Information Flow in Deep Neural Networks (DNNs)

Ghassen Zafzouf

Department of Electronic and Electrical Engineering
University of Melbourne

Information Learning & Processing
October 22, 2020

Outline

Motivation

- Unprecedented practical success in a various number of tasks
- Lacking theory
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
- In [Goldfeld et al. '19][1], the authors used Shannon information to understand how DNNs behave.

[1] Goldfeld, Z., Van Den Berg, E., Greenewald K., Melnyk I., Nguyen N. and Kingsbury B., (2019) "Estimating Information Flow in Deep Neural Networks", *Proceedings of the 36th International Conference on Machine Learning*, 2019

Motivation

- Unprecedented practical success in a various number of tasks
- Lacking theory
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
- In [Goldfeld et al. '19][1], the authors used Shannon information to understand how DNNs behave.

[1] Goldfeld, Z., Van Den Berg, E., Greenewald K., Melnyk I., Nguyen N. and Kingsbury B., (2019) "Estimating Information Flow in Deep Neural Networks", *Proceedings of the 36th International Conference on Machine Learning*, 2019

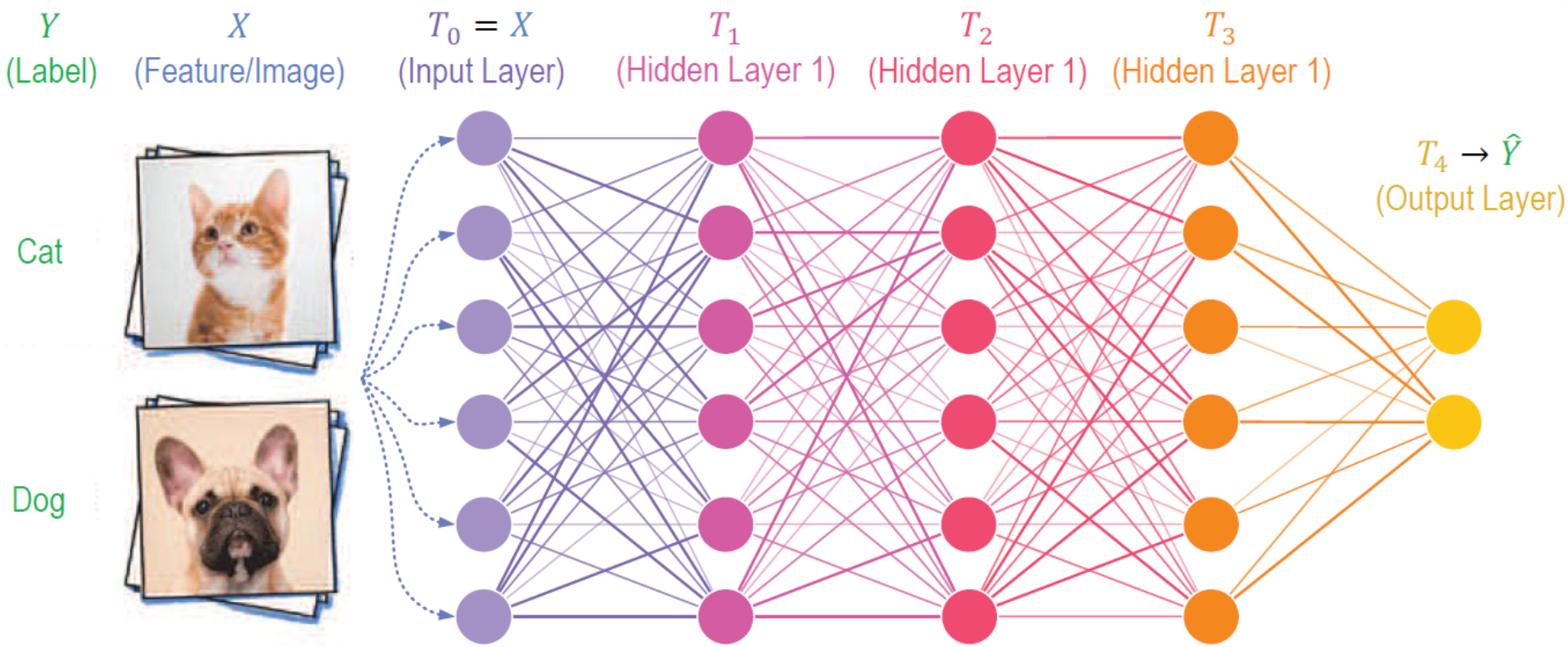
Motivation

- Unprecedented practical success in a various number of tasks
- Lacking theory
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
- In [Goldfeld et al. '19][1], the authors used Shannon information to understand how DNNs behave.

[1] Goldfeld, Z., Van Den Berg, E., Greenewald K., Melnyk I., Nguyen N. and Kingsbury B., (2019) "Estimating Information Flow in Deep Neural Networks", *Proceedings of the 36th International Conference on Machine Learning*, 2019

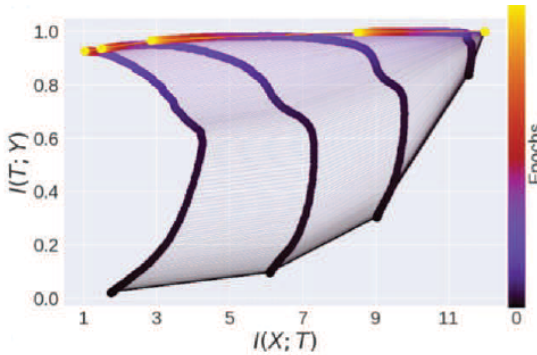
Setup & Preliminaries

Feedforward DNN for Classification:



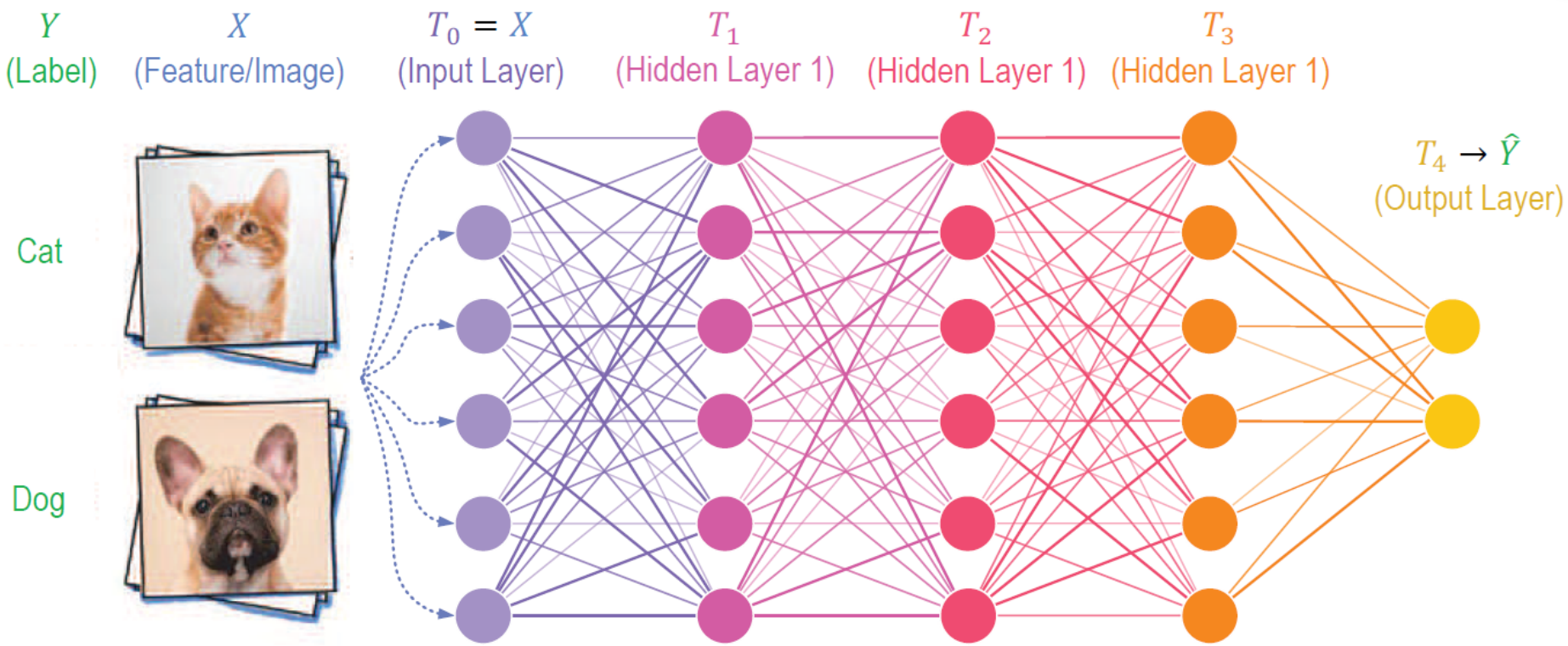
Claim: Training comprises 2 phases

- **Fitting:** $I(Y;T_\ell)$ and $I(X;T_\ell)$ increase (short)
- **Compression:** $I(X;T_\ell)$ slowly decreases (long)



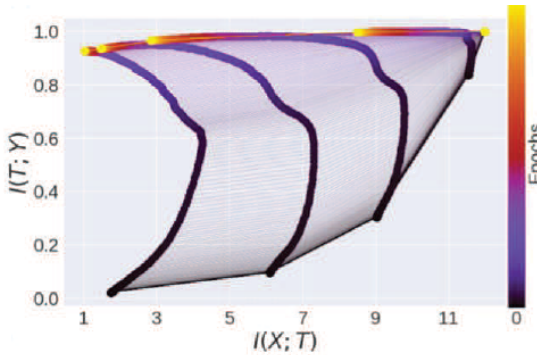
Setup & Preliminaries

Feedforward DNN for Classification:



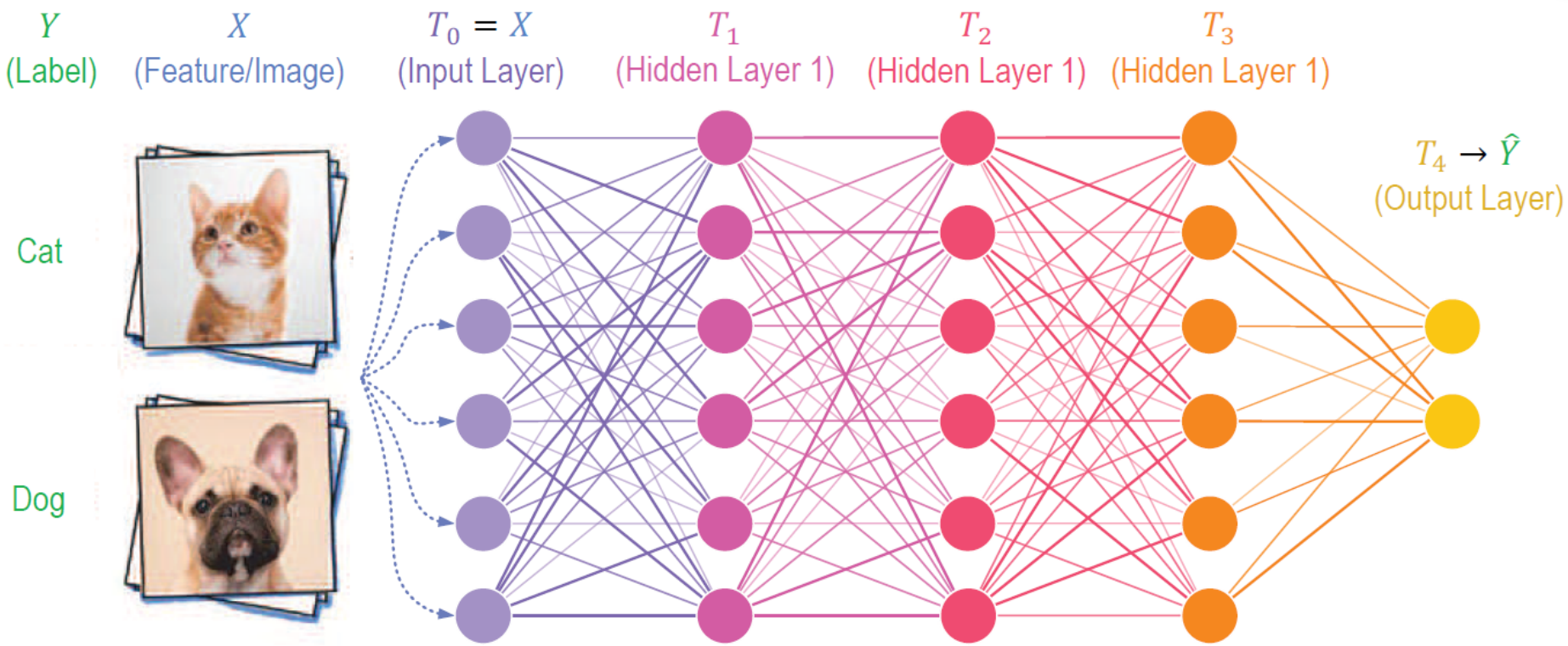
Claim: **Training comprises 2 phases**

- **Fitting:** $I(Y;T_\ell)$ and $I(X;T_\ell)$ increase (short)
- **Compression:** $I(X;T_\ell)$ slowly decreases (long)



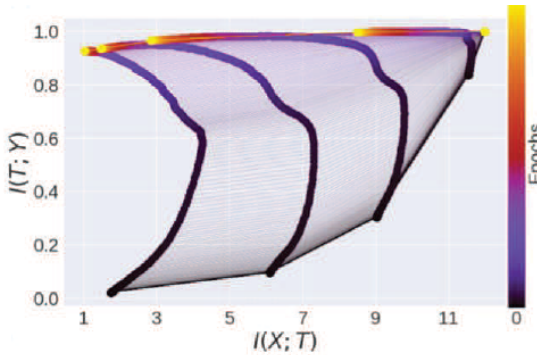
Setup & Preliminaries

Feedforward DNN for Classification:



Claim: **Training comprises 2 phases**

- **Fitting:** $I(Y;T_\ell)$ and $I(X;T_\ell)$ increase (short)
- **Compression:** $I(X;T_\ell)$ slowly decreases (long)



Outline

Mutual Information in Deterministic DNNs

Observation

Deterministic DNNs with strictly monotone nonlinearities such as tanh or sigmoid

$\implies$ $I(X;T_\ell)$ is independent of the DNN parameters

Why?

- **Continuous** X**:** $I(X;T_\ell) = h(T_\ell) - h(f_\ell(X)|X) = \infty$
- **Discrete** X**:** The mapping $X \mapsto T_\ell$ is injective $\Rightarrow$ $I(X;T_\ell) = H(X)$

Mutual Information in Deterministic DNNs

Observation

Deterministic DNNs with strictly monotone nonlinearities such as tanh or sigmoid

$\implies$ $I(X;T_\ell)$ is independent of the DNN parameters

Why?

- **Continuous** X**:** $I(X;T_\ell) = h(T_\ell) - h(f_\ell(X)|X) = \infty$
- **Discrete** X**:** The mapping $X \mapsto T_\ell$ is injective $\Rightarrow$ $I(X;T_\ell) = H(X)$

Mutual Information in Deterministic DNNs

Observation

Deterministic DNNs with strictly monotone nonlinearities such as tanh or sigmoid

$\implies$ $I(X;T_\ell)$ is independent of the DNN parameters

Why?

- **Continuous** X**:** $I(X;T_\ell) = h(T_\ell) - h(f_\ell(X)|X) = \infty$
- **Discrete** X**:** The mapping $X \mapsto T_\ell$ is injective $\Rightarrow$ $I(X;T_\ell) = H(X)$

Mutual Information in Deterministic DNNs

Observation

Deterministic DNNs with strictly monotone nonlinearities such as tanh or sigmoid

$\implies$ $I(X;T_\ell)$ is independent of the DNN parameters

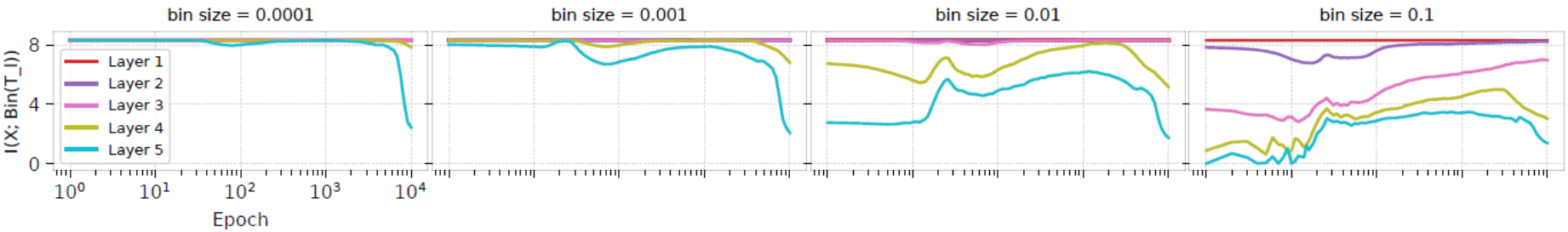
Why?

- **Continuous** X**:** $I(X;T_\ell) = h(T_\ell) - h(f_\ell(X)|X) = \infty$
- **Discrete** X**:** The mapping $X \mapsto T_\ell$ is injective $\Rightarrow$ $I(X;T_\ell) = H(X)$

What is going on?

- Plots via binning-based estimator of $I(X;T_\ell)$, for $X \sim \texttt{Unif}(\text{dataset})$

$\Longrightarrow$ Plotted values are $I(X;\texttt{Bin}(T_\ell)) \overset{??}{\approx} I(X;T_\ell)$ **No!**



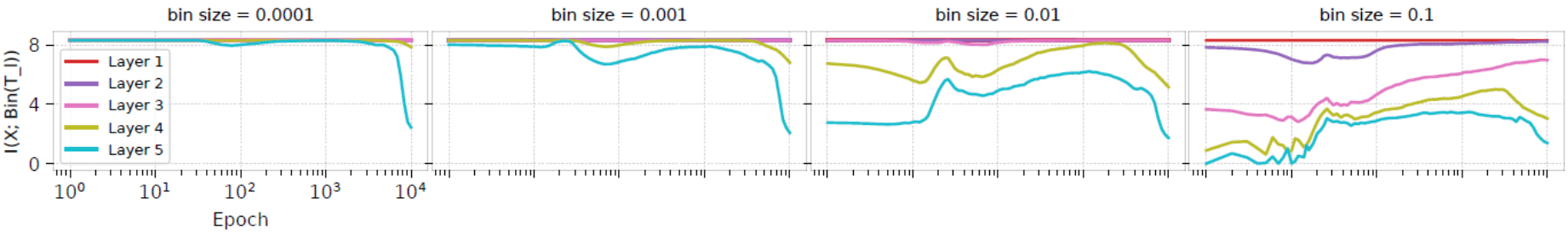
- Smaller bins $\Longrightarrow$ better accuracy: $I(X;T_\ell) = \ln(2^{12}) \approx 8.31$ bits
- Binning introduces "noise" into estimator (not present in the DNN)
- Plots showing estimation errors

$\Rightarrow$ **Problem:** $I(X;T_\ell)$ is meaningless in deterministic DNNs.

What is going on?

- Plots via binning-based estimator of $I(X;T_\ell)$, for $X \sim \texttt{Unif}(\text{dataset})$

  $\Longrightarrow$ Plotted values are $I(X;\texttt{Bin}(T_\ell)) \overset{??}{\approx} I(X;T_\ell)$ **No!**



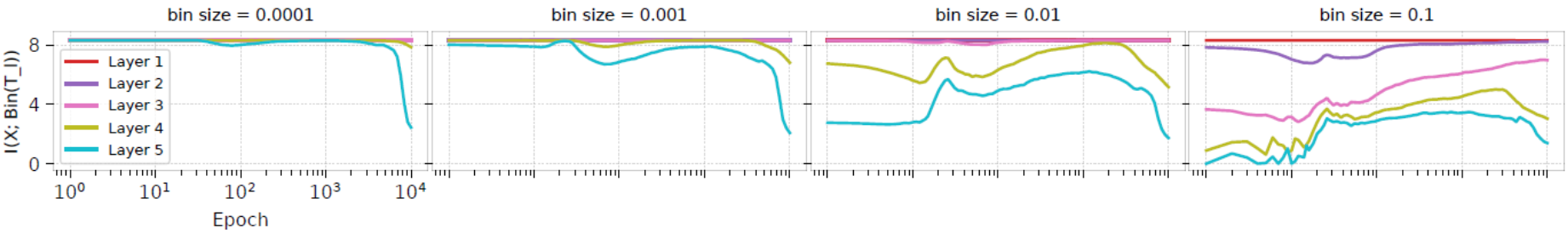
- Smaller bins $\Longrightarrow$ better accuracy: $I(X;T_\ell) = \ln(2^{12}) \approx 8.31$ bits
- Binning introduces "noise" into estimator (not present in the DNN)
- Plots showing estimation errors

$\Rightarrow$ **Problem:** $I(X;T_\ell)$ is meaningless in deterministic DNNs.

What is going on?

- Plots via binning-based estimator of $I(X;T_\ell)$, for $X \sim \texttt{Unif}(\text{dataset})$

  $\Longrightarrow$ Plotted values are $I(X;\texttt{Bin}(T_\ell)) \overset{??}{\approx} I(X;T_\ell)$ **No!**



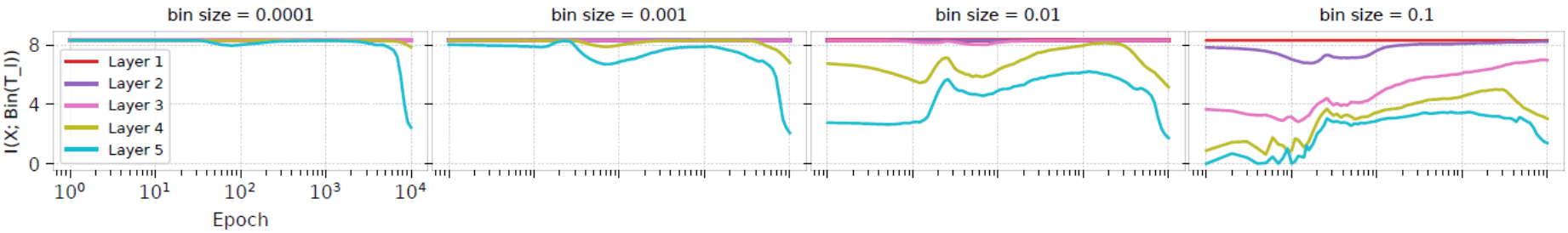
- Smaller bins $\Longrightarrow$ better accuracy: $I(X;T_\ell) = \ln(2^{12}) \approx 8.31$ bits
- Binning introduces "noise" into estimator (not present in the DNN)
- Plots showing estimation errors

$\Rightarrow$ **Problem:** $I(X;T_\ell)$ is meaningless in deterministic DNNs.

What is going on?

- Plots via binning-based estimator of $I(X;T_\ell)$, for $X \sim \texttt{Unif}(\text{dataset})$

  $\Longrightarrow$ Plotted values are $I(X;\texttt{Bin}(T_\ell)) \overset{??}{\approx} I(X;T_\ell)$ **No!**



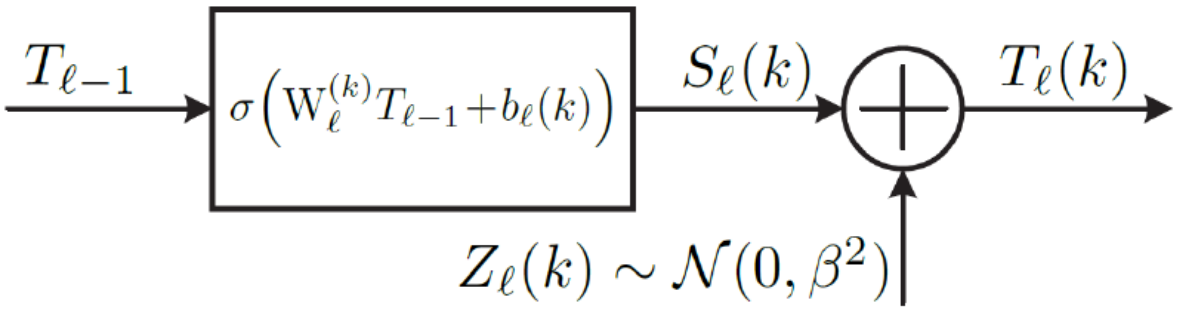
- Smaller bins $\Longrightarrow$ better accuracy: $I(X;T_\ell) = \ln(2^{12}) \approx 8.31$ bits
- Binning introduces "noise" into estimator (not present in the DNN)
- Plots showing estimation errors

$\Rightarrow$ **Problem:** $I(X;T_\ell)$ is meaningless in deterministic DNNs.

New Framework - Noisy DNNs

Idea: Inject some Gaussian noise to the layers' output

- **Model:** $T_\ell = f_\ell(T_{\ell-1}) + Z_\ell$, where $Z_\ell \sim \mathcal{N}(0, \beta^2 I)$



$\Rightarrow$ $X \mapsto T_\ell$ is a **parametrized channel** that depends on the system's parameters

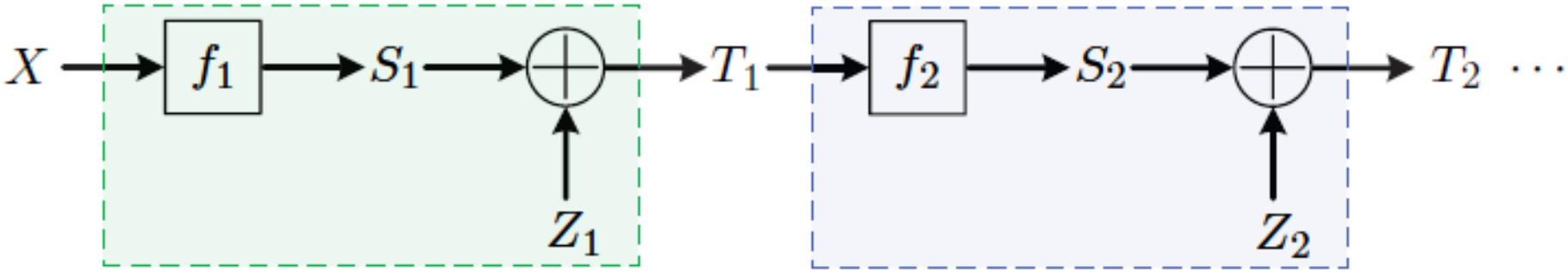
$\Rightarrow$ $I(X; T_\ell)$ is a **function** of weights and biases

- **In practice:** Performance & learned representations ($\beta \approx 10^{-1}$) comparable to deterministic DNNs

Outline

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



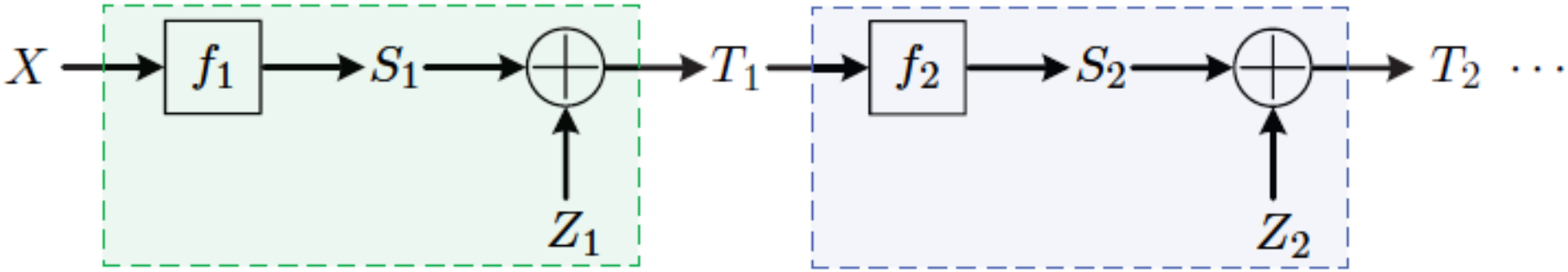
- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X; T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell | X = x_i)$

- P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate
- Both distributions can be sampled via the DNN forward pass
  - Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
  - Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



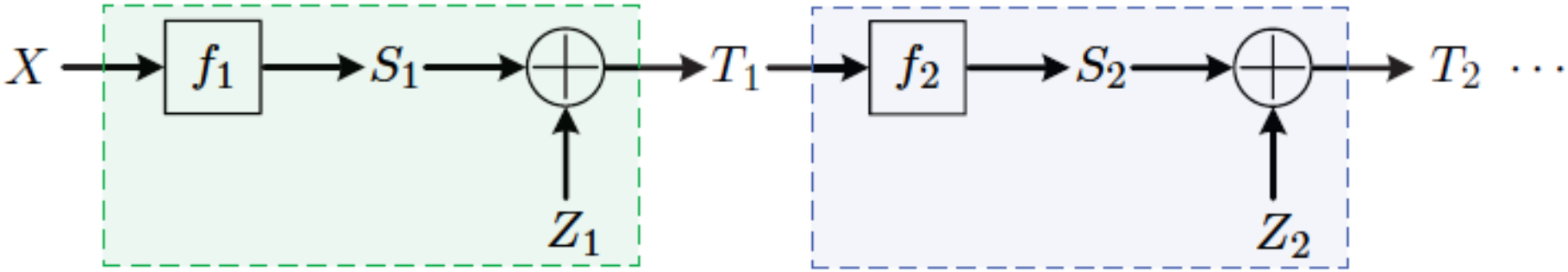
- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X;T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell|X=x_i)$

- P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate
- Both distributions can be sampled via the DNN forward pass
  - Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
  - Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



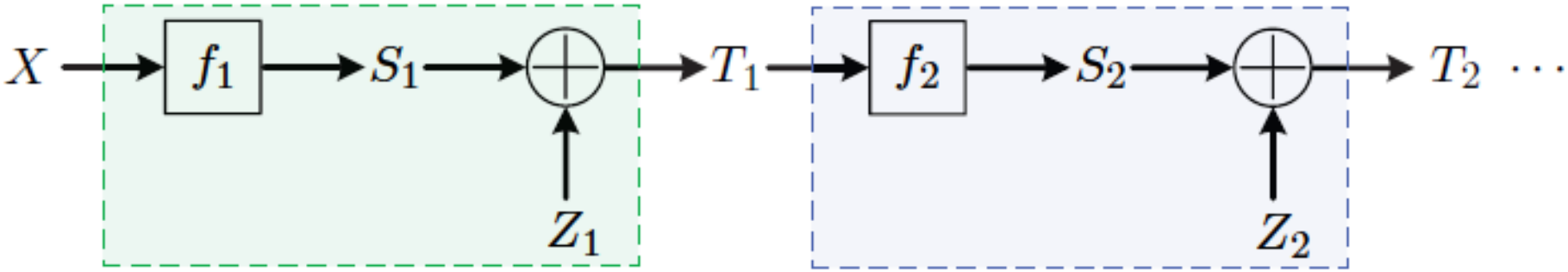
- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X; T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell | X = x_i)$

- P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate
- Both distributions can be sampled via the DNN forward pass
  - Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
  - Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X; T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell | X = x_i)$

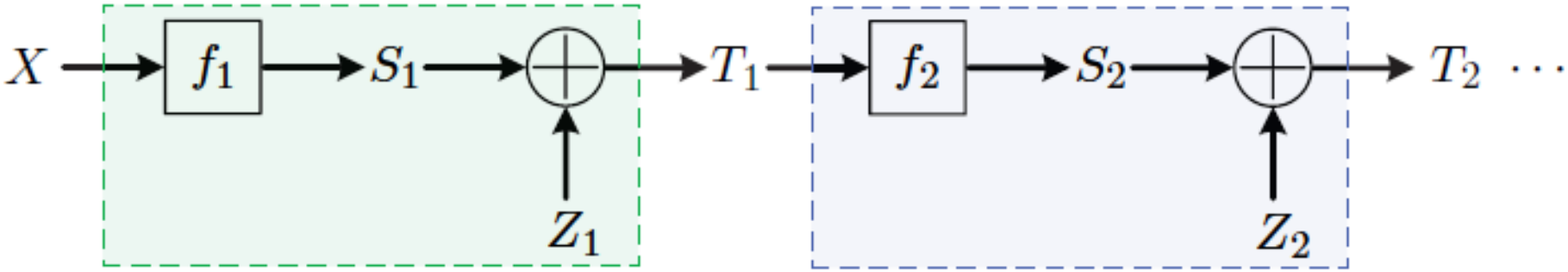
$\circledast$ P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate

$\circledast$ Both distributions can be sampled via the DNN forward pass

- Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
- Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X; T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell | X = x_i)$

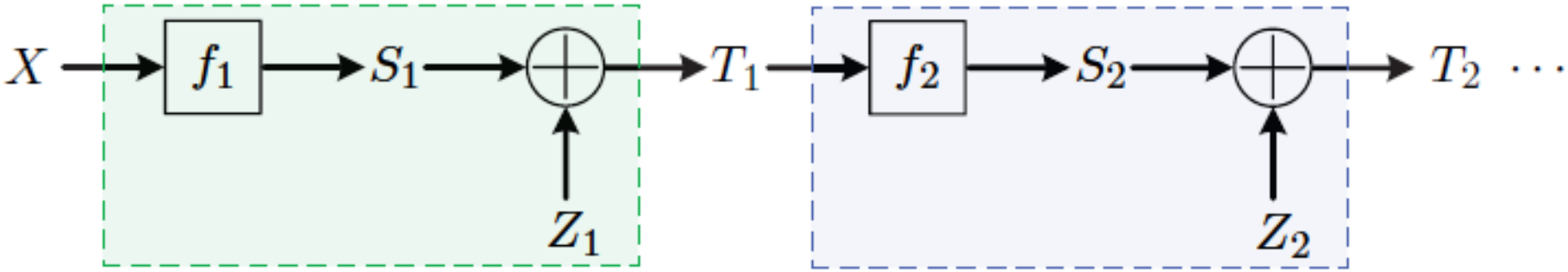
$\circledast$ P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate

$\circledast$ Both distributions can be sampled via the DNN forward pass

- Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
- Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



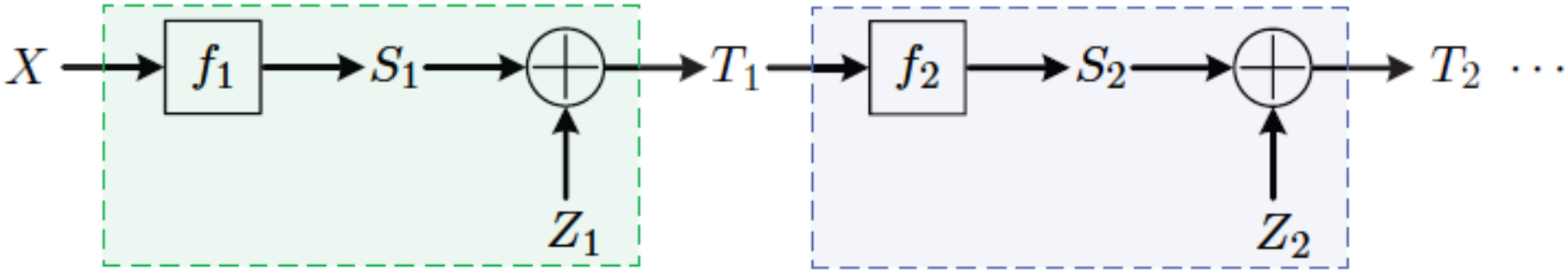
- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X; T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell | X = x_i)$

- P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate
- Both distributions can be sampled via the DNN forward pass
  - Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
  - Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Mutual Information in Noisy DNNs

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



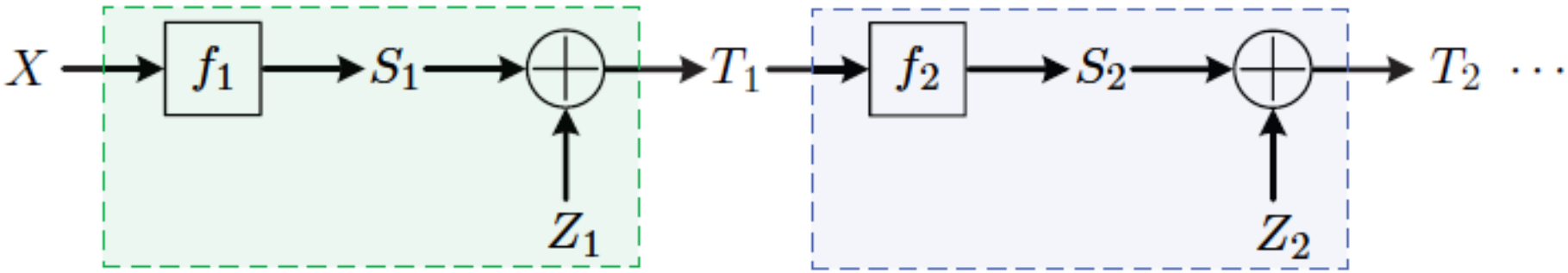
- Consider $X \sim \texttt{Unif}(\mathscr{X})$, where $\mathscr{X} = \{x_i\}_{i=1}^m$ is an empirical dataset

$\Rightarrow$ **Mutual information:** $I(X; T_\ell) = h(T_\ell) - \frac{1}{m}\sum_{i=1}^m h(T_\ell | X = x_i)$

- P_{T_ℓ} and $P_{T_\ell|X}$ are **extremely** complicated to evaluate
- Both distributions can be sampled via the DNN forward pass
  - Sampling P_{T_ℓ}: Feed **randomly chosen** x_i's and read T_ℓ values
  - Sampling $P_{T_\ell|X=x_i}$: Feed x_i **multiple times** and read T_ℓ values

Exploit Structure to Estimate Diff. Entropy

Noisy DNN: $S_\ell = f_\ell(T_{\ell-1}) \quad \Rightarrow \quad T_\ell = S_\ell + Z_\ell,\ Z_\ell \sim \mathcal{N}(0, \beta^2)$



We know that $T_\ell = S_\ell + Z_\ell \sim P_{S_\ell} * P_N$ and:

- empirical distributions $\hat{P}_{S_\ell}$ and $\hat{P}_{S_\ell|X=x_i}$ (sample $T_{\ell-1}$ and apply f_ℓ)
- noise distribution $Z_\ell \sim P_N$ (since it is designed)

$\Longrightarrow$ Estimate $h(P_{S_\ell} * P_N)$ based on n i.i.d. samples from P_{S_ℓ} and knowledge of P_N (noise distribution)

Convergence of the Estimator

Theorem (Goldfeld-Greenewald-Polyanski'19)

For $\{P : \mathrm{supp}(P) \subseteq [-1,1]^d\}$ *and any* $\beta > 0$ *and* $d \geq 1$, *we have*

$$\sup_P \mathbb{E}_{S^n} \left| h(P_{S_\ell} * P_N) - h(\hat{P}_{S_\ell} * P_N) \right| \leq O\left(\frac{(\log n)^{d/4}}{\sqrt{n}} \right) \tag{1}$$

Comments

- $h(\hat{P}_{S_\ell} * P_N)$ is the diff. entropy of a **known** Gaussian mixture
- Faster rate than known estimation methods such as kNN with $O\left(n^{-\frac{\alpha s}{\beta s + d}}\right)$
- Explicit expression enables concrete error bounds in simulations

Convergence of the Estimator

Theorem (Goldfeld-Greenewald-Polyanski'19)

For $\{P : \mathrm{supp}(P) \subseteq [-1,1]^d\}$ *and any* $\beta > 0$ *and* $d \geq 1$, *we have*

$$\sup_P \mathbb{E}_{S^n} \left| h(P_{S_\ell} * P_N) - h(\hat{P}_{S_\ell} * P_N) \right| \leq O\left(\frac{(\log n)^{d/4}}{\sqrt{n}} \right) \tag{1}$$

Comments

- $h(\hat{P}_{S_\ell} * P_N)$ is the diff. entropy of a **known** Gaussian mixture
- Faster rate than known estimation methods such as kNN with $O\left(n^{-\frac{\alpha s}{\beta s + d}}\right)$
- Explicit expression enables concrete error bounds in simulations

Convergence of the Estimator

Theorem (Goldfeld-Greenewald-Polyanski'19)

For $\{P : \mathrm{supp}(P) \subseteq [-1,1]^d\}$ *and any* $\beta > 0$ *and* $d \geq 1$, *we have*

$$\sup_P \mathbb{E}_{S^n} \left| h(P_{S_\ell} * P_N) - h(\hat{P}_{S_\ell} * P_N) \right| \leq O\left(\frac{(\log n)^{d/4}}{\sqrt{n}} \right) \tag{1}$$

Comments

- $h(\hat{P}_{S_\ell} * P_N)$ is the diff. entropy of a **known** Gaussian mixture
- Faster rate than known estimation methods such as kNN with $O\left(n^{-\frac{\alpha s}{\beta s + d}}\right)$
- Explicit expression enables concrete error bounds in simulations

Convergence of the Estimator

Theorem (Goldfeld-Greenewald-Polyanski'19)

For $\{P : \text{supp}(P) \subseteq [-1,1]^d\}$ *and any* $\beta > 0$ *and* $d \geq 1$, *we have*

$$\sup_P \mathbb{E}_{S^n} \left| h(P_{S_\ell} * P_N) - h(\hat{P}_{S_\ell} * P_N) \right| \leq O\left(\frac{(\log n)^{d/4}}{\sqrt{n}} \right) \tag{1}$$

Comments

- $h(\hat{P}_{S_\ell} * P_N)$ is the diff. entropy of a **known** Gaussian mixture
- Faster rate than known estimation methods such as kNN with $O\left(n^{-\frac{\alpha s}{\beta s + d}}\right)$
- Explicit expression enables concrete error bounds in simulations

References

- Goldfeld, Z., Van Den Berg, E., Greenewald K., Melnyk I., Nguyen N. and Kingsbury B., (2019) Estimating Information Flow in Deep Neural Networks, *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97:2299-2308, 2019.
- Goldfeld, Z., Greenewald K., Weed J. and Polyanskiy Y., (2019) Optimality of the plug-in estimator for differential entropy estimation under Gaussian convolutions, *IEEE International Symposium on Information Theory (ISIT-2019)*, July 2019.