Estimating Information Flow in Deep Neural Networks

Ziv Goldfeld

MIT

56th Allerton Conference on Communication, Control, and Computing
Monticello, Illinois, US

October 4th, 2018

Collaborators: E. van den Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury and Y. Polyanskiy
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks
- Long way to go theory-wise:
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks

- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks
- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks
- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks
- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks

- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?

- Past attempts to understand effectiveness of deep learning
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks
- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
- Past attempts to understand effectiveness of deep learning
  - Optimization in parameter space [Saxe’14, Choromanska’15, Advani’17]
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks

- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?

- Past attempts to understand effectiveness of deep learning
  - Optimization in parameter space [Saxe’14, Choromanska’15, Advani’17]
  - Classes of efficiently representable functions [Montufar’14, Poggio’17]
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks

- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?

- Past attempts to understand effectiveness of deep learning
  - Optimization in parameter space [Saxe’14, Choromanska’15, Advani’17]
  - Classes of efficiently representable functions [Montufar’14, Poggio’17]
  - Information theory [Tishby’17, Saxe’18, Gabrié’18]
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks

- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?

- Past attempts to understand effectiveness of deep learning
  - Optimization in parameter space [Saxe’14, Choromanska’15, Advani’17]
  - Classes of efficiently representable functions [Montufar’14, Poggio’17]
  - Information theory [Tishby’17, Saxe’18, Gabrié’18]
How do Deep Neural Networks Learn?

- Unprecedented practical success in hosts of tasks
- Long way to go theory-wise:
  - What drives the evolution of hidden representations?
  - What are properties of learned representations?
  - How fully trained networks process information?
- Past attempts to understand effectiveness of deep learning
  - Optimization in parameter space [Saxe’14, Choromanska’15, Advani’17]
  - Classes of efficiently representable functions [Montufar’14, Poggio’17]
  - **Information theory** [Tishby’17, Saxe’18, Gabrié’18]

★ **Goal:** Explain ‘compression’ in Information Bottleneck framework
Feedforward DNN for Classification:

\[
\begin{align*}
\mathcal{L} & = \text{Label} \\
\mathcal{K} & = \mathcal{L}(\text{Feature/Image}) \\
\mathcal{P} \mathcal{O} & = \mathcal{K}(\text{Input Layer}) \\
\mathcal{P} \mathcal{H} \mathcal{I} \mathcal{L} & = \mathcal{P} \mathcal{O} (\text{Hidden Layer 1}) \\
\mathcal{P} \mathcal{H} \mathcal{I} \mathcal{L} \mathcal{1} & = \mathcal{P} \mathcal{H} \mathcal{I} \mathcal{L} (\text{Hidden Layer 1}) \\
\mathcal{P} \mathcal{H} \mathcal{I} \mathcal{L} \mathcal{2} & = \mathcal{P} \mathcal{H} \mathcal{I} \mathcal{L} (\text{Hidden Layer 1}) \\
\mathcal{O} & = \mathcal{P} \mathcal{H} \mathcal{I} \mathcal{L} \mathcal{2} (\text{Output Layer})
\end{align*}
\]
Setup and Preliminaries

Feedforward DNN for Classification:

\[ Y \quad X \quad T_0 = X \]

(Y (Label)  X (Feature/Image)  T_0 = X (Input Layer))

Cat

Dog
Setup and Preliminaries

Feedforward DNN for Classification:

\[ Y \quad (\text{Label}) \quad X \quad (\text{Feature/Image}) \quad T_0 = X \quad (\text{Input Layer}) \quad T_1 \quad (\text{Hidden Layer 1}) \quad T_2 \quad (\text{Hidden Layer 1}) \quad T_3 \quad (\text{Hidden Layer 1}) \]

Cat

Dog

Estimating Information Flow in DNNs
Setup and Preliminaries

Feedforward DNN for Classification:

\[ \hat{Y} \] (Label) \[ X \] (Feature/Image) \[ T_0 = X \] (Input Layer) \[ T_1 \] (Hidden Layer 1) \[ T_2 \] (Hidden Layer 1) \[ T_3 \] (Hidden Layer 1) \[ T_4 = \hat{Y} \] (Output Layer)

Cat

Dog
Setup and Preliminaries

Feedforward DNN for Classification:

\( T_\ell = f_\ell(T_{\ell-1}) \) (MLP: \( T_\ell = \sigma(W_\ell T_{\ell-1} + b_\ell) \))

- Deterministic DNN:

\( Y \) (Label) \( X \) (Feature/Image) \( T_0 = X \) (Input Layer) \( T_1 \) (Hidden Layer 1) \( T_2 \) (Hidden Layer 1) \( T_3 \) (Hidden Layer 1) \( T_4 = \hat{Y} \) (Output Layer)

Cat

Dog
Setup and Preliminaries

Feedforward DNN for Classification:

- **Deterministic DNN**: $T_\ell = f_\ell(T_{\ell-1})$  
  **MLP**: $T_\ell = \sigma(W_\ell T_{\ell-1} + b_\ell)$

- **$\ell$th Hidden Layer Enc & Dec**: $P_{T_\ell|X}$ (enc) and $P_{\hat{Y}|T_\ell}$ (dec)
Feedforward DNN for Classification:

- **Deterministic DNN:** $T_\ell = f_\ell(T_{\ell-1})$ (MLP: $T_\ell = \sigma(W_\ell T_{\ell-1} + b_\ell)$)
- **$\ell$th Hidden Layer Enc & Dec:** $P_{T_\ell|X}$ (enc) and $P_{\hat{Y}|T_\ell}$ (dec)
- **IB Theory:** Track MI pairs $(I(X;T_\ell), I(Y;T_\ell))$ (information plane)
Feedforward DNN for Classification:

IB Theory Claim: Training comprises 2 phases
IB Theory Claim: Training comprises 2 phases

- **Fitting:** $I(Y; T_\ell)$ & $I(X; T_\ell)$ rise (short)
Setup and Preliminaries

Feedforward DNN for Classification:

IB Theory Claim: Training comprises 2 phases

- **Fitting:** $I(Y; T_\ell) \& I(X; T_\ell)$ rise (short)
- **Compression:** $I(X; T_\ell)$ slowly drops (long)
Feedforward DNN for Classification:

IB Theory Claim: Training comprises 2 phases

- **Fitting**: $I(Y; T_\ell) \& I(X; T_\ell)$ rise (short)
- **Compression**: $I(X; T_\ell)$ slowly drops (long)
Setup and Preliminaries

Feedforward DNN for Classification:

\[
T_0 = X \quad (Input \ Layer) \quad T_1 \quad (Hidden \ Layer \ 1) \quad T_2 \quad (Hidden \ Layer \ 1) \quad T_3 
\]

\[
T_4 = \hat{Y} \quad (Output \ Layer)
\]

IB Theory Claim: Training comprises 2 phases

- **Fitting:** \( I(Y; T_\ell) \ & I(X; T_\ell) \) rise (short)
- **Compression:** \( I(X; T_\ell) \) slowly drops (long)
Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)
**Observation**

*Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)*

\[ I(X; T_\ell) \text{ is independent of the DNN parameters} \]
Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)
\[ \implies I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why?
Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid) \( \implies I(X; T_\ell) \) is independent of the DNN parameters

**Why?** Formally…
Observation

\[ \text{Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)} \Rightarrow I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why? Formally...

- Continuous \( X \):
Meaningless Mutual Information

Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ \implies I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why? Formally...

- **Continuous** \( X \):
  \[
  I(X; T_\ell) = h(T_\ell) - h(T_\ell|X)
  \]
Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ I(X; T_\ell) \text{ is independent of the DNN parameters} \]

**Why?** Formally...

- **Continuous** \( X \):
  \[ I(X; T_\ell) = h(T_\ell) - h(T_\ell | X) \]
Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid) \[\implies I(X; T_\ell) \text{ is independent of the DNN parameters}\]

**Why?** Formally...

- **Continuous** \(X\): \[I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X) | X)\]
Meaningless Mutual Information

Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

$$I(X; T_\ell) \text{ is independent of the DNN parameters}$$

Why? Formally...

- Continuous $X$:

$$I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X) | X)$$

$$= -\infty$$
**Observation**

*Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)*

\[ I(X; T_\ell) \text{ is independent of the DNN parameters} \]

**Why?** Formally...

- **Continuous** \( X \):
  \[
  I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X)|X) = \infty
  \]
**Observation**

*Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)*

\[ \implies I(X; T_\ell) \text{ is independent of the DNN parameters} \]

**Why?** Formally...

- **Continuous** \( X \):
  \[ I(X; T_\ell) = h(T_\ell) - h(f_\ell(X)|X) = \infty \]

- **Discrete** \( X \):
Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

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

Why? Formally...

- **Continuous $X$:**
  \[
  I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X)|X) = \infty
  \]

- **Discrete $X$:** The map $X \mapsto T_\ell$ is injective*

* For almost all weight matrices and bias vectors
Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ \implies I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why? Formally...

- **Continuous** \( X \):
  \[ I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X) | X) = \infty \]

- **Discrete** \( X \): The map \( X \mapsto T_\ell \) is injective* \[ \implies I(X; T_\ell) = H(X) \]

* For almost all weight matrices and bias vectors
Meaningless Mutual Information

Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ \Rightarrow I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why?

Formally...

- **Continuous** \( X \):
  \[ I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X)\mid X) = \infty \]

- **Discrete** \( X \):
  The map \( X \mapsto T_\ell \) is injective* \( \Rightarrow I(X; T_\ell) = H(X) \)
Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ \implies I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why? Formally...

- **Continuous** \( X \): 
  \[ I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X) \mid X) = \infty \]

- **Discrete** \( X \): The map \( X \mapsto T_\ell \) is injective* \( \implies I(X; T_\ell) = H(X) \)

Intuition:
Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ I(X; T_\ell) \text{ is independent of the DNN parameters} \]

Why?

Formally...

- **Continuous** $X$:
  \[ I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X)|X) = \infty \]

- **Discrete** $X$:
  The map $X \mapsto T_\ell$ is injective

**Intuition:** Encoding all info. about $X$ is arbitrarily fine variations of $T_\ell$
### Observation

Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)

\[ I(X; T_\ell) \text{ is independent of the DNN parameters} \]

#### Why?

Formally...

- **Continuous** \( X \):
  \[ I(X; T_\ell) = h(T_\ell) - h(\tilde{f}_\ell(X)|X) = \infty \]

- **Discrete** \( X \): The map \( X \mapsto T_\ell \) is injective\(^*\) \( \implies I(X; T_\ell) = H(X) \)

#### Intuition:

Encoding all info. about \( X \) is arbitrarily fine variations of \( T_\ell \)

#### Past Works:

[Schwartz-Ziv&Tishby'17, Saxe et al. '18]
What is going on here?

- Plots via binning-based estimator of \( I(X; T_\ell) \), for \( X \sim \text{Unif(dataset)} \)
What is going on here?

- Plots via binning-based estimator of $I(X; T_\ell)$, for $X \sim \text{Unif}(\text{dataset})$
  \[\implies \text{Plotted values are } I(X; \text{Bin}(T_\ell))\]
What is going on here?

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

  $\implies$ Plotted values are $I(X; \text{Bin}(T_\ell)) \approx I(X; T_\ell)$
Plots via binning-based estimator of $I(X; T_\ell)$, for $X \sim \text{Unif}(\text{dataset})$

$\implies$ Plotted values are $I(X; \text{Bin}(T_\ell)) \approx I(X; T_\ell)$

No!
What is going on here?

- Plots via binning-based estimator of $I(X;T_\ell)$, for $X \sim \text{Unif}\,(\text{dataset})$
  
  $\implies$ Plotted values are $I(X;\text{Bin}(T_\ell)) \approx I(X;T_\ell)$ $\textbf{No!}$
What is going on here?

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

  $\implies$ Plotted values are $I(X; \text{Bin}(T_\ell))$ $\approx I(X; T_\ell)$ No!

- Smaller bins $\implies$ Closer to truth: $I(X; T_\ell) = \ln(2^{12}) \approx 8.31$
What is going on here?

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

  $\implies$ Plotted values are $I(X; \text{Bin}(T_\ell)) \approx I(X; T_\ell)$ No!

- Smaller bins $\implies$ Closer to truth: $I(X; T_\ell) = \ln(2^{12}) \approx 8.31$

- Binning introduces “noise” into estimator (not present in the DNN)
What is going on here?

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

  $\implies$ Plotted values are $I(X; \text{Bin}(T_\ell)) \approx I(X; T_\ell)$ No!

- Smaller bins $\implies$ Closer to truth: $I(X; T_\ell) = \ln(2^{12}) \approx 8.31$
- Binning introduces “noise” into estimator (not present in the DNN)
- Plots showing estimation errors
What is going on here?

- Plots via binning-based estimator of $I(X; T_\ell)$, for $X \sim \text{Unif}($dataset$)$
  
  $\implies$ Plotted values are $I(X; \text{Bin}(T_\ell)) \approx I(X; T_\ell)$ No!

- Smaller bins $\implies$ Closer to truth: $I(X; T_\ell) = \ln(2^{12}) \approx 8.31$

- Binning introduces “noise” into estimator (not present in the DNN)

- Plots showing estimation errors

- **Real Problem:** $I(X; T_\ell)$ is meaningless for studying the DNN
Proposed Fix: Inject (small) Gaussian noise to neurons’ output
Noisy Deep Neural Networks

**Proposed Fix:** Inject (small) Gaussian noise to neurons’ output

- **Formally:** \( T_\ell = f_\ell(T_{\ell-1}) + Z_\ell \), where \( Z_\ell \sim \mathcal{N}(0, \beta^2 I) \)
**Proposed Fix:** Inject (small) Gaussian noise to neurons’ output

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

```
\[
T_{\ell-1} \xrightarrow{\sigma(W_\ell^{(k)}T_{\ell-1} + b_\ell^{(k)})} S_\ell^{(k)} \xrightarrow{\text{+}} T_\ell^{(k)} \\
Z_\ell^{(k)} \sim \mathcal{N}(0, \beta^2)
\]
```
Proposed Fix: Inject (small) Gaussian noise to neurons’ output

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

\[
\begin{align*}
T_{\ell-1} & \xrightarrow{\sigma(W^{(k)}_{\ell}T_{\ell-1} + b^{(k)}_{\ell})} S_{\ell}(k) \xrightarrow{+} T_{\ell}(k) \\
Z_{\ell}(k) & \sim \mathcal{N}(0, \beta^2) \\
\end{align*}
\]

\( X \mapsto T_\ell \) is a \textbf{parametrized channel} that depends on DNN param.!
**Proposed Fix:** Inject (small) Gaussian noise to neurons’ output

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

\[
T_{\ell-1} \xrightarrow{\sigma(W^{(k)}_\ell T_{\ell-1} + b_\ell(k))} S_\ell(k) \xrightarrow{+} T_\ell(k)
\]

\( Z_\ell(k) \sim \mathcal{N}(0, \beta^2) \)

\( \Rightarrow X \mapsto T_\ell \) is a **parametrized channel** that depends on DNN param.

- **Operational Perspective:**
Proposed Fix: Inject (small) Gaussian noise to neurons’ output

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

\[
\begin{align*}
T_{\ell-1} & \xrightarrow{\sigma(W^{(k)}_{\ell}T_{\ell-1} + b^{(k)}_{\ell})} S^{(k)}_\ell \xrightarrow{+} T^{(k)}_\ell \\
Z^{(k)}_\ell & \sim \mathcal{N}(0, \beta^2)
\end{align*}
\]

\( \implies X \leftrightarrow T_\ell \) is a parametrized channel that depends on DNN param.!

- Operational Perspective:
  - Performance & learned representations similar to det. DNNs \((\beta \approx 10^{-1})\)
**Proposed Fix:** Inject (small) Gaussian noise to neurons’ output

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

\[ T_{\ell-1} \xrightarrow{\sigma(W_\ell^{(k)}T_{\ell-1} + b_\ell(k))} S_\ell(k) \xrightarrow{+} T_\ell(k) \]

\[ Z_\ell(k) \sim \mathcal{N}(0, \beta^2) \]

\[ \Rightarrow X \mapsto T_\ell \text{ is a parametrized channel that depends on DNN param.}! \]

- **Operational Perspective:**
  - Performance & learned representations similar to det. DNNs (\( \beta \approx 10^{-1} \))
  - Noise masks fine variations – MI represents relevant/distinguishable info.
**Proposed Fix:** Inject (small) Gaussian noise to neurons’ output

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

![Diagram of deep neural network](image)

\( T_{\ell-1} \xrightarrow{\sigma(W_\ell^{(k)}T_{\ell-1} + b_\ell(k))} S_\ell(k) \xrightarrow{+} T_\ell(k) \)

\( Z_\ell(k) \sim \mathcal{N}(0, \beta^2) \)

\( \implies X \mapsto T_\ell \) is a **parametrized channel** that depends on DNN param.!

- **Operational Perspective:**
  - Performance & learned representations similar to det. DNNs (\( \beta \approx 10^{-1} \))
  - Noise masks fine variations – MI represents relevant/distinguishable info.
  - Dropout & quantized DNNs widely used in practice \( \approx \) internal noise
Layer $\ell$: Denote $S_\ell \triangleq f_\ell(T_{\ell-1})$
Layer $\ell$: Denote $S_{\ell} \triangleq f_{\ell}(T_{\ell-1}) \implies T_{\ell} = S_{\ell} + Z_{\ell}, \ Z_{\ell} \sim \mathcal{N}(0, \beta^2 I)$
Mutual Information (Estimation) in Noisy DNNs

- **Layer** $\ell$: Denote $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \ Z_\ell \sim \mathcal{N}(0, \beta^2 I)$

- **Assume:** $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^m$ is empirical dataset
Layer $\ell$: Denote $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell$, $Z_\ell \sim \mathcal{N}(0, \beta^2 I)$

Assume: $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^m$ is empirical dataset

Mutual Information: $I(X; T_\ell) = h(T_\ell) - \frac{1}{m} \sum_{i=1}^m h(T_\ell | X = x_i)$
Layer $\ell$: Denote $S_{\ell} \triangleq f_{\ell}(T_{\ell-1}) \implies T_{\ell} = S_{\ell} + Z_{\ell}$, $Z_{\ell} \sim \mathcal{N}(0, \beta^2 I)$

Assume: $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^{m}$ is empirical dataset

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

Distribution of $S_{\ell}$ is extremely complicated to compute/evaluate
Mutual Information (Estimation) in Noisy DNNs

- **Layer $\ell$:** Denote $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell$, $Z_\ell \sim \mathcal{N}(0, \beta^2 I)$

- **Assume:** $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^m$ is empirical dataset

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

- Distribution of $S_\ell$ is extremely complicated to compute/evaluate

- But, $P_{S_\ell}$ and $P_{S_\ell|X=x_i}$ are easily sampled from via DNN fwd. pass
Mutual Information (Estimation) in Noisy DNNs

- **Layer ℓ**: Denote \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \ Z_\ell \sim \mathcal{N}(0, \beta^2 I) \)

- **Assume**: \( X \sim \text{Unif}(\mathcal{X}) \), where \( \mathcal{X} \triangleq \{ x_i \}_{i=1}^m \) is empirical dataset

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

- Distribution of \( S_\ell \) is **extremely** complicated to compute/evaluate

- But, \( P_{S_\ell} \) and \( P_{S_\ell | X=x_i} \) are **easily** sampled from via DNN fwd. pass

\[ \implies \text{Estimate MI from samples & Exploit noisy DNN structure} \]
Mutual Information (Estimation) in Noisy DNNs

- **Layer ℓ**: Denote $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \ Z_\ell \sim \mathcal{N}(0, \beta^2 I)$

- **Assume**: $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^m$ is empirical dataset

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

- Distribution of $S_\ell$ is extremely complicated to compute/evaluate

- But, $P_{S_\ell}$ and $P_{S_\ell | X = x_i}$ are easily sampled from via DNN f.w.d. pass

  $\implies$ Estimate MI from samples & Exploit noisy DNN structure

**Differential Entropy Estimation under Gaussian Convolutions**

Estimate $h(S + Z)$ using $n$ i.i.d. samples from $P_S \in \mathcal{F}_d$ (nonparametric class) and knowing that $Z \sim \mathcal{N}(0, \beta^2 I_d)$ independent of $S$. 

**Layer** $\ell$: Denote $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell$, $Z_\ell \sim \mathcal{N}(0, \beta^2 I)$

**Assume:** $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^m$ is empirical dataset

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

- Distribution of $S_\ell$ is extremely complicated to compute/evaluate
- But, $P_{S_\ell}$ and $P_{S_\ell | X = x_i}$ are easily sampled from via DNN fwd. pass

$\implies$ Estimate MI from samples & Exploit noisy DNN structure

---

**Differential Entropy Estimation under Gaussian Convolutions**

Estimate $h(S + Z)$ using $n$ i.i.d. samples from $P_S \in \mathcal{F}_d$ (nonparametric class) and knowing that $Z \sim \mathcal{N}(0, \beta^2 I_d)$ independent of $S$.

**Results** [ZG-Greenewald-Polyanskiy’18]:
Mutual Information (Estimation) in Noisy DNNs

- **Layer** $\ell$: Denote $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell$, $Z_\ell \sim \mathcal{N}(0, \beta^2 I)$

- **Assume**: $X \sim \text{Unif}(\mathcal{X})$, where $\mathcal{X} \triangleq \{x_i\}_{i=1}^m$ is empirical dataset

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

- Distribution of $S_\ell$ is **extremely** complicated to compute/evaluate

- But, $P_{S_\ell}$ and $P_{S_\ell | X = x_i}$ are **easily** sampled from via DNN fwd. pass

  $\implies$ Estimate MI from samples & Exploit noisy DNN structure

---

**Differential Entropy Estimation under Gaussian Convolutions**

Estimate $h(S + Z)$ using $n$ i.i.d. samples from $P_S \in \mathcal{F}_d$ (nonparametric class) and knowing that $Z \sim \mathcal{N}(0, \beta^2 I_d)$ independent of $S$.

**Results** [ZG-Greenewald-Polyanskiy’18]:

- Sample complexity is exponential in $d$
Mutual Information (Estimation) in Noisy DNNs

- **Layer \( \ell \):** Denote \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \ Z_\ell \sim \mathcal{N}(0, \beta^2 I) \)

- **Assume:** \( X \sim \text{Unif}(\mathcal{X}), \) where \( \mathcal{X} \triangleq \{x_i\}_{i=1}^m \) is empirical dataset

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

- Distribution of \( S_\ell \) is extremely complicated to compute/evaluate

- But, \( P_{S_\ell} \) and \( P_{S_\ell | X = x_i} \) are easily sampled from via DNN fwd. pass

  \[ \implies \text{Estimate MI from samples & Exploit noisy DNN structure} \]

**Differential Entropy Estimation under Gaussian Convolutions**

Estimate \( h(S + Z) \) using \( n \) i.i.d. samples from \( P_S \in \mathcal{F}_d \) (nonparametric class) and knowing that \( Z \sim \mathcal{N}(0, \beta^2 I_d) \) independent of \( S \).

**Results** [ZG-Greenewald-Polyanskiy’18]:

- Sample complexity is exponential in \( d \)

- Absolute-error minimax risk is \( O((\log n)^{d/4} / \sqrt{n}) \) (all const. explicit)
Single Neuron Classification:

$I(X; T_{\ell})$ Dynamics - Illustrative Minimal Example
Single Neuron Classification:

$$X \xrightarrow{\text{tanh}(wX + b)} S_{w,b} \xrightarrow{\text{}} T$$

$$Z \sim \mathcal{N}(0, \beta^2)$$
Single Neuron Classification:

- **Input**: \( X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1) \)
  
  \( \mathcal{X}_{-1} \triangleq \{-3, -1, 1\} \), \( \mathcal{X}_1 \triangleq \{3\} \)
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1)$
  - $\mathcal{X}_{-1} \triangleq \{-3, -1, 1\}$, $\mathcal{X}_1 \triangleq \{3\}$

\[X \xrightarrow{\tanh(wX + b)} S_{w,b} \xrightarrow{T} Z \sim \mathcal{N}(0, \beta^2)\]
**Single Neuron Classification:**

- **Input:** $X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1)$

  $\mathcal{X}_{-1} \triangleq \{-3, -1, 1\}$, $\mathcal{X}_1 \triangleq \{3\}$

- $Z \sim \mathcal{N}(0, \beta^2)$

- Move tanh center $x = 2$ (⟺ $b = -2$)
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1) \)
  \[
  \mathcal{X}_{-1} \triangleq \{-3, -1, 1\}, \quad \mathcal{X}_1 \triangleq \{3\}
  \]
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}(X_{-1} \cup X_1)$
  
  $X_{-1} \triangleq \{-3, -1, 1\}$, $X_1 \triangleq \{3\}$

\[ X \xrightarrow{\tanh(wX + b)} S_{w,b} \xrightarrow{\pm} T \]

\[ Z \sim \mathcal{N}(0, \beta^2) \]

\(\star\) **Sharpen** \(\tanh\) transition (\(\iff\) increase \(w\) and keep \(b = -2w\))
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}(X_- \cup X_1)$
  
  $X_- \triangleq \{-3, -1, 1\}$, $X_1 \triangleq \{3\}$
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1) \)
  \( \mathcal{X}_{-1} \triangleq \{-3, -1, 1\} \), \( \mathcal{X}_1 \triangleq \{3\} \)

\[
S_{5,-10}
\]

\[
\begin{align*}
X &\xrightarrow{\text{tanh}(wX + b)} S_{w,b} \\
&\xrightarrow{+} T \\
Z &\sim \mathcal{N}(0, \beta^2)
\end{align*}
\]

✓ Correct classification performance
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}(\mathcal{X}_- \cup \mathcal{X}_1)$
  
  $\mathcal{X}_- \triangleq \{-3, -1, 1\}$, $\mathcal{X}_1 \triangleq \{3\}$

- **Empirical Results:**

\[
Z \sim \mathcal{N}(0, \beta^2)
\]
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1)$
  \[ \mathcal{X}_{-1} \triangleq \{-3, -1, 1\}, \quad \mathcal{X}_1 \triangleq \{3\} \]

- **Mutual Information:**
  \[ I(X; T) \]
Input: \(X \sim \text{Unif}(X_{-1} \cup X_1)\)
\[X_{-1} \triangleq \{-3, -1, 1\}, \quad X_1 \triangleq \{3\}\]

Mutual Information:
\[I(X; T) = I(X; S_{w,b} + Z)\]
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1) \)
  \( \mathcal{X}_{-1} \triangleq \{-3, -1, 1\} \), \( \mathcal{X}_1 \triangleq \{3\} \)

- **Mutual Information:**
  \[
  I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z)
  \]
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1)$
  \[ \mathcal{X}_{-1} \triangleq \{-3, -1, 1\}, \mathcal{X}_1 \triangleq \{3\} \]

- **Mutual Information:**
  \[
  I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z) = I(S_{w,b}; S_{w,b} + Z)
  \]

\[ Z \sim \mathcal{N}(0, \beta^2) \]
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(X_{-1} \cup X_1) \)
  \[ X_{-1} \triangleq \{-3, -1, 1\}, \; X_1 \triangleq \{3\} \]

- **Mutual Information:**
  \[ I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z) = I(S_{w,b}; S_{w,b} + Z) \]
  \[ \Rightarrow \quad I(X; T) \text{ is the aggregate info. transmitted over AWGN w. symbols} \]
  \[ S_{w,b} \triangleq \{\tanh(-3w+b), \tanh(-w+b), \tanh(w+b), \tanh(3w+b)\} \]
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1) \)
  \( \mathcal{X}_{-1} \triangleq \{-3, -1, 1\} \), \( \mathcal{X}_1 \triangleq \{3\} \)

- **Mutual Information:**

\[
I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z) = I(S_{w,b}; S_{w,b} + Z)
\]

\( \Rightarrow \) \( I(X; T) \) is the aggregate info. transmitted over AWGN w. symbols

\( S_{w,b} \triangleq \{\tanh(-3w + b), \tanh(-w + b), \tanh(w + b), \tanh(3w + b)\} \rightarrow \{\pm 1\} \)
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(\mathcal{X}_- \cup \mathcal{X}_1) \)
  \[ \mathcal{X}_- \triangleq \{-3, -1, 1\}, \mathcal{X}_1 \triangleq \{3\} \]

- **Mutual Information:**

  \[
  I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z) = I(S_{w,b}; S_{w,b} + Z)
  \]

  \(\Rightarrow\) \( I(X; T) \) is the aggregate info. transmitted over AWGN w. symbols

- **Symbols:**
  \[
  S_{w,b} \triangleq \{\tanh(-3w+b), \tanh(-w+b), \tanh(w+b), \tanh(3w+b)\} \rightarrow \{\pm 1\}
  \]
Single Neuron Classification:

- **Input:** \( X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1) \)
  \( \mathcal{X}_{-1} \triangleq \{-3, -1, 1\} \), \( \mathcal{X}_1 \triangleq \{3\} \)

- **Mutual Information:**
  \[
  I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z) = I(S_{w,b}; S_{w,b} + Z)
  \]
  \[
  \Rightarrow I(X; T) \text{ is the aggregate info. transmitted over AWGN w. symbols}
  \]
  \[
  S_{w,b} \triangleq \{\tanh(-3w+b), \tanh(-w+b), \tanh(w+b), \tanh(3w+b)\} \rightarrow \{\pm 1\}
  \]
$I(X; T_{\ell})$ Dynamics - Illustrative Minimal Example

**Single Neuron Classification:**

- **Input:** $X \sim \text{Unif}(\mathcal{X}_{-1} \cup \mathcal{X}_1)$
  \[ \mathcal{X}_{-1} \triangleq \{-3, -1, 1\}, \mathcal{X}_1 \triangleq \{3\} \]

- **Mutual Information:**
  \[
  I(X; T) = I(X; S_{w,b} + Z) = I(\tanh(wX + b); S_{w,b} + Z) = I(S_{w,b}; S_{w,b} + Z)
  \]
  \[
  \Rightarrow \quad I(X; T) \text{ is the aggregate info. transmitted over AWGN w. symbols}
  \]
  \[
  S_{w,b} \triangleq \{\tanh(-3w + b), \tanh(-w + b), \tanh(w + b), \tanh(3w + b)\} \rightarrow \{\pm 1\}
  \]
Clustering of Representations - Larger Networks

Noisy version of DNN from [Schwartz-Ziv&Tishby’17]:
Clustering of Representations - Larger Networks

Noisy version of DNN from [Schwartz-Ziv&Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
Noisy version of DNN from [Schwartz-Ziv&Tishby’17]:

- **Binary Classification:** 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.:** Set to $\beta = 0.1$
Clustering of Representations - Larger Networks

Noisy version of DNN from [Schwartz-Ziv&Tishby'17]:

- **Binary Classification:** 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.:** Set to $\beta = 0.1$
Clustering of Representations - Larger Networks

Noisy version of DNN from [Schwartz-Ziv&Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\beta = 0.1$
Clustering of Representations - Larger Networks

Noisy version of DNN from [Schwartz-Ziv&Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\beta = 0.1$
Clustering of Representations - Larger Networks

Noisy version of DNN from [Schwartz-Ziv&Tishby’17]:

- **Binary Classification:** 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.:** Set to $\beta = 0.1$

Compression of $I(X; T_\ell)$ driven by clustering of representations
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
Circling back to Deterministic DNNs

- \( I(X; T_\ell) \) is constant \( \implies \) Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy $H(\text{Bin}(T_\ell))$
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\iff$ Doesn’t measure clustering

- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy $H(\text{Bin}(T_\ell))$

- **Noisy DNNs:** $I(X; T_\ell)$ and $H(\text{Bin}(T_\ell))$ highly correlated!*
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\iff$ Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy $H(\text{Bin}(T_\ell))$
- **Noisy DNNs:** $I(X; T_\ell)$ and $H(\text{Bin}(T_\ell))$ highly correlated!*  
- **Det. DNNs:** $H(\text{Bin}(T_\ell))$ compresses (resolution wrt bins size)
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering

- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy $H(\text{Bin}(T_\ell))$

- **Noisy DNNs:** $I(X; T_\ell)$ and $H(\text{Bin}(T_\ell))$ highly correlated!

- **Det. DNNs:** $H(\text{Bin}(T_\ell))$ compresses (resolution wrt bins size)

- **Past Works:** Estimated $I(X; T_\ell)$ by $I(X; \text{Bin}(T_\ell))$
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering

- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy $H(\text{Bin}(T_\ell))$

- **Noisy DNNs:** $I(X; T_\ell)$ and $H(\text{Bin}(T_\ell))$ highly correlated!*

- **Det. DNNs:** $H(\text{Bin}(T_\ell))$ compresses (resolution wrt bins size)

- **Past Works:** Estimated $I(X; T_\ell)$ by $I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell))$
Circling back to Deterministic DNNs

- $I(X; T_\ell)$ is constant $\implies$ Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy $H(\text{Bin}(T_\ell))$
- **Noisy DNNs:** $I(X; T_\ell)$ and $H(\text{Bin}(T_\ell))$ highly correlated!
- **Det. DNNs:** $H(\text{Bin}(T_\ell))$ compresses (resolution wrt bins size)
- **Past Works:** Estimated $I(X; T_\ell)$ by $I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell))$
  - Incapable of accurately estimating MI values
Circling back to Deterministic DNNs

- \( I(X; T_\ell) \) is constant \( \implies \) Doesn’t measure clustering
- Alternative measures for clustering (det. and noisy DNNs):
  - Scatter plots (up to 3D layers)
  - Within-class & In-between-class pairwise distance distribution
  - Binned entropy \( H(\text{Bin}(T_\ell)) \)

- **Noisy DNNs:** \( I(X; T_\ell) \) and \( H(\text{Bin}(T_\ell)) \) highly correlated!*  

- **Det. DNNs:** \( H(\text{Bin}(T_\ell)) \) compresses (resolution wrt bins size)

- **Past Works:** Estimated \( I(X; T_\ell) \) by \( I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell)) \)
  - \( \times \) Incapable of accurately estimating MI values
  - \( \checkmark \) Still, simple to compute & follows MI in tracking clustering!
Circling back to Deterministic DNNs (Cntd.)

Comparing to Previously Shown MI Plots:
Circling back to Deterministic DNNs (Cntd.)

Comparing to Previously Shown MI Plots:
Circling back to Deterministic DNNs (Cntd.)

Comparing to Previously Shown MI Plots:

⇒ Past works we not showing MI but clustering (via binned-MI)!
Summary

- Reexamined Information Bottleneck Compression:
Reexamined Information Bottleneck Compression:

- $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
Reexamined Information Bottleneck Compression:

- $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
- Yes, past works presented $I(X;T)$ dynamics during training
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X;T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X;T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
Summary

• Reexamined Information Bottleneck Compression:
  • $I(X; T)$ fluctuations in det. DNNs are theoretically impossible
  • Yes, past works presented $I(X; T)$ dynamics during training

• Noisy DNN Framework: Studying IT quantities over DNNs
  • Toolkit for accurate MI estimation over this framework
  • Clustering of the learned representations is the source of compression
Reexamined Information Bottleneck Compression:

- $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
- Yes, past works presented $I(X;T)$ dynamics during training

Noisy DNN Framework: Studying IT quantities over DNNs

- Toolkit for accurate MI estimation over this framework
- Clustering of the learned representations is the source of compression
- Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)
**Summary**

- **Reexamined Information Bottleneck Compression:**
  - $I(X; T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X; T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
  - Clustering of the learned representations is the source of compression
  - Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)

- **Det. DNNs cluster representations**
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X;T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
  - Clustering of the learned representations is the source of compression
  - Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)

- ![Det. DNNs cluster representations](image-url) ➞ Clarify past observations
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X; T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X; T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
  - Clustering of the learned representations is the source of compression
  - Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)

- **Det. DNNs cluster representations** $\implies$ Clarify past observations

- **Future Research:**
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X; T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X; T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
  - Clustering of the learned representations is the source of compression
  - Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)

- **Det. DNNs cluster representations** $\implies$ Clarify past observations

- **Future Research:**
  - Curse of dimensionality: How to track clustering in high-dimensions?
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X;T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
  - Clustering of the learned representations is the source of compression
  - Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)

- **Det. DNNs cluster representations** $\Rightarrow$ Clarify past observations

- **Future Research:**
  - Curse of dimensionality: How to track clustering in high-dimensions?
  - Is compression necessary? Desirable?
Summary

- **Reexamined Information Bottleneck Compression:**
  - $I(X;T)$ fluctuations in det. DNNs are theoretically impossible
  - Yes, past works presented $I(X;T)$ dynamics during training

- **Noisy DNN Framework:** Studying IT quantities over DNNs
  - Toolkit for accurate MI estimation over this framework
  - Clustering of the learned representations is the source of compression
  - Methods to track clustering in det. DNNs (incl. $H(\text{Bin}(T_\ell))$)

- **Det. DNNs cluster representations** $\implies$ Clarify past observations

- **Future Research:**
  - Curse of dimensionality: How to track clustering in high-dimensions?
  - Is compression necessary? Desirable?
  - Build on findings to improve DNN training alg. and architectures