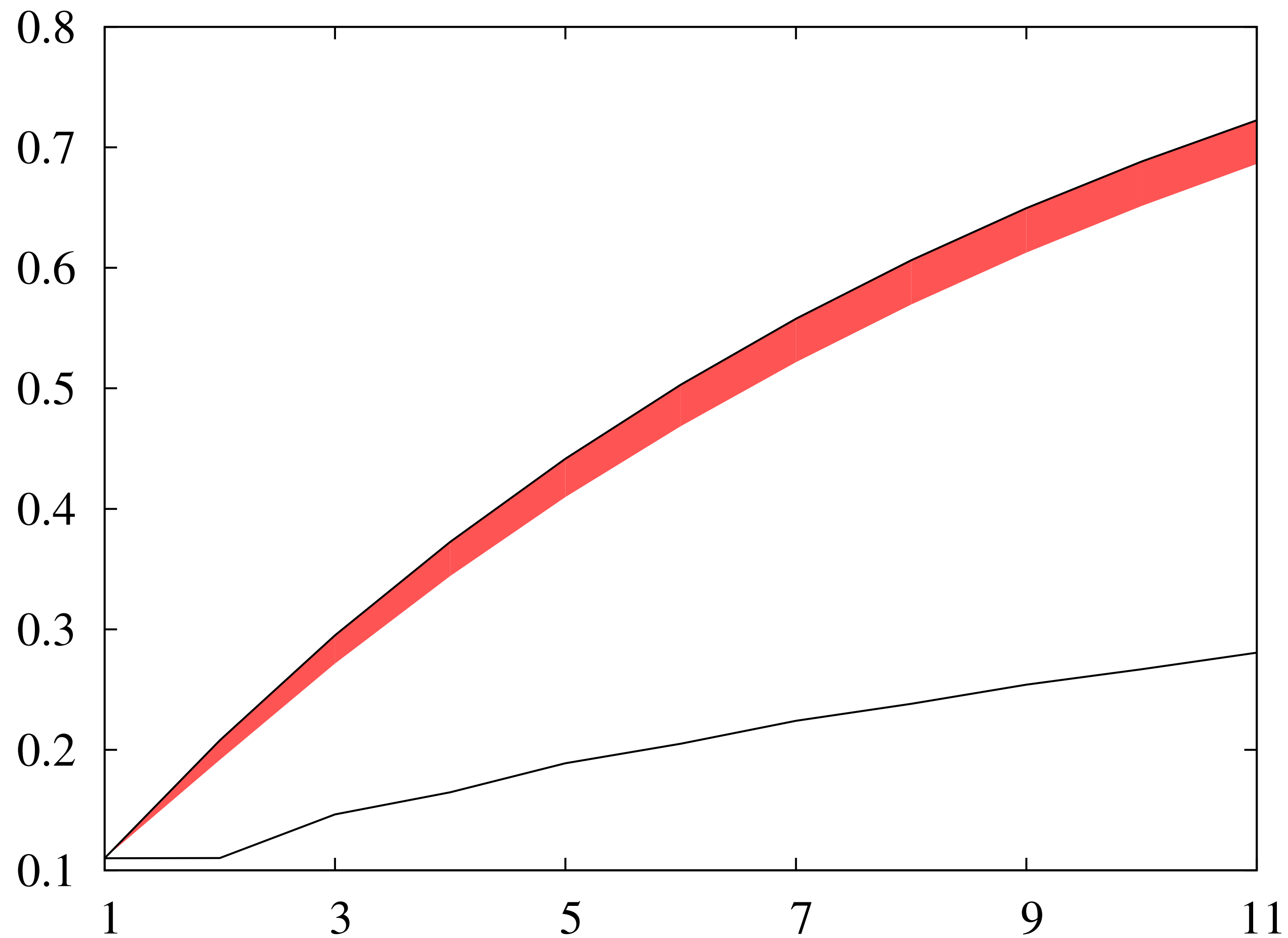
Analysis of lifted TV with Mode Collapse

$$\max_{P,Q} / \min_{P,Q} \quad d_{\mathrm{TV}}(P^m, Q^m)$$

$$\text{subject to} \quad d_{\mathrm{TV}}(P, Q) = \tau$$

$$\text{with } (\varepsilon_0, \delta_0)\text{-mode collapse}$$

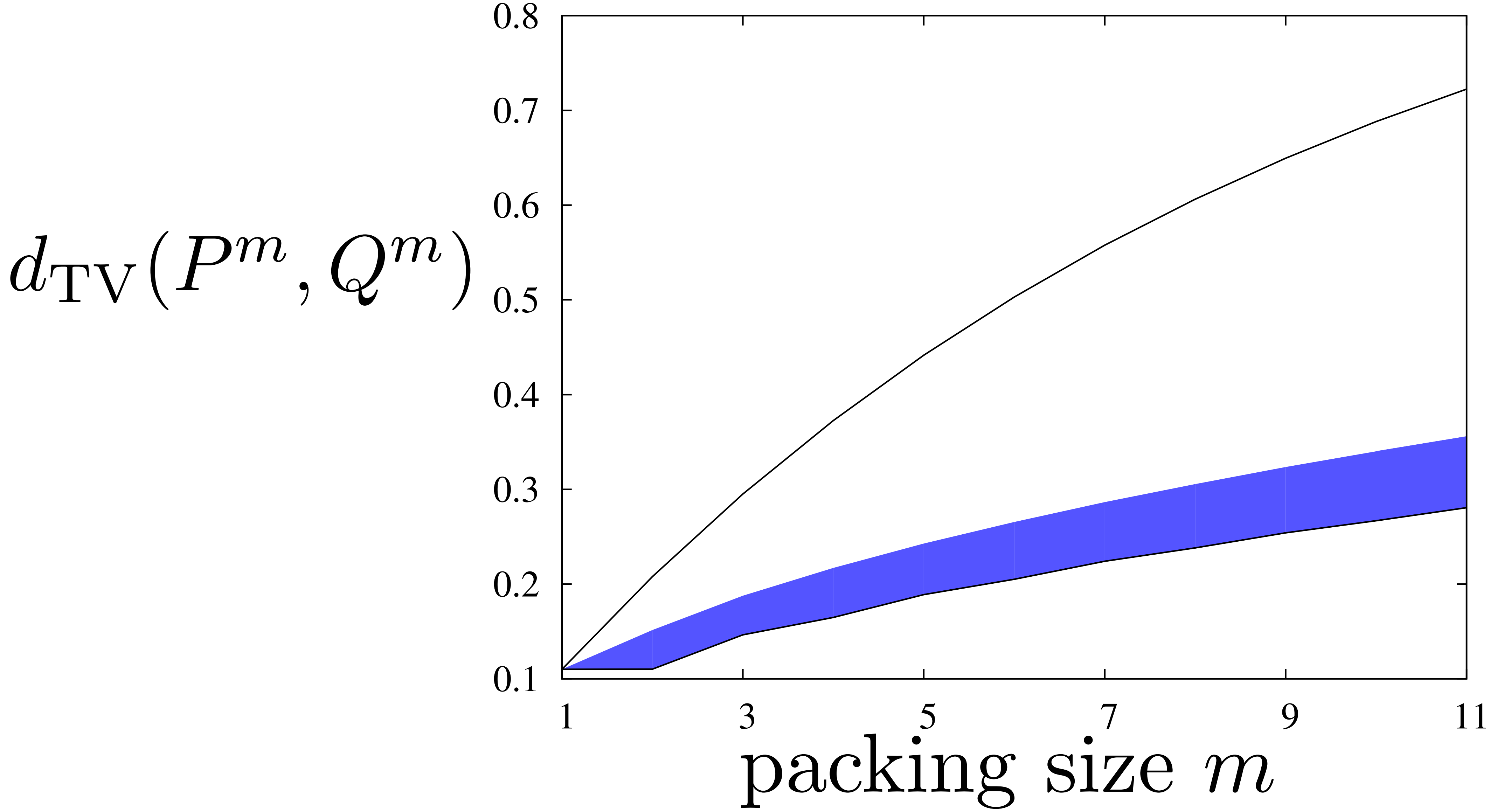
$d_{\mathrm{TV}}(P^m, Q^m)$



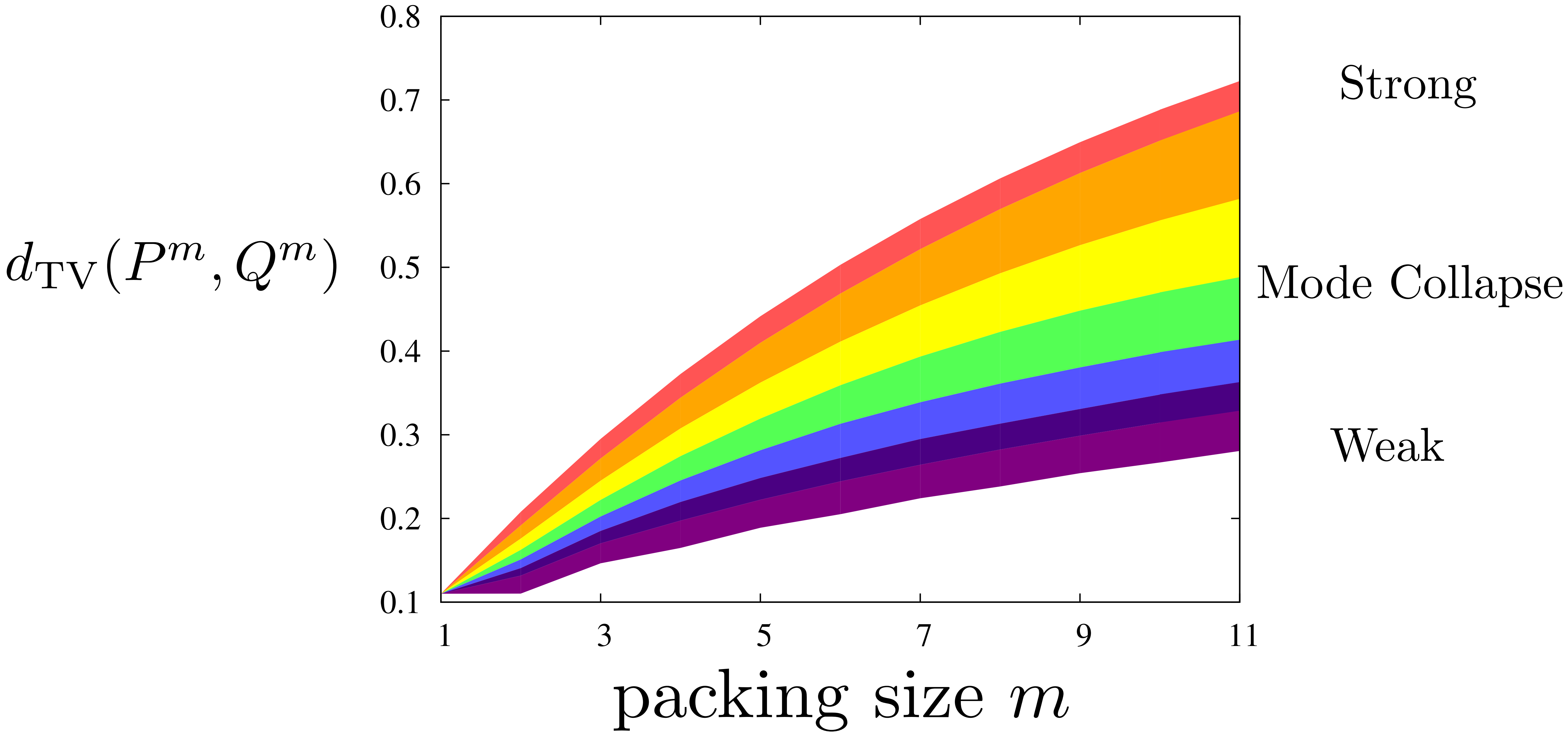
packing size m

Analysis of lifted TV with Mode Collapse

$$\max_{P,Q} / \min_{P,Q} \quad d_{\mathrm{TV}}(P^m, Q^m)$$
$$\text{subject to} \quad d_{\mathrm{TV}}(P, Q) = \tau$$
$$\text{without } (\varepsilon_0, \delta_0)\text{-mode collapse}$$



Analysis of lifted TV with Mode Collapse



Remaining challenges in Mode Collapse

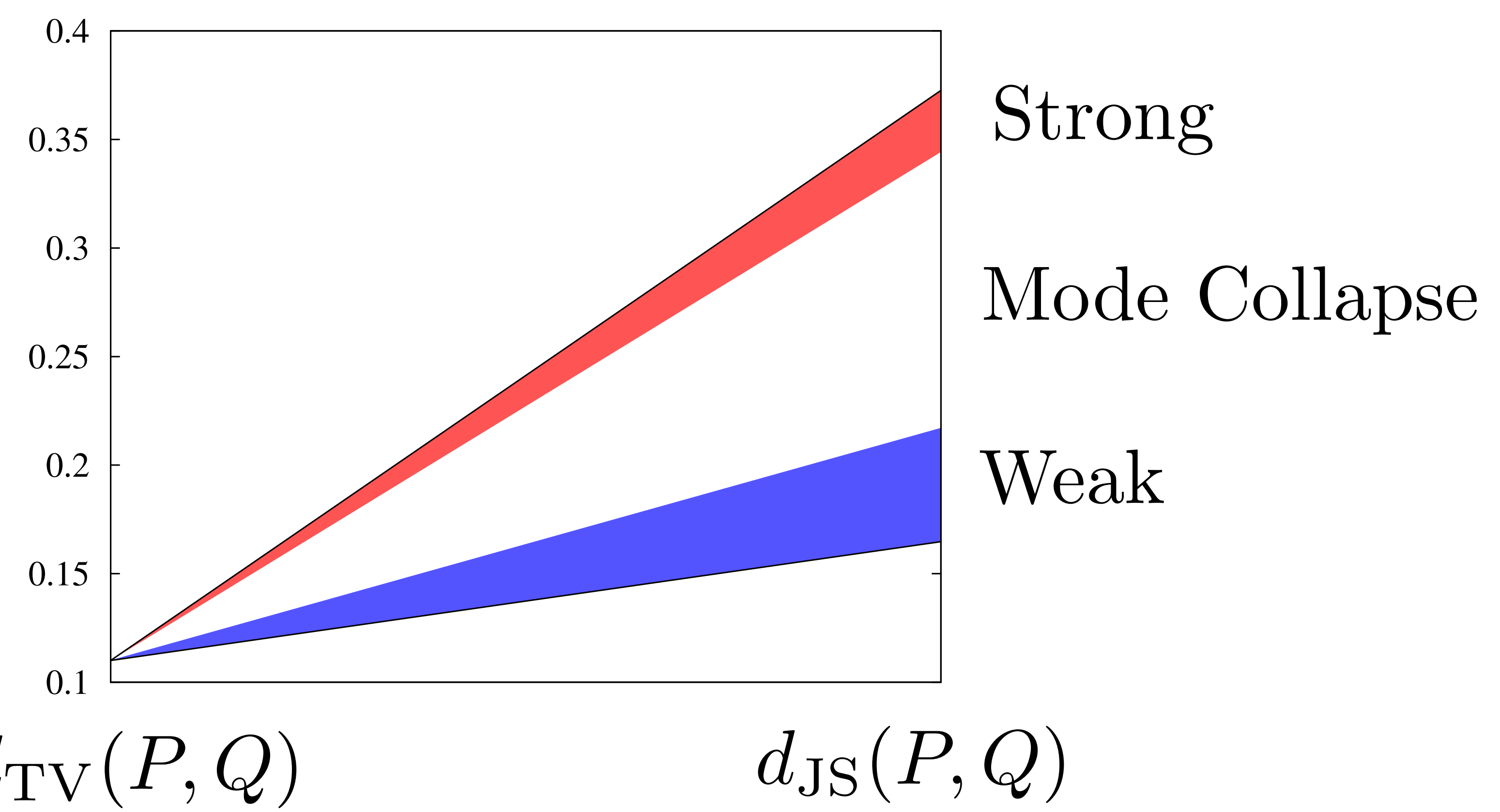
- There has been extensive effort on designing new losses for GANs, but empirically compared
- We give a formal comparisons of loss function

$$d_{\mathrm{TV}}(P,Q) \prec_{mode} d_{\mathrm{TV}}(P^m,Q^m)$$

- Can we formally compare other popular loss functions?

$$d_{\mathrm{TV}}(P,Q) \prec_{mode} d_{\mathrm{JS}}(P,Q)$$

$$\begin{array}{ll} \max_{P,Q} / \min_{P,Q} & d_{\mathrm{JS}}(P^m,Q^m) \\ \text{subject to} & d_{\mathrm{TV}}(P,Q)=\tau \\ \quad \text{with } (\varepsilon_0,\delta_0)\text{-mode collapse} & \end{array}$$



Generalization [Theorem4.1,BCST18]

Suppose D_w and G_θ are Lipschitz in $w \in W \subseteq \mathbb{R}^p$ and $\theta \in \Theta \subseteq \mathbb{R}^q$

$$\hat{\theta} \in \arg\min_{\theta\in\Theta} \max_{w\in W} \frac{1}{n}\sum_{i=1}^{n}\log(D_w(X_i)) + \frac{1}{n}\sum_{i=1}^{n}\log(1 - D_w(G_\theta(Z_i)))$$

$$\theta^* \in \arg\min_{\theta\in\Theta} D_{\mathrm{JS}}(P_{\mathrm{real}}\|P_\theta)$$

and for all $\theta \in \Theta$, there exists $w \in W$ such that $\|D_w - D^*(P_\theta)\|_\infty \le \varepsilon$

then $\mathbb{E}[D_{\mathrm{JS}}(P_{\mathrm{real}}\|P_{\hat{\theta}})] = \mathbb{E}[D_{\mathrm{JS}}(P_{\mathrm{real}}\|P_{\theta^*})] + O\Big(\varepsilon^2 + \sqrt{\frac{p+q}{n}}\Big)$

representation power of Θ

Lipschitz condition

representation power of W

[“Generalization and Equilibrium in Generative Adversarial Network”,Arora et al.,2017]
[“On the Discrimination-Generalization Tradeoff in GANs”,Zhang et al., 2017]
[“Some Theoretical Properties of GANs”,Biau,Cadre,Sangnier,Tanielian 2018]

Generalization [Arora et al. 17]

- Neural network generative modes are not Lipschitz in general. In one extreme, if we allow the generator to be chosen from any distribution, then GAN does not generalize in JS-divergence [Lemma 1, Arora et al.17].

$$D_{\text{JS}}(P_{\text{real}}, P_{\hat{\theta}}) = \log 2$$

  In other words, memorization or overfitting happens.

- However, they generalize in the loss (which is the property of the NN discriminator, and not the generator) [Theorem 3.1, Arora et al. 17]:

$$\boxed{\left|\mathcal{L}(P_{\text{real}}, P) - \hat{\mathcal{L}}(P_{\text{real}}, P)\right| \;=\; \tilde{O}\Big(\sqrt{\frac{p}{n}}\Big)}$$

with high probability

Lipschitz condition

Open questions in generalization

- Can we provide more fine grained generalization bounds that differentiate different choices of the **loss functions**?

- The analysis critically relies on Lipschitz condition. In practice, **regularizers** are commonly used in training the discriminator. Can generalization bounds help design new regularizers, and understand their roles?

- How do we solve the minimax optimization and **learn** $\hat{\theta}$?

Role of the discriminator for Gaussian [FSXT17]

- Some of the open equations are answered in LQG setting with Linear generator, Quadratic loss, and Gaussian distribution. If the discriminator is constrained to be quadratic function of the input, then [Theorem 3,FSXT17]
- $$\|\Sigma^* - \hat{\Sigma}\| \; = \; O\Big(\sqrt{\frac{d}{n}}\Big)$$
  $$\text{with high probability}$$
- However, for unconstrained discriminator [Theorem 2,FSXT17]

$$\|\Sigma^* - \hat{\Sigma}\| \; = \; O\Big(n^{-\frac{2}{d}}\Big)$$

- Discriminator of matching complexity is critical

[“Understanding GANs: the LQG Setting”,Feizi,Suh,Xia,Tse, 2017]

Open questions in the role of the discriminator

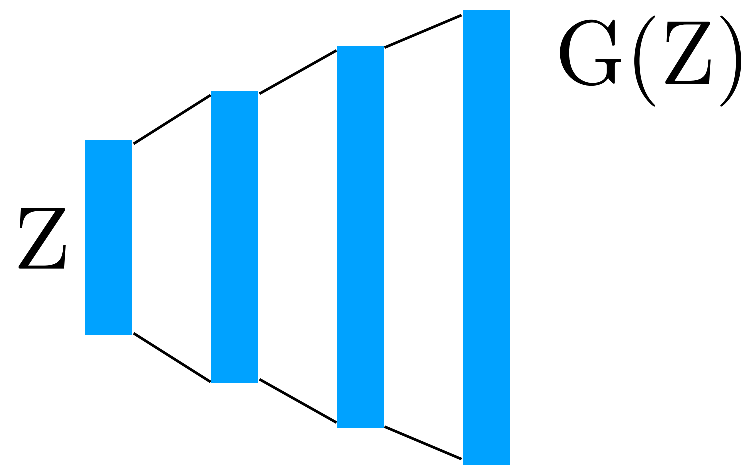
- What about mixture of two Gaussians?

- For Gaussian, constraining to linear generators reduces the problem to standard parameter learning (in this case the covariance matrix). For mixture of Gaussians, the counterpart is two linear generators with gating. However, this is further departure from the typical GAN.

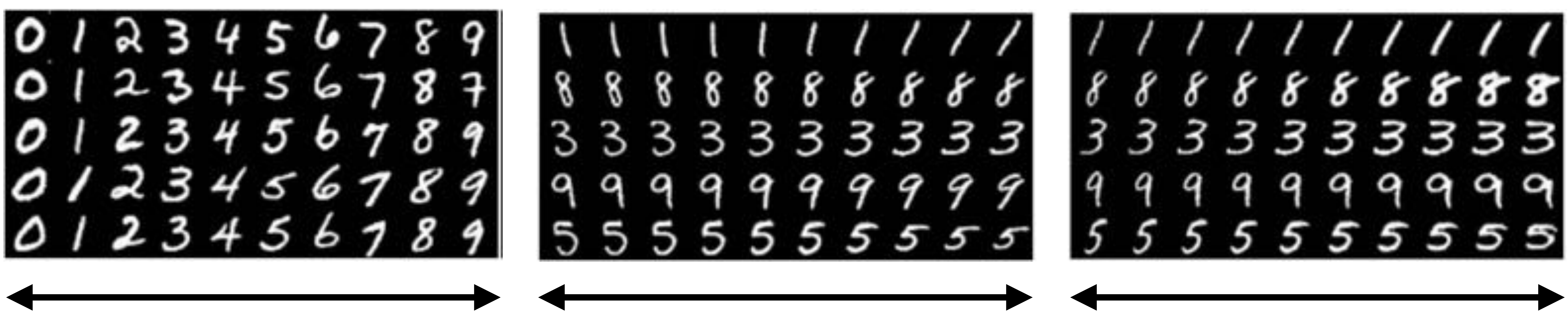
- At the discriminator, a counterpart will be tensor methods, which is only known to recover the mean of the mixtures and not the covariance matrices.

Interpretability / Disentangling Representation

- One weakness of GAN is that the latent variable Z has no interpretable meaning



- Ideally,



Z_1 : digits Z_2 : rotation Z_3 : width

[“InfoGAN: Interpretable Representation Learning by Information Maximizing GANs”,Chen et al., 2016]

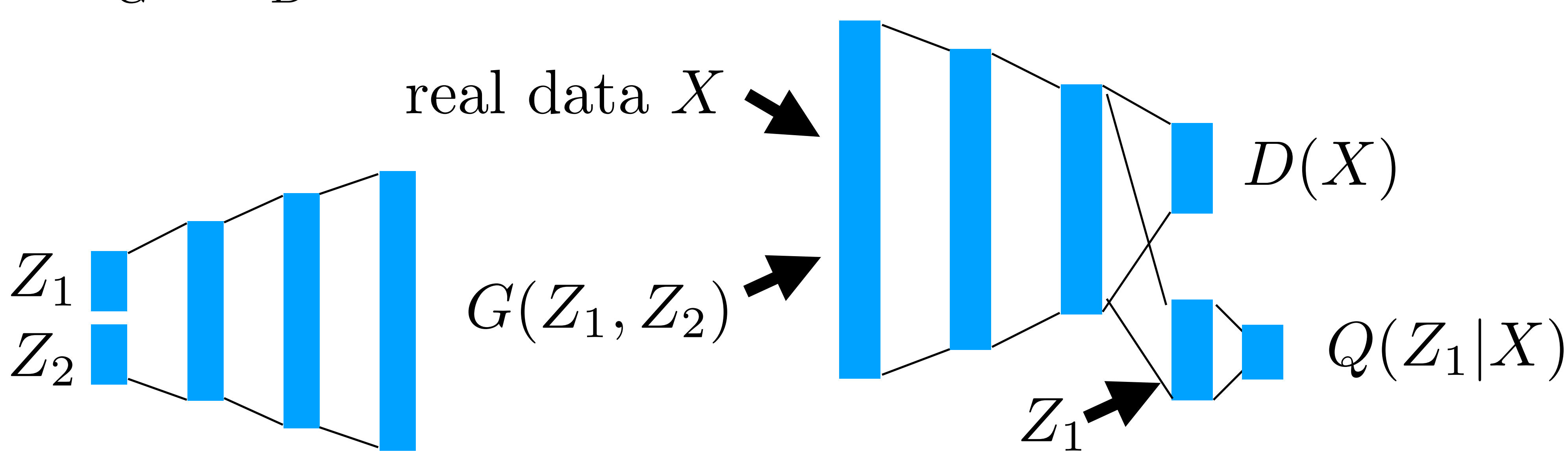
InfoGAN, Chen et al. 2016

- Proposes maximizing mutual information between the image $G(Z_1,Z_2)$ and a part of the latent representation Z_1

$$\min_G \max_D V(D,G) - \lambda I(Z_1; G(Z_1, Z_2))$$

- Challenge: minimizing (negative) mutual information Solution: Variational method to optimize over another neural network for $Q(Z_1|X)$

$$\min_G \max_D V(D,G) - \lambda \mathbb{E}_{X\sim G(Z_1,Z_2)}[\mathbb{E}_{\tilde{Z}_1\sim P(Z_1|X)}[\log Q(\tilde{Z}_1|X)]]$$



Summary

- Mode Collapse
  - [PacGAN: the power of two samples in generative adversarial networks, Lin,Khetan,Fanti,Oh,2017]
  - Theoretical understanding leads to the design of new principled architectures
- Generalization
  - Beginning of theoretical understanding of the tradeoffs involved
  - Potential to lead to new designs of loss and regularizers
- Interpretation
  - Powerful tool via mutual information
  - Theoretical understanding is missing

Collaborators

Ashish Khetan (Amazon AI)

Giulia Fanti (CMU)

Zinan Lin (CMU)

Kiran Thekumparampil (UIUC)

Organization: This Tutorial

Part-1: Deep learning for information theory

1a. Deep learning for communication

1b. Deep learning for statistical inference

Part-2: Information theory for deep learning

2a. Theory for GAN

2b. Learning Gated Neural Networks

Learning in Gated Neural Networks

Ashok Vardhan Makkuva (UIUC)
Sewoong Oh (UIUC)
Pramod Viswanath (UIUC)
Sreeram Kannan (UW, Seattle)

Gated Recurrent Neural Networks

- Well-known examples: LSTM and GRU
- State-of-the-art results in many challenging ML tasks

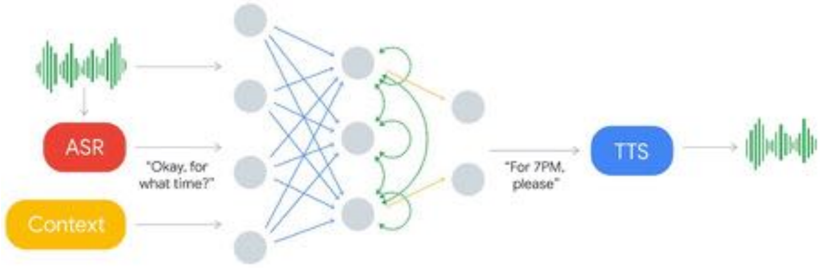


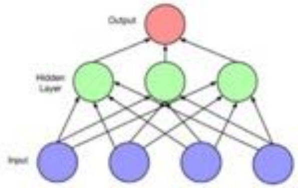
Figure: Google Duplex

Siri, Alexa and more...

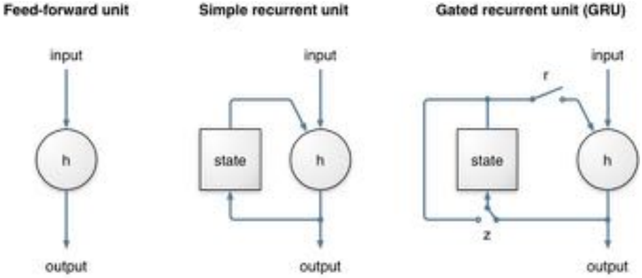
- Language translation
- Speech recognition
- Phrase completion

NNs and RNNs

- Feed-forward neural networks

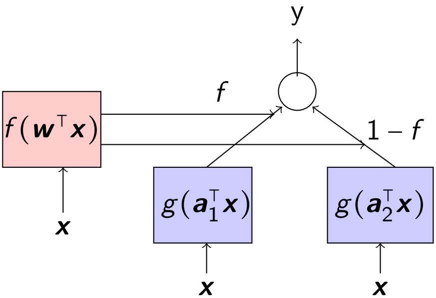


- Recurrent neural networks (Gating)



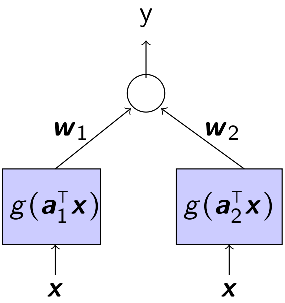
Mixture-of-Experts

- Jacobs, Jordan, Nowlan and Hinton, 1991 $f = \mathrm{sigmoid}$, $g = \mathrm{linear}, \tanh, \mathrm{ReLU}$

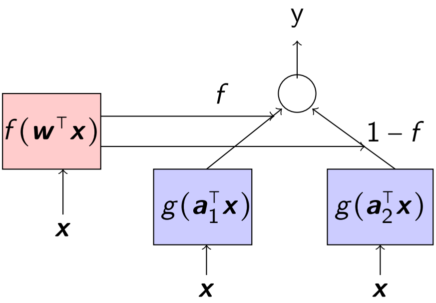


$f = \mathrm{sigmoid}$, $g = \mathrm{linear}, \tanh, \mathrm{ReLU}$

MoE generalizes 2-layer Neural Network



(a) 2-node NN



(b) 2-MoE

MoE: Modern relevance

- Outrageously large neural networks

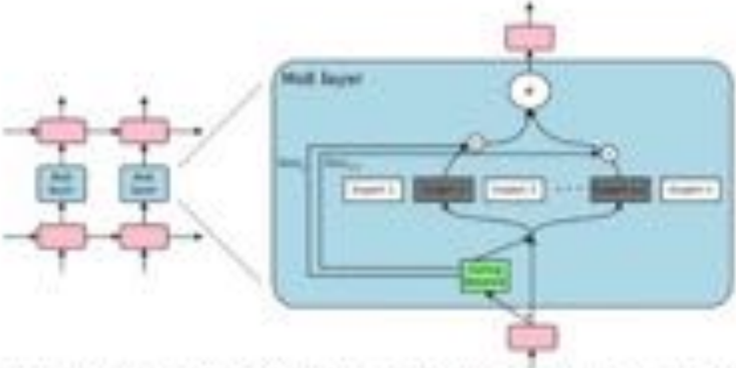


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

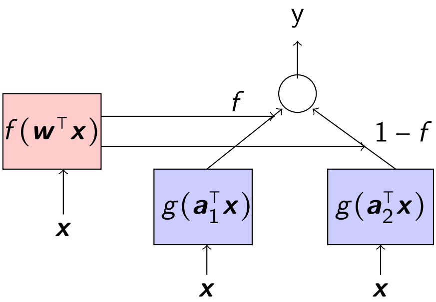
What is known about MoE?

Adaptive mixtures of local experts RA Jacobs, MI Jordan, SJ Nowlan, GE Hinton Neural computation 3 (1), 79-87	3663	1991
Sharing clusters among related groups: Hierarchical Dirichlet processes YW Teh, MI Jordan, MJ Beal, DM Blei Advances in neural information processing systems, 1385-1392	3273	2005
Hierarchical mixtures of experts and the EM algorithm MI Jordan, RA Jacobs Neural computation 6 (2), 181-214	3090	1994

- No provable learning algorithms for parameters[1] ☹

[1] 20 years of MoE, MoE: a literature survey

Open problem for 25+ years



$$\Leftrightarrow P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$

Open question

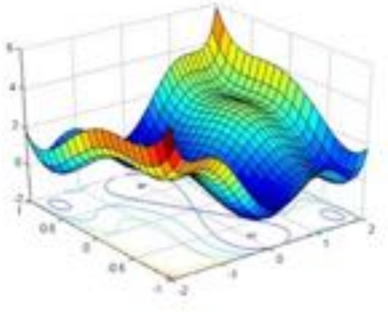
Given n i.i.d. samples $(\boldsymbol{x}^{(i)}, y^{(i)})$, does there exist an efficient learning algorithm with provable theoretical guarantees to learn the regressors $\boldsymbol{a}_1, \boldsymbol{a}_2$ and the gating parameter $\boldsymbol{w}$?

Gradient descent

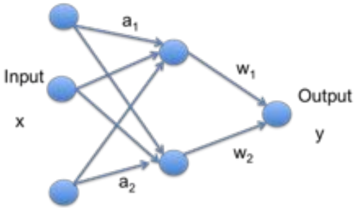
$$\min_\theta \mathbb{E} L(y, \psi_\theta(x)) \tag{1}$$

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta \mathbb{E} L(y, \psi_\theta(x)) \tag{2}$$

- If loss is convex in parameters, problem is easy.
- However, loss is highly non-convex



Fundamental Reason for Non-convexity



- Let w_1, w_2, a_1, a_2 be the true parameters.
- Permutation invariance:

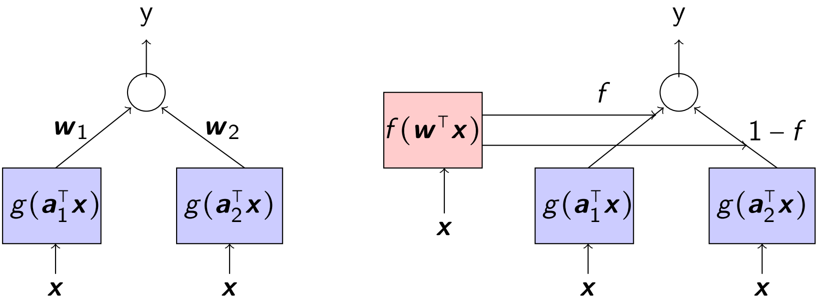
$$L(a_1, a_2, w_1, w_2) = L(a_2, a_1, w_2, w_1) \tag{3}$$

- If loss is convex, choosing all hidden nodes same is optimal!!!

$$L\left(\frac{a_1 + a_2}{2}, \frac{a_1 + a_2}{2}, \frac{w_1 + w_2}{2}, \frac{w_1 + w_2}{2}\right) = L(a_1, a_2, w_1, w_2) \tag{4}$$

- Loss cannot be convex in NN or MoE!

MoE vs. 2-layer Neural Network



(a) 2-node NN (b) 2-MoE

- MoE has both classifier and regressor!

MoE: Modern relevance

- Outrageously large neural networks

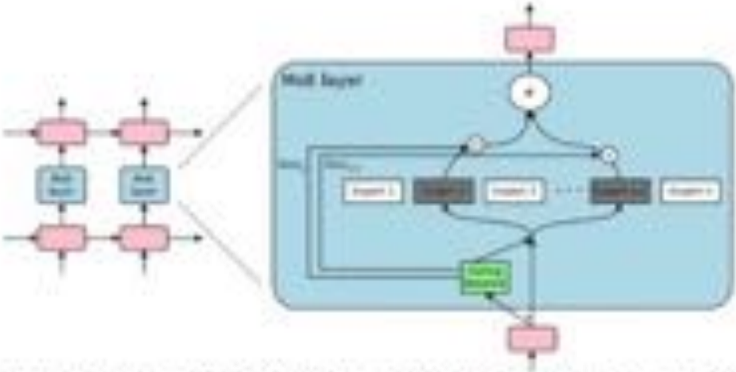
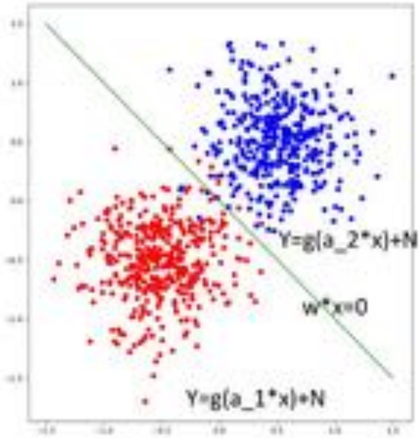


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

MoE: Modular structure



Key observation

If we know the regressors, learning the gating parameter is easy and vice-versa. How to break the gridlock?

Focus of this talk: Breaking the gridlock

- **First** learning guarantees for MoE
- Two novel approaches to learn the parameters:

Method 1: Beyond gradient descent

Novel algorithm with first recoverable guarantees

Method 2: Change the loss function

Non-trivial loss function for which GD optimal

- Both approaches work with **global initializations**
  - restriction: $\boldsymbol{x}$ is Gaussian

Generalizability

k-MoE

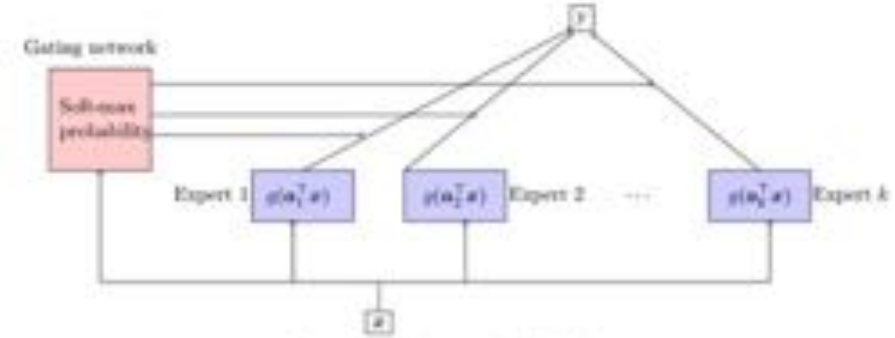


Figure 1: Architecture for k-MoE

Generalizability

Hierarchical mixture of experts (HME)

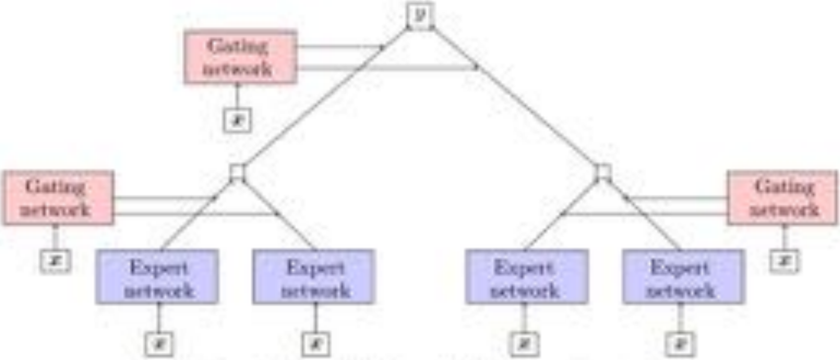


Figure 2: A two-level hierarchical mixture of experts

Method 1: Design of algorithms

Algorithmic approach: Simplified model

Model for MoE:

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$

Without gating:

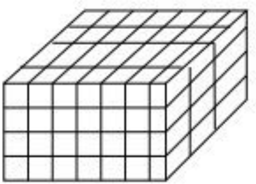
$$P_{y|\boldsymbol{x}} = p \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - p) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$

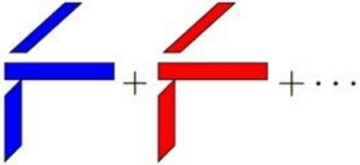
- Mixture of generalized linear models (GLMs)!
  - Similar to 2-layer NN
  - How do we learn $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ without knowing p?
  - Method of moments [Sedghi, Janzamin and Anandkumar '16]

Method of moments in GLMs

- Basic idea [Sedghi et al '16]: Construct a **third-order super-symmetric** tensor from data such that

$$\mathbb{E}(\psi(X,Y)) = \sum_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{a}_i \Rightarrow \boldsymbol{a}_i \text{ can be recovered}$$





- How do we construct ψ?
    - Stein's lemma

Stein's lemma 101

Stein's lemma

For $f : \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$,

$$\mathbb{E}[f(\boldsymbol{x}) \cdot \boldsymbol{x}] = \mathbb{E}[\nabla_{\boldsymbol{x}} f(\boldsymbol{x})] \in \mathbb{R}^d.$$

Non-linear regression using Stein's lemma: If $y = g(\boldsymbol{a}_1^\top \boldsymbol{x}) + N$, then

$$\begin{aligned}
\underbrace{\mathbb{E}[y \cdot \boldsymbol{x}]}_{\text{Estimated from samples}} &= \mathbb{E}[g(\boldsymbol{a}_1^\top \boldsymbol{x}) \cdot \boldsymbol{x}] + \underbrace{\mathbb{E}[N \cdot \boldsymbol{x}]}_{=0} \\
&= \mathbb{E}[\nabla_{\boldsymbol{x}} g(\boldsymbol{a}_1^\top \boldsymbol{x})] \\
&\propto \boldsymbol{a}_1
\end{aligned}$$

Mixture of GLMs: Stein's lemma 101

- Recall, for mixture of GLMs:

$$P_{y|\boldsymbol{x}} = p \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1-p) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$

- From Stein's lemma,

$$\mathbb{E}[y \cdot \boldsymbol{x}] \propto p \cdot \boldsymbol{a}_1 + (1-p) \cdot \boldsymbol{a}_2.$$

- Not unique in $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$
- How can we ensure uniqueness?

Stein's lemma 102

2nd order Stein's lemma

$$\mathbb{E}[f(\boldsymbol{x}) \cdot \underbrace{(\boldsymbol{x}\boldsymbol{x}^\top - I)}_{\mathcal{S}_2(\boldsymbol{x})}] = \mathbb{E}[\nabla_{\boldsymbol{x}}^{(2)} f(\boldsymbol{x})] \in \mathbb{R}^{d \times d}.$$

- Mixture of GLMs:

$$P_{y|\boldsymbol{x}} = p \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1-p) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$

$$\Rightarrow \mathbb{E}[y \cdot (\boldsymbol{x}\boldsymbol{x}^\top - I)] \propto 2p \cdot \boldsymbol{a}_1 \boldsymbol{a}_1^\top + 2(1-p) \cdot \boldsymbol{a}_2 \boldsymbol{a}_2^\top.$$

- Not unique!
- How can we ensure uniqueness?

Stein's lemma 103

3rd order Stein's lemma

$$\mathbb{E}[f(\boldsymbol{x}) \cdot \mathcal{S}_3(\boldsymbol{x})] = \mathbb{E}[\nabla_{\boldsymbol{x}}^{(3)} f(\boldsymbol{x})] \in \mathbb{R}^{d \times d \times d}$$

- Score transformation $\mathcal{S}_3(\boldsymbol{x}) = \boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x} - \sum_{i \in [d]} \mathrm{sym}(\boldsymbol{x} \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_i)$

- Mixture of GLMs:

$$P_{y|\boldsymbol{x}} = p \cdot \mathcal{N}(y | g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1-p) \cdot \mathcal{N}(y | g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$
$$\Rightarrow \mathbb{E}[y \cdot \mathcal{S}_3(\boldsymbol{x})] \propto p \cdot \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + (1-p) \cdot \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_2.$$

- Unique! (by Kruskal's theorem)
- Note: LHS estimated from samples!

MoE: Stein's lemma

- For MoE, $p = p(x) = f(\boldsymbol{w}^\top \boldsymbol{x})$ since

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$$

- Can we use Stein's lemma to learn $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$?
- Natural attempt:

$$\mathbb{E}[y \cdot S_3(\boldsymbol{x})] = \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + \boldsymbol{w} \otimes \boldsymbol{a}_1 \otimes \boldsymbol{w} + \ldots + \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{w} + \ldots$$

  Not a super-symmetric tensor

- Can we construct a super-symmetric tensor for MoE?

Key insight: Hermite polynomial transformation

Suppose g =linear and $\sigma = 0$. Then

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathbb{1}\{y = \boldsymbol{a}_1^\top \boldsymbol{x}\} + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))\mathbb{1}\{y = \boldsymbol{a}_1^\top \boldsymbol{x}\}$$

$$\Rightarrow \mathbb{E}[y^3 - 3y|\boldsymbol{x}] = \sum_{i \in \{1,2\}} f(\boldsymbol{w}_i^\top \boldsymbol{x})((\boldsymbol{a}_i^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_i^\top \boldsymbol{x})), \quad \boldsymbol{w}_2 = -\boldsymbol{w}_1$$

Now applying Stein's lemma,

$$\mathbb{E}[(y^3 - 3y) \cdot \mathcal{S}_3(\boldsymbol{x})] = \mathbb{E}[\nabla_{\boldsymbol{x}}^3 \mathbb{E}[y^3 - 3y|\boldsymbol{x}]] = 3 \sum_{i \in \{1,2\}`} \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{a}_i$$

How do cross terms like $\boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{w}$ disappear?

- Reason: $\mathbb{E}[H_3'(Z)] = \mathbb{E}[H_3''(Z)] = \mathbb{E}[H_3'''(Z)] = 0$
- $H_3(z) = z^3 - 3z$ is third-Hermite polynomial

Does this work for $\sigma \neq 0$?

Linear experts: Hermite-like-polynomials

Suppose g = linear and $\sigma \neq 0$:

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|\boldsymbol{a}_1^\top \boldsymbol{x}, \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|\boldsymbol{a}_2^\top \boldsymbol{x}, \sigma^2)$$

Super-symmetric tensor

$$\mathcal{T}_3 = \mathbb{E}[(y^3 - 3y(1+\sigma^2)) \cdot \mathcal{S}_3(\boldsymbol{x})] = 3(\boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_2)$$

- This very much needs special linear structure. What about other non-linearities for g?

Generalization: Cubic polynomial transformations

- For a wide class of non-linearities such as g=linear, sigmoid, ReLU, etc.

$$\mathcal{T}_3 = \mathbb{E}[(y^3 + \alpha y^2 + \beta y) \cdot \mathcal{S}_3(\boldsymbol{x})] = c(\boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_2)$$

- How do we choose α and β?
  - Solving a linear system
  - **Example:** For sigmoid,

$$\begin{bmatrix} 0.2067 & 0.2066 \\ 0.0624 & -0.0001 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -0.1755 - 0.6199\sigma^2 \\ -0.0936 \end{bmatrix}$$

- **Key idea:** Acts like a 'Hermite' like polynomial for general g and cancels cross terms

Learning regressors: Spectral decomposition

Algorithm

- Input: Samples $(\boldsymbol{x}_i, y_i)$
- Compute $\hat{\mathcal{T}}_3 = (1/n) \sum_i H_3(y_i) \cdot \mathcal{S}_3(\boldsymbol{x}_i)$
- $\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2$ = Rank-2 decomposition on $\mathcal{T}_3$

Learning the gating

- Recall

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|\boldsymbol{a}_1^\top \boldsymbol{x}, \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|\boldsymbol{a}_2^\top \boldsymbol{x}, \sigma^2)$$

- If we know $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, learning $\boldsymbol{w}$ is a classification problem!
- Traditional methods:
  - EM algorithm
  - Gradient descent on log-likelihood

Theoretical contributions

- Show global convergence for existing methods
- Provide convergence rate
- Finite sample complexity
- First theoretical guarantees

Learning the gating parameters

$\hat{Y}$

Suppose spectral methods give $\hat{\boldsymbol{a}}_i$ with $\|\hat{\boldsymbol{a}}_i - \boldsymbol{a}_i\|_2 \le \sigma^2\varepsilon$

For high SNR, i.e. $\sigma < \sigma_0$, σ_0 is a dimension independent constant:

- EM iterates converge geometrically to $\hat{\boldsymbol{w}}$
- Convergence rate is a dimension-independent constant depending on σ and $\|\boldsymbol{a}_1 - \boldsymbol{a}_2\|$
- $\hat{\boldsymbol{w}}$ is ε-close to the ground truth

Method 2: Optimization framework-loss function design

Loss function design: Paradigm

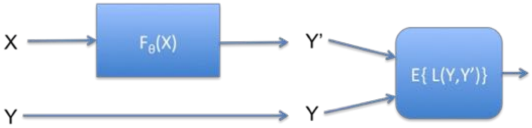


Figure: Standard Loss function architecture

- Standard approaches Get stuck in local minima, no theoretical analysis, and use single loss function
- Modify the architecture to design a loss function g
  - Building on [R.Ge, J.D. Lee, T. Ma '18]

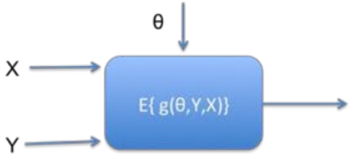
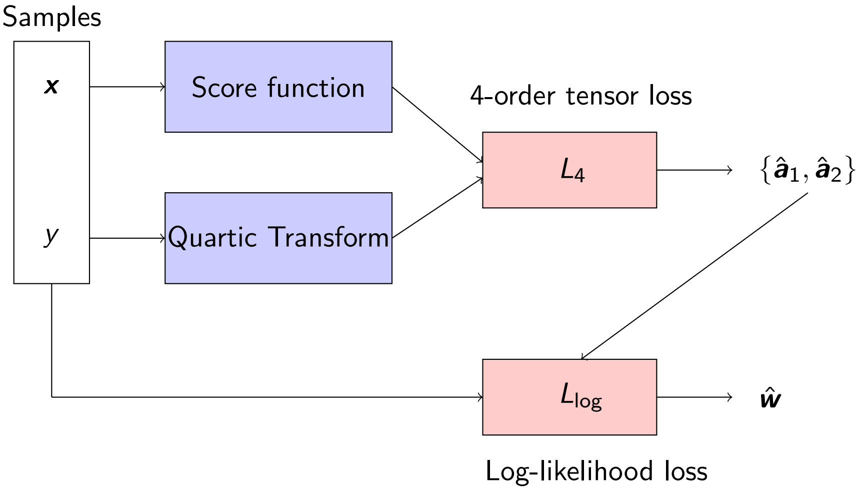


Figure: Modified Loss function architecture

Main contributions

- Separate loss functions L_4 and $L_{\log}$ to learn $(\boldsymbol{a}_1, \boldsymbol{a}_2)$ and $\boldsymbol{w}$



- Gradient descent on both L_4 and $L_{\log}$. What are they?

Tensor based loss function for regressors

- For linear experts,

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|\boldsymbol{a}_1^\top \boldsymbol{x}, \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|\boldsymbol{a}_2^\top \boldsymbol{x}, \sigma^2)$$

- Stein's lemma+ 4-Hermite polynomial implies

$$\mathcal{T}_4 = \mathbb{E}[(y^4 - 6y^2(1+\sigma^2)) \cdot \mathcal{S}_4(\boldsymbol{x})] = 12(\boldsymbol{a}_1^{\otimes 4} + \boldsymbol{a}_2^{\otimes 4})$$

- If $\hat{\boldsymbol{a}}_1$ and $\hat{\boldsymbol{a}}_2$ are parameters,

$$L_4(\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2) \triangleq \sum_{j \neq k} \mathcal{T}_4(\hat{\boldsymbol{a}}_j, \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{a}}_k, \hat{\boldsymbol{a}}_k) - \mu \sum_{j \in \{1,2\}} \mathcal{T}_4(\hat{\boldsymbol{a}}_j, \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{a}}_j) + \lambda \sum_{j \in \{1,2\}} (\|\hat{\boldsymbol{a}}_j\|^2 - 1)^2$$

Landscape of L_4

Properties

- No spurious local minima: All local minima are global
- Global minima are ground truth (upto permutation and sign-flip)
- All saddle points have negative curvature
- SGD converges to approximate global minima

Why L_4?

Summary

- Algorithmic innovation: First provably consistent algorithms for MoE in 25+ years
- Loss function innovation: First SGD based algorithm on novel loss functions with provably nice landscape properties
- Sample complexity: First sample complexity results for MoE
- Global convergence: Our algorithms work with global initializations

Open questions

- Generalizing to non-Gaussian inputs
  - Results: In the absence of gating, we have a loss function framework to provably learn the regressors
  - With gating?
- Learning algorithms for time-series?
- Learning algorithms and sample complexity for deep neural networks.

Thanks for support

Thank you!