Estimating the Flow of Information in Deep Neural Networks

Ziv Goldfeld

MIT

March 2019
Deep Learning - What's Under the Hood?

- Unprecedented practical success in hosts of tasks
Deep Learning - What’s Under the Hood?

- Unprecedented practical success in hosts of tasks
Deep Learning - What’s Under the Hood?

- Unprecedented practical success in hosts of tasks
- **Lacking Theory**: Macroscopic understanding of Deep Learning
Unprecedented practical success in hosts of tasks

Lacking Theory: Macroscopic understanding of Deep Learning
Deep Learning - What’s Under the Hood?

- Unprecedented practical success in hosts of tasks

- **Lacking Theory**: Macroscopic understanding of Deep Learning

---

![Diagram of a neural network with labeled inputs and outputs. The network includes layers labeled as Input Layer, Hidden Layer 1, Hidden Layer 2, Hidden Layer 3, and Output Layer. The labels Cat and Dog are shown at the input layer, and the output layers show percentage values C% and D%. The diagram also highlights an Internal Representation.]
Unprecedented practical success in hosts of tasks

**Lacking Theory:** Macroscopic understanding of Deep Learning

What drives the evolution of internal representations?
Deep Learning - What’s Under the Hood?

- Unprecedented practical success in hosts of tasks
- **Lacking Theory:** Macroscopic understanding of Deep Learning

What drives the evolution of internal representations?

What are properties of learned representations?
Unprecedented practical success in hosts of tasks

**Lacking Theory:** Macroscopic understanding of Deep Learning

What drives the evolution of internal representations?
What are properties of learned representations?
How fully trained networks process information?
Deep Learning - An Information-Theoretic Lens

Past Attempts to Understand Effectiveness of DL:
Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]
Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli'14, Chen-Wang-Zhao-Papailiopoulos-Koutris'18, Li-Liang’18]

- Classes of efficiently representable functions
  [Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]
Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]

- Classes of efficiently representable functions
  [Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]

- Information Bottleneck Theory
  [Tishby-Zaslavsky’15, Shwartz-Tishby’17, Saxe et al.’18, Gabrié et al.’18]
Deep Learning - An Information-Theoretic Lens

Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]

- Classes of efficiently representable functions
  [Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]

- Information Bottleneck Theory
  [Tishby-Zaslavsky’15, Shwartz-Tishby’17, Saxe et al.’18, Gabrié et al.’18]

  Study DL dynamics via an information-theoretic perspective
Optimization dynamics in parameter space
[Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]

Classes of efficiently representable functions
[Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]

Information Bottleneck Theory
[Tishby-Zaslavsky’15, Shwartz-Tishby’17, Saxe et al.’18, Gabrié et al.’18]

- Study DL dynamics via an information-theoretic perspective
- Intriguing empirical results
Deep Learning - An Information-Theoretic Lens

Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]

- Classes of efficiently representable functions
  [Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]

- Information Bottleneck Theory
  [Tishby-Zaslavsky’15, Shwartz-Tishby’17, Saxe et al.’18, Gabrié et al.’18]
  - Study DL dynamics via an information-theoretic perspective
  - Intriguing empirical results
  - Theory mostly relies on heuristic arguments
Deep Learning - An Information-Theoretic Lens

Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]

- Classes of efficiently representable functions
  [Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]

- Information Bottleneck Theory
  [Tishby-Zaslavsky’15, Shwartz-Tishby’17, Saxe et al.’18, Gabrié et al.’18]
  - Study DL dynamics via an information-theoretic perspective
  - Intriguing empirical results
  - Theory mostly relies on heuristic arguments

⭐ **Goal:** IB theory mathematical analysis
Past Attempts to Understand Effectiveness of DL:

- Optimization dynamics in parameter space
  [Saxe-McClelland-Ganguli’14, Chen-Wang-Zhao-Papailiopoulos-Koutris’18, Li-Liang’18]

- Classes of efficiently representable functions
  [Hajnal-et al’93, Delalleau-Bengio’11, Eldan-Shamir’15, Telgarsky’16, Poggio-et al’17]

- Information Bottleneck Theory
  [Tishby-Zaslavsky’15, Shwartz-Tishby’17, Saxe et al.’18, Gabrié et al.’18]
  - Study DL dynamics via an information-theoretic perspective
  - Intriguing empirical results
  - Theory mostly relies on heuristic arguments

★ Goal: IB theory mathematical analysis ⇒ better DNN designs
Setup and Preliminaries

(Deterministic) Feedforward DNN: Each layer $T_\ell = f_\ell(T_{\ell-1})$
Setup and Preliminaries

(Deterministic) Feedforward DNN: Each layer $T_\ell = f_\ell(T_{\ell-1})$

Joint Distribution: $P_{X,Y}$
Setup and Preliminaries

**(Deterministic) Feedforward DNN:** Each layer $T_\ell = f_\ell(T_{\ell-1})$

- **Joint Distribution:** $P_{X,Y} \implies P_{X,Y} \cdot P_{T_1,\ldots,T_L|X}$
Setup and Preliminaries

**(Deterministic) Feedforward DNN:** Each layer $T_\ell = f_\ell(T_{\ell-1})$

- **Joint Distribution:** $P_{X,Y} \implies P_{X,Y} \cdot P_{T_1,\ldots,T_L|X}$
- **Information Plane:** Evolution of $(I(X;T_\ell), I(Y;T_\ell))$ during training
**Setup and Preliminaries**

**Deterministic Feedforward DNN:** Each layer $T_\ell = f_\ell(T_{\ell-1})$

- **Joint Distribution:** $P_{X,Y} \Rightarrow P_{X,Y} \cdot P_{T_1,\ldots,T_L|X}$
- **Information Plane:** Evolution of $(I(X;T_\ell), I(Y;T_\ell))$ during training

\[
I(A;B) = D_{KL}(P_{A,B} \parallel P_A \otimes P_B)^{\text{Discrete}} = \sum_{a,b} P_{A,B}(a,b) \log \frac{P_{A,B}(a,b)}{P_A(a)P_B(b)} \]
**Setup and Preliminaries**

**(Deterministic) Feedforward DNN:** Each layer $T_\ell = f_\ell(T_{\ell-1})$

**IB Theory Claim:** Training comprises 2 phases
**Setup and Preliminaries**

**(Deterministic) Feedforward DNN:** Each layer $T_\ell = f_\ell(T_{\ell-1})$

---

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

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

(Deterministic) Feedforward DNN: Each layer $T_\ell = f_\ell(T_{\ell-1})$

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

1. **Fitting:** $I(Y; T_\ell) \& I(X; T_\ell)$ rise (short)
2. **Compression:** $I(X; T_\ell)$ slowly drops (long)
(Deterministic) Feedforward DNN: Each layer $T_\ell = f_\ell(T_{\ell-1})$

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

1. **Fitting:** $I(Y; T_\ell) \& I(X; T_\ell)$ rise (short)
2. **Compression:** $I(X; T_\ell)$ slowly drops (long)
Proposition (Informal)

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

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

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

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

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

- \( I(X; T_\ell) \) a.s. infinite (continuous \( X \)) or constant \( H(X) \) (discrete \( X \))
Proposition (Informal)

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

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

- \( I(X; T_\ell) \) a.s. infinite (continuous \( X \)) or constant \( H(X) \) (discrete \( X \))

Feature Space (\( X \))

\[ X \sim \text{Unif}(\mathcal{X}) \]

\[ |\mathcal{X}| = 3000 \]
Proposition (Informal)

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

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

- \( I(X; T_\ell) \) a.s. infinite (continuous \( X \)) or constant \( H(X) \) (discrete \( X \))

\[
\begin{align*}
X &\sim \text{Unif}(\mathcal{X}) \\
|\mathcal{X}| &= 3000 \\
T_\ell &\sim \text{Unif}(\mathcal{T}_\ell) \\
|\mathcal{T}_\ell| &= |\mathcal{X}| = 3000
\end{align*}
\]
Proposition (Informal)

\textbf{Det. DNNs with strictly monotone nonlinearities (e.g., tanh or sigmoid)}

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

- \( I(X; T_\ell) \) a.s. \textbf{infinite} (continuous \( X \)) or \textbf{constant} \( H(X) \) (discrete \( X \))

- \textbf{Past Works:} Use binning-based proxy of \( I(X; T_\ell) \) (aka quantization)
**Proposition (Informal)**

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

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

- \( I(X; T_\ell) \) a.s. infinite (continuous \( X \)) or constant \( H(X) \) (discrete \( X \))

- **Past Works:** Use binning-based proxy of \( I(X; T_\ell) \) (aka quantization)
  - For non-negligible bin size \( I(X; \text{Bin}(T_\ell)) \neq I(X; T_\ell) \)
Proposition (Informal)

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

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

- \( I(X;T_\ell) \) a.s. **infinite** (continuous \( X \)) or **constant** \( H(X) \) (discrete \( X \))

- **Past Works:** Use binning-based proxy of \( I(X;T_\ell) \) (aka quantization)
  1. For non-negligible bin size \( I(X;\text{Bin}(T_\ell)) \neq I(X;T_\ell) \)
  2. \( I(X;\text{Bin}(T_\ell)) \) highly sensitive to user-defined bin size:
Proposition (Informal)

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

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

- \( I(X; T_\ell) \) a.s. **infinite** (continuous \( X \)) or **constant** \( H(X) \) (discrete \( X \))

**Past Works:** Use binning-based proxy of \( I(X; T_\ell) \) (aka quantization)

1. For non-negligible bin size \( I(X; \text{Bin}(T_\ell)) \neq I(X; T_\ell) \)
2. \( I(X; \text{Bin}(T_\ell)) \) highly sensitive to user-defined bin size:
Proposition (Informal)

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

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

- \( I(X; T_\ell) \) a.s. infinite (continuous \( X \)) or constant \( H(X) \) (discrete \( X \))
- **Past Works:** Use binning-based proxy of \( I(X; T_\ell) \) (aka quantization)
  - For non-negligible bin size \( I(X; \text{Bin}(T_\ell)) \neq I(X; T_\ell) \)
  - \( I(X; \text{Bin}(T_\ell)) \) highly sensitive to user-defined bin size:

*Real Problem:* Mismatch between \( I(X; T_\ell) \) measurement and model
**Modification:** Inject (small) Gaussian noise to neurons’ output

[Goldfeld-Berg-Greenewald-Melnyk-Nguyen-Kingsbury-Polyanskiy’18]
**Modification:** Inject (small) Gaussian noise to neurons’ output

[Goldfeld-Berg-Greenewald-Melnyk-Nguyen-Kingsbury-Polyanskiy’18]

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

[Goldfeld-Berg-Greenewald-Melnyk-Nguyen-Kingsbury-Polyanskiy’18]

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

\[
T_{\ell-1} \xrightarrow{\sigma(W^{(k)}_\ell T_{\ell-1} + b_\ell(k))} S_\ell(k) \xrightarrow{+} T_\ell(k) \\
Z_\ell(k) \sim \mathcal{N}(0, \sigma^2)
\]

$\implies X \mapsto T_\ell$ is a **parametrized channel** (by DNN’s parameters)
Modification: Inject (small) Gaussian noise to neurons’ output

[Goldfeld-Berg-Greenewald-Melnyk-Nguyen-Kingsbury-Polyanskiy’18]

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

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

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

\( X \mapsto T_\ell \) is a **parametrized channel** (by DNN’s parameters)

\( I(X; T_\ell) \) is a **function** of weights and biases!
Estimating $I(X; T_{\ell})$ in Noisy DNNs
Noisy DNN:

\[ X \xrightarrow{f_1} S_1 \xrightarrow{+} T_1 \xrightarrow{f_2} S_2 \xrightarrow{+} T_2 \ldots \]

\[ Z_1 \]

\[ Z_2 \]
Mutual Information Estimation in Noisy DNNs

**Noisy DNN:** \( S_\ell \triangleq f_\ell(T_{\ell-1}) \)

\[ X \xrightarrow{f_1} S_1 \xrightarrow{T_1} T_2 \ldots \]
Noisy DNN: \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)
Mutual Information Estimation in Noisy DNNs

**Noisy DNN:** \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)

\[
\begin{array}{c}
X \quad f_1 \quad S_1 \quad Z_1 \quad f_2 \quad S_2 \quad Z_2 \quad T_1 \quad f_3 \quad S_3 \quad Z_3 \quad T_2 \quad \cdots
\end{array}
\]

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

**Noisy DNN:** \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \ Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)

\[ X \xrightarrow{\mathbf{f}_1} S_1 \xrightarrow{+} T_1 \xrightarrow{\mathbf{f}_2} S_2 \xrightarrow{+} T_2 \cdots \]

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

\[ \implies \text{Mutual Information:} \quad I(X; T_\ell) = h(T_\ell) - \frac{1}{m} \sum_{i=1}^m h(T_\ell | X = x_i) \]
Mutual Information Estimation in Noisy DNNs

**Noisy DNN:** \[ S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2I_d) \]

\[ \begin{array}{cccc}
X & \rightarrow & f_1 & \rightarrow & S_1 & \rightarrow & T_1 & \rightarrow & f_2 & \rightarrow & S_2 & \rightarrow & T_2 & \cdots \\
& & Z_1 & & & & & & Z_2 & & & & & & \\
\end{array} \]

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

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

- **Structure:** \( S_\ell \perp Z_\ell \implies T_\ell = S_\ell + Z_\ell \sim P \ast \mathcal{N}_{\sigma} \)
Mutual Information Estimation in Noisy DNNs

Noisy DNN: \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)

\[ \begin{array}{c}
X \xrightarrow{f_1} S_1 \xrightarrow{\oplus} T_1 \xrightarrow{f_2} S_2 \xrightarrow{\oplus} T_2 \ldots
\end{array} \]

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

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

- Structure: \( S_\ell \perp Z_\ell \implies T_\ell = S_\ell + Z_\ell \sim \mathcal{P} \ast \mathcal{N}_\sigma \)
**Mutual Information Estimation in Noisy DNNs**

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

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

**Structure:** 
\( S_\ell \perp Z_\ell \implies T_\ell = S_\ell + Z_\ell \sim P * \mathcal{N}_\sigma \)
Mutual Information Estimation in Noisy DNNs

**Noisy DNN:** $S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d)$

\[
\begin{array}{c}
X \xrightarrow{f_1} S_1 \xrightarrow{\oplus} T_1 \xrightarrow{f_2} S_2 \xrightarrow{\oplus} T_2 \ldots
\end{array}
\]

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

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

- **Structure:** $S_\ell \perp Z_\ell \implies T_\ell = S_\ell + Z_\ell \sim P \ast \mathcal{N}_\sigma$

- **Know** the distribution $\mathcal{N}_\sigma$ of $Z_\ell$ (noise injected by design)
Mutual Information Estimation in Noisy DNNs

Noisy DNN: \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)

\[ X \xrightarrow{f_1} S_1 \xrightarrow{} T_1 \xrightarrow{f_2} S_2 \xrightarrow{} T_2 \cdots \]

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

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

- **Structure:** \( S_\ell \perp Z_\ell \implies T_\ell = S_\ell + Z_\ell \sim P * \mathcal{N}_\sigma \)

- **Know** the distribution \( \mathcal{N}_\sigma \) of \( Z_\ell \) (noise injected by design)
Mutual Information Estimation in Noisy DNNs

Noisy DNN: \( S_\ell \triangleq f_\ell(T_{\ell-1}) \quad \implies \quad T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)

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

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

- **Structure:** \( S_\ell \perp Z_\ell \quad \implies \quad T_\ell = S_\ell + Z_\ell \sim P * \mathcal{N}_\sigma \)

- **Know** the distribution \( \mathcal{N}_\sigma \) of \( Z_\ell \) (noise injected by design)

- **Extremely complicated** \( P \quad \implies \text{Treat as unknown} \)
Mutual Information Estimation in Noisy DNNs

Noisy DNN: \( S_\ell \triangleq f_\ell(T_{\ell-1}) \implies T_\ell = S_\ell + Z_\ell, \quad Z_\ell \sim \mathcal{N}(0, \sigma^2 I_d) \)

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

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

- **Structure:** \( S_\ell \perp Z_\ell \implies T_\ell = S_\ell + Z_\ell \sim P \ast \mathcal{N}_\sigma \)

- **Know** the distribution \( \mathcal{N}_\sigma \) of \( Z_\ell \) (noise injected by design)

- **Extremely complicated** \( P \implies \text{Treat as unknown} \)

- **Easily** get i.i.d. samples from \( P \) via DNN forward pass
Estimate $h(P \ast \mathcal{N}_\sigma)$ via $n$ i.i.d. samples $S^n \triangleq (S_i)_{i=1}^n$ from unknown $P \in \mathcal{F}_d$ (nonparametric class) and knowledge of $\mathcal{N}_\sigma$ (noise distribution).
### Differential Entropy Estimation under Gaussian Convolutions

Estimate $h(P \ast \mathcal{N}_\sigma)$ via $n$ i.i.d. samples $S^n \triangleq (S_i)_{i=1}^n$ from unknown $P \in \mathcal{F}_d$ (nonparametric class) and knowledge of $\mathcal{N}_\sigma$ (noise distribution).

**Nonparametric Class:** Specified by DNN architecture ($d = T_\ell$ ‘width’)
Structured Estimator (with Implementation in Mind)

Differential Entropy Estimation under Gaussian Convolutions

Estimate \( h(P \ast \mathcal{N}_\sigma) \) via \( n \) i.i.d. samples \( S^n \triangleq (S_i)_{i=1}^n \) from unknown \( P \in \mathcal{F}_d \) (nonparametric class) and knowledge of \( \mathcal{N}_\sigma \) (noise distribution).

**Nonparametric Class:** Specified by DNN architecture \((d = T_\ell \text{ ‘width’})\)

**Goal:** Simple & parallelizable for efficient implementation
Structured Estimator (with Implementation in Mind)

<table>
<thead>
<tr>
<th>Differential Entropy Estimation under Gaussian Convolutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate $h(P \ast \mathcal{N}<em>\sigma)$ via $n$ i.i.d. samples $S^n \triangleq (S_i)</em>{i=1}^n$ from unknown $P \in \mathcal{F}<em>d$ (nonparametric class) and knowledge of $\mathcal{N}</em>\sigma$ (noise distribution).</td>
</tr>
</tbody>
</table>

**Nonparametric Class:** Specified by DNN architecture ($d = T_\ell$ ‘width’)

**Goal:** Simple & parallelizable for efficient implementation

**Estimator:** $\hat{h}(S^n, \sigma) \triangleq h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma)$, where $\hat{P}_{S^n} \triangleq \frac{1}{n} \sum_{i=1}^{n} \delta_{S_i}$
Structured Estimator (with Implementation in Mind)

Nonparametric Class: Specified by DNN architecture ($d = T\ell$ ‘width’)

Goal: Simple & parallelizable for efficient implementation

Estimator: $\hat{h}(S^n, \sigma) \triangleq h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma)$, where $\hat{P}_{S^n} \triangleq \frac{1}{n} \sum_{i=1}^{n} \delta_{S_i}$

- Plug-in: $\hat{h}$ is plug-in est. for the functional $T_\sigma(P) \triangleq h(P \ast \mathcal{N}_\sigma)$

Differential Entropy Estimation under Gaussian Convolutions

Estimate $h(P \ast \mathcal{N}_\sigma)$ via $n$ i.i.d. samples $S^n \triangleq (S_i)_{i=1}^{n}$ from unknown $P \in \mathcal{F}_d$ (nonparametric class) and knowledge of $\mathcal{N}_\sigma$ (noise distribution).
Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)

For any $\sigma > 0$, $d \geq 1$, we have

$$\sup_{P \in \mathcal{F}_{d,K}^{(SG)}} \mathbb{E}\left| h(P \ast N_{\sigma}) - h(\hat{P}_{S_n} \ast N_{\sigma}) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}$$

where $C_{\sigma,d,K} = O_{\sigma,K}(c^d)$ for a constant $c$. 
Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)

For any $\sigma > 0$, $d \geq 1$, we have

$$\sup_{P \in \mathcal{F}^{(SG)}_{d,K}} \mathbb{E} \left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}$$

where $C_{\sigma,d,K} = O_{\sigma,K}(c^d)$ for a constant $c$.

Comments:
Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)

For any $\sigma > 0$, $d \geq 1$, we have

$$\sup_{P \in \mathcal{F}_{d,K}^{(SG)}} \mathbb{E} \left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_S \ast \mathcal{N}_\sigma) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}$$

where $C_{\sigma,d,K} = O_{\sigma,K}(c^d)$ for a constant $c$.

Comments:

- **Explicit Expression:** Enables concrete error bounds in simulations
Structured Estimator - Convergence Rate

**Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)**

For any \( \sigma > 0, \ d \geq 1, \) we have

\[
\sup_{P \in \mathcal{F}^{(SG)}_{d,K}} \mathbb{E} \left| h(P * \mathcal{N}_\sigma) - h(\hat{P}_n * \mathcal{N}_\sigma) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}
\]

where \( C_{\sigma,d,K} = O_{\sigma,K}(c^d) \) for a constant \( c. \)

**Comments:**

- **Explicit Expression:** Enables concrete error bounds in simulations

\[
C_{\sigma,d,K} = \frac{4}{\sigma^2} \sqrt{32d^2K^4 + d(d + 2) \left( \frac{\sigma}{\sqrt{2}} + K \right)^4 \left( \left( \frac{1}{\sqrt{2}} + \frac{K}{\sigma} \right) e^{\frac{3}{8}} \right)^{\frac{d}{2}}}
\]
Structured Estimator - Convergence Rate

**Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)**

For any $\sigma > 0$, $d \geq 1$, we have

$$\sup_{P \in F_{d,K}^{(SG)}} \mathbb{E} \left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_{\mathcal{S}^n} \ast \mathcal{N}_\sigma) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}$$

where $C_{\sigma,d,K} = O_{\sigma,K}(c^d)$ for a constant $c$.

**Comments:**

- **Explicit Expression:** Enables concrete error bounds in simulations
- **Minimax Rate Optimal:** Attains parametric estimation rate $O(n^{-\frac{1}{2}})$
Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)

For any $\sigma > 0$, $d \geq 1$, we have

$$\sup_{P \in \mathcal{F}^{(SG)}_{d,K}} \mathbb{E} \left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}$$

where $C_{\sigma,d,K} = O_{\sigma,K}(c^d)$ for a constant $c$.

Comments:

- **Explicit Expression:** Enables concrete error bounds in simulations
- **Minimax Rate Optimal:** Attains parametric estimation rate $O(n^{-\frac{1}{2}})$

Proof (initial step): Based on [Polyanskiy-Wu’16]

$$\left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma) \right| \lesssim W_1(P \ast \mathcal{N}_\sigma, \hat{P}_{S^n} \ast \mathcal{N}_\sigma)$$
### Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)

For any $\sigma > 0$, $d \geq 1$, we have

$$\sup_{P \in \mathcal{F}^{(SG)}_{d,K}} \mathbb{E}\left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma) \right| \leq C_{\sigma,d,K} \cdot n^{-\frac{1}{2}}$$

where $C_{\sigma,d,K} = O_{\sigma,K}(c^d)$ for a constant $c$.

### Comments:

- **Explicit Expression**: Enables concrete error bounds in simulations
- **Minimax Rate Optimal**: Attains parametric estimation rate $O(n^{-\frac{1}{2}})$

### Proof (initial step): Based on [Polyanskiy-Wu’16]

$$\left| h(P \ast \mathcal{N}_\sigma) - h(\hat{P}_{S^n} \ast \mathcal{N}_\sigma) \right| \lesssim W_1(P \ast \mathcal{N}_\sigma, \hat{P}_{S^n} \ast \mathcal{N}_\sigma)$$

$\implies$ Analyze empirical 1-Wasserstein distance under Gaussian convolutions
\textbf{p-Wasserstein Distance:} For two distributions \(P\) and \(Q\) on \(\mathbb{R}^d\) and \(p \geq 1\)

\[ W_p(P, Q) \triangleq \inf \left( \mathbb{E} \| X - Y \|^p \right)^{1/p} \]

infimum over all couplings of \(P\) and \(Q\)
Empirical $W_1$ & The Magic of Gaussian Convolution

\textbf{$p$-Wasserstein Distance:} For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \triangleq \inf (\mathbb{E}\|X - Y\|^p)^{1/p}$$

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**
**Empirical $W_1$ & The Magic of Gaussian Convolution**

**$p$-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \triangleq \inf \left( \mathbb{E} \| X - Y \|^p \right)^{1/p}$$

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d$
**p-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \triangleq \inf \left( \mathbb{E} \|X - Y\|^p \right)^{1/p}$$

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d \Rightarrow$ i.i.d. Samples $(S_i)_{i=1}^n$
**$p$-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

\[
W_p(P, Q) \triangleq \inf \left( \mathbb{E} \|X - Y\|^p \right)^{1/p}
\]

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d \quad \rightarrow \quad$ i.i.d. Samples $(S_i)_{i=1}^n$
- Empirical distribution $\hat{P}_{Sn} \triangleq \frac{1}{n} \sum_{i=1}^n \delta_{S_i}$
**p-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \triangleq \inf \left( \mathbb{E}\|X - Y\|^p \right)^{1/p}$$

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d$ $\implies$ i.i.d. Samples $(S_i)_{i=1}^{n}$
- Empirical distribution $\hat{P}_{S^n} \triangleq \frac{1}{n} \sum_{i=1}^{n} \delta_{S_i}$

$\implies$ Dependence on $(n, d)$ of $\mathbb{E}W_1(P, \hat{P}_{S^n})$
**$p$-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \triangleq \inf (\mathbb{E}\|X - Y\|^p)^{1/p}$$

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d \implies$ i.i.d. Samples $(S_i)_{i=1}^n$
- Empirical distribution $\hat{P}_{Sn} \triangleq \frac{1}{n} \sum_{i=1}^n \delta_{S_i}$

$\implies$ Dependence on $(n, d)$ of $\mathbb{E}W_1(P, \hat{P}_{Sn}) \gtrsim n^{-\frac{1}{d}}$
**p-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

\[
W_p(P, Q) \triangleq \inf \left( \mathbb{E} \| X - Y \|^p \right)^{1/p}
\]

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d \implies$ i.i.d. Samples $(S_i)_{i=1}^n$
- Empirical distribution $\hat{P}_{Sn} \triangleq \frac{1}{n} \sum_{i=1}^n \delta_{S_i}$

\[\implies\text{Dependence on } (n, d) \text{ of } \mathbb{E} W_1(P, \hat{P}_{Sn}) \gtrsim n^{-\frac{1}{d}}\]
\textbf{Empirical $W_1$ & The Magic of Gaussian Convolution}

\textbf{$p$-Wasserstein Distance:} For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \equiv \inf (\mathbb{E}\|X - Y\|^p)^{1/p}$$

infimum over all couplings of $P$ and $Q$

\textbf{Empirical 1-Wasserstein Distance:}

- Distribution $P$ on $\mathbb{R}^d \implies$ i.i.d. Samples $(S_i)_{i=1}^n$
- Empirical distribution $\hat{P}_{Sn} \equiv \frac{1}{n} \sum_{i=1}^n \delta_{S_i}$

$\implies$ Dependence on $(n, d)$ of $\mathbb{E}W_1(P, \hat{P}_{Sn}) \gtrsim n^{-\frac{1}{d}}$

\textbf{Theorem (Goldfeld-Greenewald-Weed-Polyanskiy’19)}

For any $d$, we have $\mathbb{E}W_1(P \ast \mathcal{N}_\sigma, \hat{P}_{Sn} \ast \mathcal{N}_\sigma) \leq O_{\sigma, d}(n^{-\frac{1}{2}})$
**$p$-Wasserstein Distance:** For two distributions $P$ and $Q$ on $\mathbb{R}^d$ and $p \geq 1$

$$W_p(P, Q) \triangleq \inf \left( \mathbb{E}\|X - Y\|^p \right)^{1/p}$$

infimum over all couplings of $P$ and $Q$

**Empirical 1-Wasserstein Distance:**

- Distribution $P$ on $\mathbb{R}^d \implies$ i.i.d. Samples $(S_i)_{i=1}^n$
- Empirical distribution $\hat{P}_{Sn} \triangleq \frac{1}{n} \sum_{i=1}^n \delta_{S_i}$

$\implies$ Dependence on $(n, d)$ of $\mathbb{E}W_1(P, \hat{P}_{Sn}) \gtrsim n^{-\frac{1}{d}}$

**Theorem (Goldfeld-Greenewald-Weed-Polyanskiy ’19)**

For any $d$, we have

$$\mathbb{E}W_1(P \ast N_\sigma, \hat{P}_{Sn} \ast N_\sigma) \leq O_{\sigma,d}(n^{-\frac{1}{2}}) = O_{\sigma}(c^d n^{-\frac{1}{2}})$$
Is Exponentiality in Dimension Necessary?

**Theorem (Goldfeld-Greenewald-Polyanskiy’18)**

For any $\sigma > 0$, sufficiently large $d$ and sufficiently small $\eta > 0$, we have

$$n^*(\eta, \sigma, F_d) = \Omega\left(\frac{2\gamma(\sigma)^d}{\eta^d}\right),$$

where $\gamma(\sigma) > 0$ is monotonically decreasing in $\sigma$. 

Theorem (Goldfeld-Greenewald-Polyanskiy’18)

For any $\sigma > 0$, sufficiently large $d$ and sufficiently small $\eta > 0$, we have

$$n^*(\eta, \sigma, \mathcal{F}_d) = \Omega\left(\frac{2\gamma(\sigma)^d}{\eta d}\right),$$

where $\gamma(\sigma) > 0$ is monotonically decreasing in $\sigma$.

$$\implies O\left(\frac{c^d}{\sqrt{n}}\right)$$

rate attained by the plugin estimator is sharp in $n$ and $d$.
Theorem (Goldfeld-Greenewald-Polyanskiy’18)

For any $\sigma > 0$, sufficiently large $d$ and sufficiently small $\eta > 0$, we have $n^*(\eta, \sigma, \mathcal{F}_d) = \Omega \left( \frac{2^{\gamma(\sigma)d}}{\eta d} \right)$, where $\gamma(\sigma) > 0$ is monotonically decreasing in $\sigma$.

$\implies O \left( \frac{c^d}{\sqrt{n}} \right)$ rate attained by the plugin estimator is sharp in $n$ and $d$

Proof (main ideas):
Theorem (Goldfeld-Greenewald-Polyanskiy’18)

For any $\sigma > 0$, sufficiently large $d$ and sufficiently small $\eta > 0$, we have

$$n^*(\eta, \sigma, F_d) = \Omega\left(\frac{2^{\gamma(\sigma)d}}{\eta^d}\right),$$

where $\gamma(\sigma) > 0$ is monotonically decreasing in $\sigma$.

$\Rightarrow O\left(\frac{c^d}{\sqrt{n}}\right)$ rate attained by the plugin estimator is sharp in $n$ and $d$

Proof (main ideas):

- Relate $h(P * \mathcal{N}_\sigma)$ to Shannon entropy $H(Q)$

  $\text{supp}(Q) =$ peak-constrained AWGN capacity achieving codebook $C_d$
Is Exponentiality in Dimension Necessary?

**Theorem (Goldfeld-Greenewald-Polyanskiy’18)**

For any $\sigma > 0$, sufficiently large $d$ and sufficiently small $\eta > 0$, we have

$$n^*(\eta, \sigma, \mathcal{F}_d) = \Omega\left(\frac{2^{\gamma(\sigma)d}}{\eta d}\right),$$

where $\gamma(\sigma) > 0$ is monotonically decreasing in $\sigma$.

$$\implies O\left(\frac{c^d}{\sqrt{n}}\right)$$

rate attained by the plugin estimator is sharp in $n$ and $d$

**Proof (main ideas):**

- Relate $h(P \ast \mathcal{N}_\sigma)$ to Shannon entropy $H(Q)$
  $\text{supp}(Q) =$ peak-constrained AWGN capacity achieving codebook $C_d$

- $H(Q)$ estimation sample complexity $\Omega\left(\frac{|C_d|}{\eta \log |C_d|}\right)$ [Valiant-Valiant’10]
Back to Noisy DNNs
Back to Noisy DNNs

✓ Simple-to-compute & Parallelizable estimator for $I(X; T_\ell)$
Back to Noisy DNNs

✓ Simple-to-compute & Parallelizable estimator for $I(X; T_\ell)$

✓ Statistically minimax rate optimal
Single Neuron Classification:

\[ I(X; T_\ell) \] Dynamics - Illustrative Minimal Example

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

**Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$

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

$$X \xrightarrow{\text{tanh}(wX + b)} S_{w,b} \xrightarrow{+} T \xrightarrow{} Z \sim \mathcal{N}(0, \sigma^2)$$

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

**Input:**  $X \sim \text{Unif}\{\pm 1, \pm 3\}$

$X_{y=-1} \triangleq \{-3, -1, 1\}$, $X_{y=1} \triangleq \{3\}$
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$

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

- **Output:**
  
  $Z \sim \mathcal{N}(0, \sigma^2)$

- **Center & sharpen transition:** (increase $w$ and keep $b = -2w$)

$$
\begin{align*}
  X & \xrightarrow{\text{tanh}(wX + b)} S_{w,b} \\
  S_{w,b} & \xrightarrow{+} T \\
  Z & \sim \mathcal{N}(0, \sigma^2)
\end{align*}
$$
**Single Neuron Classification:**

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$
  
  $X_{y=-1} \triangleq \{-3, -1, 1\}$, $X_{y=1} \triangleq \{3\}$
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$
  
  $\mathcal{X}_{y=-1} \triangleq \{-3, -1, 1\}$, $\mathcal{X}_{y=1} \triangleq \{3\}$

\[
S_{1,0} \quad X \xrightarrow{\text{tanh}(wX + b)} S_{w,b} \xrightarrow{} T \quad Z \sim \mathcal{N}(0, \sigma^2)
\]

- Correct classification performance
Single Neuron Classification:

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$
  
  $\mathcal{X}_{y=-1} \triangleq \{-3, -1, 1\}$, $\mathcal{X}_{y=1} \triangleq \{3\}$

- **Mutual Information:**
Single Neuron Classification:

**Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$

$x_{y=-1} \triangleq \{-3, -1, 1\}$, $x_{y=1} \triangleq \{3\}$

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

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$
  $$X_{y=-1} \triangleq \{-3, -1, 1\}, \ X_{y=1} \triangleq \{3\}$$

- **Mutual Information:**
  $$I(X; T) = I(S_{w,b}; S_{w,b} + Z)$$

$$\implies I(X; T)$$ is $\#$ bits (nats) transmittable over AWGN with symbols

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

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$
  
  $X_{y=-1} \triangleq \{-3, -1, 1\}$, $X_{y=1} \triangleq \{3\}$

- **Mutual Information:** $I(X; T) = I(S_{w,b}; S_{w,b} + Z)$

$\implies I(X; T)$ is \# bits (nats) transmittable over AWGN with symbols

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

- **Input:** \( X \sim \text{Unif}\{\pm 1, \pm 3\} \)
  
  \[ \mathcal{X}_{y=-1} \triangleq \{-3, -1, 1\}, \quad \mathcal{X}_{y=1} \triangleq \{3\} \]

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

\[ \Rightarrow \quad I(X; T)\text{ is \# bits (nats) transmittable over AWGN with symbols} \]

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

- **Input:** $X \sim \text{Unif}\{\pm 1, \pm 3\}$
  
  $\mathcal{X}_{y=-1} \triangleq \{-3, -1, 1\}$, $\mathcal{X}_{y=1} \triangleq \{3\}$

- **Mutual Information:** $I(X; T) = I(S_{w,b}; S_{w,b} + Z)$

  $\implies I(X; T)$ is \# bits (nats) transmittable over AWGN with 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 [Shwartz-Tishby’17]:

Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
Clustering of Representations - Larger Networks

Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\sigma = 0.01$
Clustering of Representations - Larger Networks

Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\sigma = 0.01$
Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\sigma = 0.01$
Clustering of Representations - Larger Networks

Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\sigma = 0.01$

★ weight orthonormality regularization
Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\sigma = 0.01$
- Verified in multiple additional experiments
  [Goldfeld-Berg-Greenewald-Melnyk-Nguyen-Kingsbury-Polyanskiy’18]
Noisy version of DNN from [Shwartz-Tishby’17]:

- **Binary Classification**: 12-bit input & 12–10–7–5–4–3–2 MLP arch.
- **Noise std.**: Set to $\sigma = 0.01$
- Verified in multiple additional experiments
  
  [Goldfeld-Berg-Greenewald-Melnyk-Nguyen-Kingsbury-Polyanskiy’18]

$\rightarrow$ Compression of $I(X; T_\ell)$ driven by clustering of representations
\( I(X; T_\ell) \) is constant/infinite \( \Rightarrow \) Doesn't measure clustering
Circling Back to Deterministic DNNs

\[ I(X; T_\ell) \text{ is constant/infinite} \implies \text{Doesn't measure clustering} \]

**Reexamining Past Measurements:** Computed \( I(X; \text{Bin}(T_\ell)) \)
Circling Back to Deterministic DNNs

$I(\mathbf{X}; T_\ell)$ is constant/infinite \implies \text{Doesn't measure clustering}

Reexamining Past Measurements: Computed $I(\mathbf{X}; \text{Bin}(T_\ell))$

- $T_\ell = \tilde{f}_\ell(\mathbf{X})$
Circling Back to Deterministic DNNs

\[ I(X; T_\ell) \text{ is constant/infinite} \implies \text{Doesn't measure clustering} \]

**Reexamining Past Measurements:** Computed \( I(X; \text{Bin}(T_\ell)) \)

- \( T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell)) \)
Circling Back to Deterministic DNNs

$I(X; T_\ell)$ is constant/infinite $\implies$ Doesn't measure clustering

Reexamining Past Measurements: Computed $I(X; \text{Bin}(T_\ell))$

- $T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell))$
- $H(\text{Bin}(T_\ell))$ measures clustering
Circling Back to Deterministic DNNs

\[ I(X; T_\ell) \text{ is constant/infinite } \implies \text{ Doesn't measure clustering} \]

Reexamining Past Measurements: Computed \( I(X; \text{Bin}(T_\ell)) \)

- \( T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell)) \)
- \( H(\text{Bin}(T_\ell)) \) measures clustering \( \uparrow \)
Circling Back to Deterministic DNNs

$I(X; T_\ell)$ is constant/infinite $\implies$ Doesn't measure clustering

Reexamining Past Measurements: Computed $I(X; \text{Bin}(T_\ell))$

- $T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell))$
- $H(\text{Bin}(T_\ell))$ measures clustering
Circling Back to Deterministic DNNs

$I(X; T_\ell)$ is constant/infinite $\implies$ Doesn't measure clustering

Reexamining Past Measurements: Computed $I(X; \text{Bin}(T_\ell))$

- $T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell))$
- $H(\text{Bin}(T_\ell))$ measures clustering

Test: $I(X; T_\ell)$ and $H(\text{Bin}(T_\ell))$ highly correlated in noisy DNNs*

* When bin size chosen $\propto$ noise std.
Circling Back to Deterministic DNNs

\[ I(X; T_{\ell}) \text{ is constant/infinite} \implies \text{Doesn't measure clustering} \]

**Reexamining Past Measurements:** Computed \( I(X; \text{Bin}(T_{\ell})) \)

- \( T_{\ell} = \tilde{f}_{\ell}(X) \implies I(X; \text{Bin}(T_{\ell})) = H(\text{Bin}(T_{\ell})) \)
- \( H(\text{Bin}(T_{\ell})) \) measures clustering

**Test:** \( I(X; T_{\ell}) \) and \( H(\text{Bin}(T_{\ell})) \) highly correlated in noisy DNNs*

\* When bin size chosen \( \propto \) noise std.
I(X; T_\ell) is constant/infinite \implies Doesn't measure clustering

Reexamining Past Measurements: Computed I(X; \text{Bin}(T_\ell))

- T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell))

- H(\text{Bin}(T_\ell)) measures clustering

Test: I(X; T_\ell) and H(\text{Bin}(T_\ell)) highly correlated in noisy DNNs

\implies Past works not measuring MI but clustering (via binned-MI)!
Circling Back to Deterministic DNNs

\[ I(X; T_\ell) \text{ is constant/infinite} \implies \text{Doesn't measure clustering} \]

**Reexamining Past Measurements:** Computed \( I(X; \text{Bin}(T_\ell)) \)

- \( T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell)) \)
- \( H(\text{Bin}(T_\ell)) \) measures clustering

**Test:** \( I(X; T_\ell) \) and \( H(\text{Bin}(T_\ell)) \) highly correlated in noisy DNNs*

\[ \implies \text{Past works not measuring MI but clustering (via binned-MI)!} \]

**Bi-Product Results:**
Circling Back to Deterministic DNNs

\[ I(X; T_\ell) \text{ is constant/infinite} \implies \text{Doesn't measure clustering} \]

**Reexamining Past Measurements:** Computed \( I(X; \text{Bin}(T_\ell)) \)

- \( T_\ell = \tilde{f}_\ell(X) \implies I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell)) \)
- \( H(\text{Bin}(T_\ell)) \text{ measures clustering} \)

**Test:** \( I(X; T_\ell) \text{ and } H(\text{Bin}(T_\ell)) \text{ highly correlated in noisy DNNs}^* \)

\[ \implies \text{Past works not measuring MI but clustering (via binned-MI)!} \]

**Bi-Product Results:**

1. Refute ‘compression (tight clustering) improves generalization’ claim
Circling Back to Deterministic DNNs

\[ I(X; T_\ell) \] is constant/infinite \( \Rightarrow \) Doesn't measure clustering

**Reexamining Past Measurements:** Computed \( I(X; \text{Bin}(T_\ell)) \)

- \( T_\ell = \tilde{f}_\ell(X) \) \( \Rightarrow \) \( I(X; \text{Bin}(T_\ell)) = H(\text{Bin}(T_\ell)) \)
- \( H(\text{Bin}(T_\ell)) \) measures clustering

**Test:** \( I(X; T_\ell) \) and \( H(\text{Bin}(T_\ell)) \) highly correlated in noisy DNNs

\[ \Rightarrow \text{Past works not measuring MI but clustering (via binned-MI)!} \]

**Bi-Product Results:**

1. Refute ‘compression (tight clustering) improves generalization’ claim
2. Computational feasibility of tracking clustering
Towards Broader Impact

Deep Learning
Towards Broader Impact

Deep Learning
Towards Broader Impact

Deep Learning
Towards Broader Impact

Deep Learning
Towards Broader Impact

Deep Learning

Design
Towards Broader Impact

Deep Learning

Design

Optimize
Towards Broader Impact

Deep Learning

Design
Optimize
Understand
Towards Broader Impact: Design

How to Better Design DNNs?
Towards Broader Impact: Design

How to Better Design DNNs?

- $T_\ell$ compression $\implies T_{\ell-1}$ linear separation
Towards Broader Impact: Design

How to Better Design DNNs?

- $T_\ell$ compression $\implies T_{\ell-1}$ linear separation
Towards Broader Impact: Design

How to Better Design DNNs?

- $T_\ell$ compression $\implies T_{\ell-1}$ linear separation

- Combine with efficient high-dimensional clustering measure
Towards Broader Impact: Design

How to Better Design DNNs?

- $T_\ell$ compression $\Rightarrow T_{\ell-1}$ linear separation

- Combine with efficient high-dimensional clustering measure

$\Rightarrow$ **Optimize architecture** by shedding redundant layers
Towards Broader Impact: Optimization

How to Better Train DNNs?
Towards Broader Impact: Optimization

How to Better Train DNNs?

- Regularize intermediate layer to increase $I(Y; T_\ell)$
How to Better Train DNNs?

- Regularize intermediate layer to increase $I(Y; T_\ell)$
- Learn well-separated (nonlinear) representations
Towards Broader Impact: Optimization

How to Better Train DNNs?

- Regularize intermediate layer to increase $I(Y; T_\ell)$
- Learn well-separated (nonlinear) representations

⇒ Enhanced algorithms for faster convergence
Towards Broader Impact: Understanding

How to Better Understand DNNs?
Towards Broader Impact: Understanding How to Better Understand DNNs?

1. **Channel Synthesis**: Quantify \#bits needed for ‘emulating’ a channel
How to Better Understand DNNs?

1. **Channel Synthesis**: Quantify \#bits needed for ‘emulating’ a channel

\[
X = \text{Dog} \quad \xrightarrow{P_{\hat{Y}_\Theta | x}} \quad \text{DNN}(\Theta) \quad \xrightarrow{} \quad \hat{Y}_\Theta = \text{Dog}
\]
Towards Broader Impact: Understanding

How to Better Understand DNNs?

1. **Channel Synthesis:** Quantify \#bits needed for ‘emulating’ a channel

   \[ P_{\hat{Y}_\Theta | X} \]

   \[ \text{DNN}(\Theta) \]

   \[ \hat{Y}_\Theta = \text{Dog} \]

   Compare DNN synthesis \#bits vs. \( \log(\#\text{classes}) \)
Towards Broader Impact: Understanding How to Better Understand DNNs?

1. **Channel Synthesis**: Quantify \( \# \text{bits} \) needed for ‘emulating’ a channel

\[
X = \begin{array}{c}
P_{\hat{Y}_\Theta|X} \\
\text{DNN}(\Theta) \\
\hat{Y}_\Theta = \text{Dog}
\end{array}
\]

- Compare **DNN synthesis \#bits** vs. \( \log(\#\text{classes}) \)

\( \Rightarrow \) **Scoring systems** for DNNs performing the same task
Towards Broader Impact: Understanding

How to Better Understand DNNs?

1. **Channel Synthesis**: Quantify \#bits needed for ‘emulating’ a channel

   \[
   X = \text{Dog} 
   \]

   - Compare **DNN synthesis \#bits vs. \log (\#classes)**

   \[P_{\hat{Y}_\Theta | X} \]  
   \[\text{DNN}(\Theta) \]  
   \[\hat{Y}_\Theta = \text{Dog} \]

   \[\Rightarrow \text{Scoring systems} \text{ for DNNs performing the same task} \]

2. **DNN Neural Activity**: Which are the ‘dog’ neurons?
Towards Broader Impact: Understanding

How to Better Understand DNNs?

1. **Channel Synthesis:** Quantify \#bits needed for ‘emulating’ a channel

   \[ P_{\hat{Y}_{\Theta}}|X \]

   \[ \text{DNN}(\Theta) \]

   \[ \hat{Y}_{\Theta} = \text{Dog} \]

   - Compare **DNN synthesis \#bits vs. \log (\#classes)**

   \[ \Rightarrow \text{Scoring systems} \text{ for DNNs performing the same task} \]

2. **DNN Neural Activity:** Which are the ‘dog’ neurons?

   - MI estimator convergence rate **independent of input dimension**!
Towards Broader Impact: Understanding How to Better Understand DNNs?

1. **Channel Synthesis**: Quantify \#bits needed for ‘emulating’ a channel

   \[
   X = \xrightarrow{P_{Y_\Theta | x}} \text{DNN}(\Theta) \rightarrow \hat{Y}_\Theta = \text{Dog}
   \]

   - Compare **DNN synthesis \#bits** vs. \(\log(\#\text{classes})\)
   - \(\Rightarrow\) **Scoring systems** for DNNs performing the same task

2. **DNN Neural Activity**: Which are the ‘dog’ neurons?

   - MI estimator convergence rate **independent of input dimension**!
   - Measure MI between \(X_{\text{Dog}}/X_{\text{Cat}}\) and **single** (pairs, triples of) neurons
Towards Broader Impact: Understanding

How to Better Understand DNNs?

1. **Channel Synthesis**: Quantify #bits needed for ‘emulating’ a channel

   \[
   X = \begin{array}{c}
   \text{Dog}
   \end{array} \quad \xrightarrow{P_{\hat{Y}_\Theta|x}} \quad \text{DNN}(\Theta) \quad \xrightarrow{} \quad \hat{Y}_\Theta = \text{Dog}
   \]

   - Compare DNN synthesis #bits vs. \( \log(\#\text{classes}) \)
   - **Scoring systems** for DNNs performing the same task

2. **DNN Neural Activity**: Which are the ‘dog’ neurons?

   - MI estimator convergence rate independent of input dimension!
   - Measure MI between \( X_{\text{Dog}}/X_{\text{Cat}} \) and single (pairs, triples of) neurons
   - **Heatmap** of DNN neural activity
**Goal:** Fundamental properties and opt. designs (math. modeling & solutions)
**Goal:** Fundamental properties and opt. designs (math. modeling & solutions)
Goal: Fundamental properties and opt. designs (math. modeling & solutions)
Goal: Fundamental properties and opt. designs (math. modeling & solutions)
**Goal:** Fundamental properties and opt. designs (math. modeling & solutions)

**Emerging Technologies:**
Shrink magnetic region per bit
Goal: Fundamental properties and opt. designs (math. modeling & solutions)

Emerging Technologies:
Shrink magnetic region per bit

Challenges:
Stabilization of written data
**Goal:** Fundamental properties and opt. designs (math. modeling & solutions)

- **Emerging Technologies:**
  - Shrink magnetic region per bit

- **Challenges:**
  - Stabilization of written data

- **Model & Study:**
  - Interacting particle sys.
  - Storage capacity & HDD designs
Goal: Fundamental properties and opt. designs (math. modeling & solutions)
Goal: Fundamental properties and opt. designs (math. modeling & solutions)

Physical-Layer Security (PLS):
Use noise in communication channel as security resource
Information Theory in the Age of Information

Goal: Fundamental properties and opt. designs (math. modeling & solutions)

Physical-Layer Security (PLS):
Use noise in communication channel as security resource

Practical Perspective
Beneficial properties but impractical assumptions
Information Theory in the Age of Information

**Goal:** Fundamental properties and opt. designs (math. modeling & solutions)

- **Processing**
  - **Physical-Layer Security (PLS):** Use noise in communication channel as security resource

**Practical Perspective**
Beneficial properties but impractical assumptions

**Work & Vision:**
Bridge gaps for interdisciplinary security paradigm