A Novel Entropy and Mutual Information Measure for High Dimensional Data and Deep Neural Networks

Chen Liu
chen.liu.cl2482@yale.edu

Abstract

This is neither a research article nor a review article. This is a casual explanation of our paper that introduced the concept of Diffusion Spectral Entropy (DSE) and Diffusion Spectral Mutual Information (DSMI) [1–3][1]. **Treat this as a blog.** You are welcome to cite the papers listed on the next page if you find this blog helpful.

TLDR: DSE and DSMI are not intended to be unbiased estimators of the *Shannon* entropy and mutual information, unlike earlier works such as Mutual Information Neural Estimation (MINE) [4] or Smoothed Mutual Information "Lower-bound" Estimator (SMILE) [5]. Instead, they serve as alternative measures based on the *von Neumann* formulation, which we found to be well-suited for high dimensional data and particularly descriptive for deep neural networks.

1 Introduction

In short, DSE and DSMI can be understood as an alternative realization of a classic work, the Information Bottleneck Principle (IB) [6] (cited 1685 times as of 07/19/2024).

We used a different definition to circumvent the widely criticized binning problem, allowing for a more reliable estimation of entropy and mutual information in deep neural networks.

Most existing entropy and mutual information estimators – for example, the methods in the original IB paper and in a popular follow-up work [7] – lost reliability when the number of neurons in a single layer of the neural network reached double digits, thus being limited to toy models. In contrast, our method can be effectively applied to truly practical neural networks, such as ResNet-50 with linear layers that have up to 2048 neurons.

2 Motivation and Background

2.1 What is the information bottleneck principle?

There have been many interpretations of this article before, such as Natalie Wolchover's blog [8].

In our humble opinion, one of the contributions of this article is the introduction of a framework for calculating the mutual information between different layers of a neural network.

If we use Z to denote the variable of a certain layer in the neural network, X to denote the input information, Y to denote the output information, and let $I(A; B)$ represent the mutual information between variables A and B.

[1] These are different versions of the same work. [1] is the final published version at an IEEE information theory conference. [2] is the preprint version with the most amount of details. [3] is an ICML workshop version.

Preprint. Under review.

A straightforward application of this article is to calculate the changing trends of $I(Z;X)$ and $I(Z;Y)$ during the neural network training process. Similarly, this can be extended to different intermediate layers Z_i to observe the changing trends of mutual information as we go deeper into the network.

As for what this represents and what applications it can enable, the IB paper [6] and some subsequent works provide various interpretations, showcasing a diversity of ideas.

2.2 What are the limitations of existing estimators?

According to Shannon, entropy is defined as Eqn (1) and mutual information is defined as Eqn (2).

$$\text{Shannon Entropy} \quad H(X) := -\mathbb{E}[-\log p(X)] = -\sum_{x \in X} p(x) \log p(x) \tag{1}$$

$$\text{Shannon MI} \quad I(X;Y) := H(X) - H(X|Y) = H(X) - \sum_{i} p(Y = y_i) H(X|Y = y_i) \tag{2}$$

When calculating entropy and mutual information, a central issue is how to estimate the probability distribution $p(X)$. Probably the earliest and most common method is ***quantization*** and ***binning***.

When your variable is a **scalar**, this method is very effective (as shown in Figure 1 left panel).

- For a **scalar**, if you quantize it into b intervals, there will be b different possible values (we call them buckets in our paper). As long as the sample size n is greater than b, there will be a bucket with a frequency greater than 1. We only need a reasonable sample size to estimate the probability distribution $p(x)$ fairly well.

However, when your variable is a **vector**, you encounter the curse of dimensionality.

- For a **vector** of dimension $1 \times D$, if you quantize each dimension into b intervals, you will end up with b^D buckets. When D is large, for any reasonably feasible sample size n, we almost always face a troublesome situation where the frequency of each bucket is less than or equal to 1. Consequently, our estimation of the probability distribution $p(x)$ turns into a uniform distribution (over the buckets with frequency 1). Clearly, this will result in a very inaccurate estimation in most cases. Consequently, our estimation of entropy and mutual information become unreliable.
- This problem has led to the use of very small toy models in the Information Bottleneck Principle [6] and some subsequent works. For example, in the well-known "On the information bottleneck theory of deep learning" [7] (cited 598 times as of 07/19/2024), the authors used the following model: "Model width is 12-10-7-5-4-3-2". This shows the significant impact of the curse of dimensionality, as even at ICLR 2018, such small models were still in use.

3 Proposed Entropy and Mutual Information Measures: DSE and DSMI

We took an alternative approach, bypassing the quantization and binning method.

3.1 Definition

According to von Neumann, entropy can be alternatively defined as Eqn (3), where ρ is a density matrix and ρ_i is its i-th eigenvalue.

$$\text{von Neumann Entropy} \quad H(\rho) := -tr(\rho \log \rho) = -\sum_{i} \rho_i \log \rho_i \tag{3}$$

We built on von Neumann's idea and proposed our new measures, Diffusion Spectral Entropy (DSE) as in Eqn (4) and Diffusion Spectral Mutual Information (DSMI) as in Eqn (5).

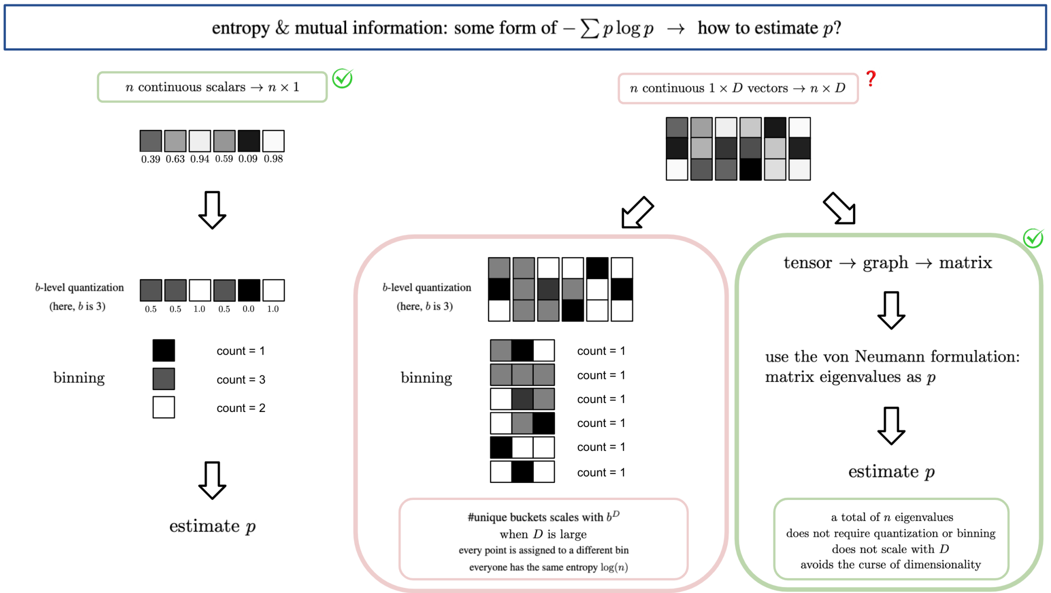
entropy & mutual information: some form of $-\sum p \log p$ → how to estimate p?
n continuous scalars → $n \times 1$
0.39 0.63 0.94 0.59 0.09 0.98
b-level quantization
(here, b is 3)
0.5 0.5 1.0 0.5 0.0 1.0
binning
count = 1
count = 3
count = 2
estimate p
n continuous $1 \times D$ vectors → $n \times D$
?
b-level quantization
(here, b is 3)
binning
count = 1
count = 1
count = 1
count = 1
count = 1
count = 1
#unique buckets scales with b^D
when D is large
every point is assigned to a different bin
everyone has the same entropy $\log(n)$
tensor → graph → matrix
use the von Neumann formulation:
matrix eigenvalues as p
estimate p
a total of n eigenvalues
does not require quantization or binning
does not scale with D
avoids the curse of dimensionality

Figure 1: Curse of dimensionality in Classic Shannon Entropy / Mutual Information.

$$\text{DSE} \quad S_D(\mathbf{P}, t) := -\sum_i \alpha_{i,t} \log \alpha_{i,t} \tag{4}$$

$$\text{DSMI} \quad I_D(X, Y) := S_D(\mathbf{P}_X, t) - \sum_{y_i \in Y} p(Y = y_i) S_D(\mathbf{P}_{X|Y=y_i}, t) \tag{5}$$

In the above definition,

- $\mathbf{P}$ is a diffusion matrix,
- $\mathbf{P}_X$ is the diffusion matrix built from data X,
- t (integer) is the diffusion time, and
- $\alpha_{i,t}$ is the i-th eigenvalue of the diffusion matrix after t diffusion iterations.

Here, the term "diffusion" is more closely tied to the concept of "diffusion maps" [9] or "diffusion condensation" [10, 11] as opposed to "diffusion models" [12] which is very popular lately.

3.2 Advantages

The main advantage compared to the Shannon counterpart is that, no matter how high dimensional the data are, we will eventually turn the data into a $n \times n$ matrix, leading to n eigenvalues. That is to say, this method can avoid the awkward situation where "no matter what we try, we will estimate $p(x)$ as a uniform distribution, unless we sample a prohibitively large number of data points". This feature can be shown in our simulations.

4 Theoretical Results

To keep this blog engaging, we would not include all the formulas for the theoretical part. Please do us a favor and refer to the paper [1, 2] if you are interested. Basically, we proved some simple properties of DSE and DSMI, such as upper and lower bounds.

An interesting proposition showed that **if the data distribution shifts from being a single cluster to multiple clusters, the upper bound of our defined DSE increases** (the exact amount of increase is complex to describe, but it increases).

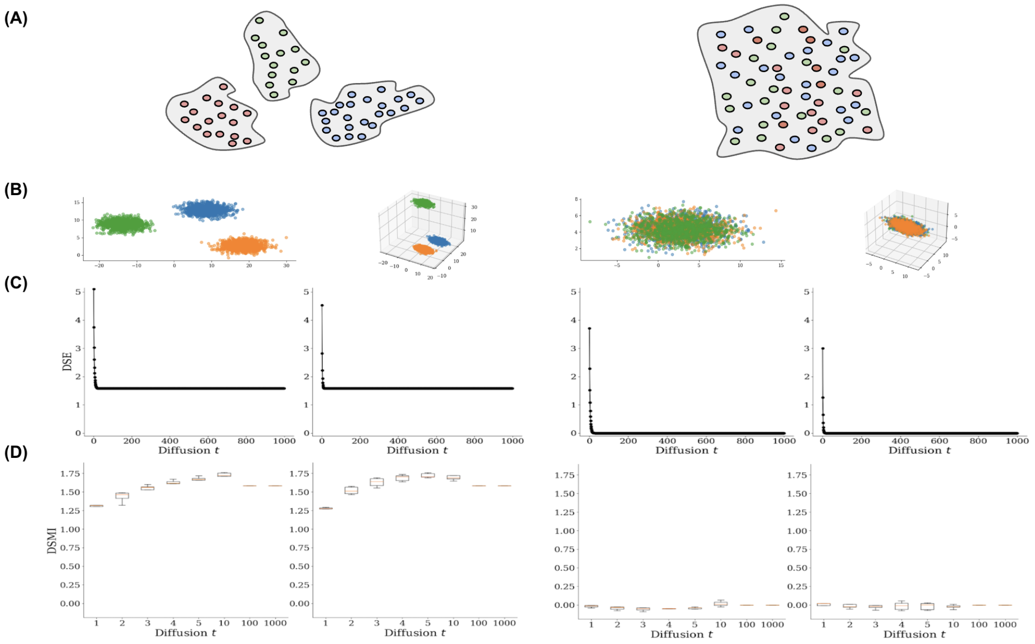

Figure 2: Intuition for DSE and DSMI. X: data, Y: class label. Colors represent classes. (A) sketch, (B) simulation, (C) DSE, (D) DSMI. In all panels, **Left:** X forms k clusters and Y corresponds to clusters. $S_D(\mathbf{P}_X, t) \approx \log(k)$ at sufficiently large t and $I_D(X;Y) > 0$. **Right:** X forms a single cluster and Y is randomly distributed over X. $S_D(\mathbf{P}_X, t) \approx 0$ at sufficiently large t and $I_D(X;Y) \approx 0$.

Think about it. Does the description of such data distribution changes remind you of something? Yes, it resembles the dynamics of latent representations of a classification model as it gradually learns from the data. For example, the latent space of a MNIST classifier would form a single blob upon initialization, but would eventually grow into 10 fairly distinct clusters representing the 10 digits. This proposition implies that, in theory, the upper bound of DSE will increase during classifier training.

Another interesting and important feature is that the definition of DSE focuses more on **the number of non-trivial feature directions** in the data distribution than the variance of data or noise level. This means DSE is more representative of the underlying structure of the data (such as the number of clusters, branches, etc.), which we believe is a beneficial property for deep learning research.

The manifold hypothesis suggests that data from natural systems can often be represented as originating from a low-dimensional manifold within a high-dimensional measurement space [13–15]. Understanding the structure of this manifold is crucial for unveiling deep neural networks. In this context, the novel DSE and DSMI can be particularly helpful.

Here, there should be an illustration to explain this matter. In Figure 2 below, when we set the diffusion time t sufficiently large, both DSE and DSMI converge to $\log(k)$ where k represents the number of independent structures in the data.

5 Empirical Results

5.1 Simulations

We can start with simulation experiments to visually demonstrate some qualitative and quantitative properties of DSE and DSMI, as summarized in Figure 3.

The upper panel of this figure presents several simulation experiments on DSE. On the left, it shows that as a Positive Semi-Definite (PSD) matrix approaches the identity matrix, its DSE increases. This is reasonable because the identity matrix indicates the maximum number of possible feature directions

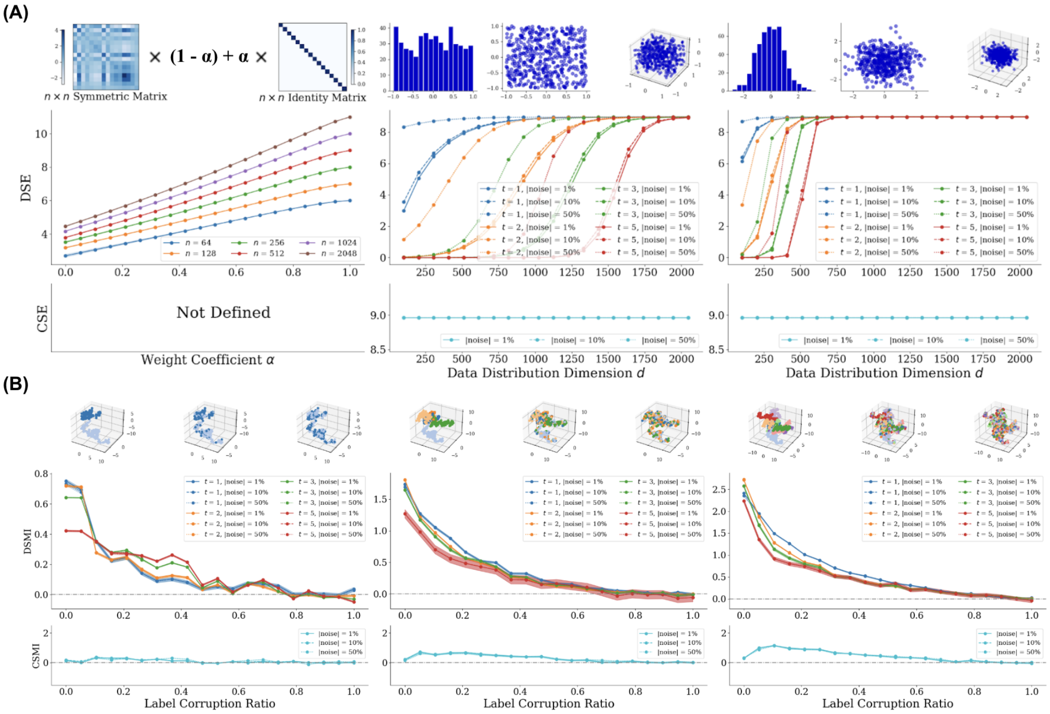

Figure 3: Diffusion Spectral Entropy (DSE) and Diffusion Spectral Mutual Information (DSMI) on toy data. **(A)** *DSE increases as intrinsic dimension grows, while classic Shannon entropy (CSE) saturates to* $\log(n)$ *due to curse of dimensionality.* **Left**: Weighted sum of a random $n \times n$ symmetric positive definite matrix (to simulate a diffusion matrix) and the $n \times n$ identity matrix. In theory, the identity matrix shall have the highest entropy at each respective n. **Mid**: d-dimensional $U[-1, 1]$ inside a 2048 dimensional space. **Right**: d-dimensional $\mathcal{N}(0, I)$ inside a 2048 dimensional space. Shaded areas indicate the standard deviation from 5 independent runs. For the latter two distributions, additive noise is injected into the coordinates and schematics for $d = \{1, 2, 3\}$ are illustrated on top. The number of data points for the simulation is 500. Note that in the latter two cases for DSE we compute the diffusion matrix of the data manifold prior to entropy evaluation, whereas in the first case we skip that step because the matrix is already provided. In the latter two cases, CSE saturates to $\log(n) = \log_2(500) = 8.966$. **(B)** *When two random variables are dependent, DSMI negatively correlates with the level of data corruption, while classic Shannon mutual information (CSMI) does not capture this trend.* DSMI $I_D(Z; Y)$ and CSMI are computed on synthetic, 20-dimensional trees with {2, 5, 10} branches (**Left, Mid, Right**). The x-axis represents the level of label corruption, ranging from completely clean label (left) to fully corrupted label (right), with 3 dimensional schematics demonstrating corruption ratios {0.0, 0.5, 1.0} displayed on top. At full corruption, $I_D(Z; Y)$ converges to zero, as the embedding vectors do not contain information on the labels.

or states, corresponding to the theoretical maximum value of DSE. On the right, it illustrates that as the intrinsic dimension d of a D-dimensional vector increases, the DSE also increases.

This might sound a bit convoluted, so consider an example. Suppose that we have a vector with an external dimension of 1000, but these dimensions might not be independent. For example, 800 of these dimensions could be redundant or linear combinations of other dimensions. In this case, D is 1000, while the intrinsic dimension d is 200.

The lower panel of this figure presents simulation experiments on DSMI. In this simulation, we used k-branched trees, where each branch corresponds to a class label. It can be observed that when the class labels are perfectly organized, the value of DSMI is positive. However, as we progressively shuffle the class labels, the value of DSMI gradually approaches zero. The three graphs correspond to binary trees, 5-branched trees, and 10-branched trees, respectively.

5.2 Comparisons on high dimensional data

Next, we conducted a set of comparative experiments to demonstrate that our proposed entropy and mutual information metrics remain robust in high dimensions. We primarily compared Diffusion Spectral Mutual Information (DSMI), Classical Shannon Mutual Information (CSMI), a toolkit based on the Kraskov method (NPEET) [16], and Mutual Information Neural Estimation (MINE) [4]. We also attempted several other methods, but have to exclude them since their publicly available code do not support calculating mutual information between high dimensional vectors.

Why compare mutual information instead of entropy? The reason is that (1) mutual information results are more intuitive, and (2) comparing entropy would exclude MINE from the analysis.

In Figure 4, the two panels on the left show that all methods perform reasonably well, as they gradually approach zero when the class labels are progressively shuffled. The rightmost panel is the core comparison, illustrating how each method performs as we increase the data dimension. It highlights that our method remains more robust with high dimensional data, decaying slower than the other methods.

As a side note, there are some inherent difficulties in conducting research on mutual information estimation that we would like to complain about — we usually do not have definitive ground truth, which itself is an important reason why this field is important. Especially in our case, when we completely use a different underlying mathematical formulation, it is difficult to determine whether our method is more accurate than others. All we can say is that it is (hopefully) theoretically sound, performs reasonably well in practice, and remains more robust with high dimensional data, making it suitable for modern deep neural networks.

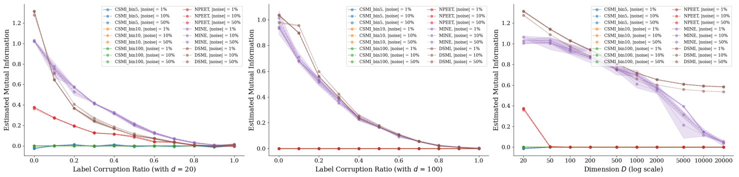

Figure 4: Mutual information estimation on Gaussian blobs.

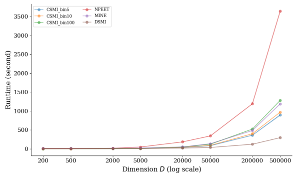

Figure 5: Runtime scaling.

Furthermore, we compared the runtime of the algorithms (Figure 5). As we increase the data dimensions, the runtime of DSMI increases the slowest compared to other methods. It is not in our benefit to say this, but we would like to point out that our method does not have a runtime advantage at very low dimensions. It just scales better as the dimensions increase.

5.3 Assessing DSE and DSMI in real neural network training

This was our most time-consuming experiment. Thanks to some reviewers' suggestions, we increased the scale of our experiments. We now have results from running 6 models on 3 datasets using 3 training methods, with each experiment repeated 3 times with different random seeds. Sounds like a lot, doesn't it? Especially for poor students like us without tens or hundreds of GPUs.

The 3 training methods are: supervised learning, self-supervised learning (SimCLR) [17], and deliberate overfitting on random labels.

In the following text, Z refers to the output of the final fully connected layer before the classification head. This is the layer typically used for visualizing dimensionality reduction effects or for linear probing.

First, we observed that $S_D(Z)$ (DSE of Z) increases during training (Figure 6). In both supervised and self-supervised learning, $S_D(Z)$ increases as training performance improves (Does the proposition we mentioned earlier ring a bell?). However, even with deliberate overfitting on random labels, where model performance is abysmal, $S_D(Z)$ still rises. In short, regardless of training quality, $S_D(Z)$ increases.

Next, we look at the mutual information with the true labels (Figure 7). **Simply put, for $I_D(Z;Y)$ (DSMI between Z and Y), models that are properly trained show an increase, whereas those overfitting to random labels do not.** This makes perfect sense.

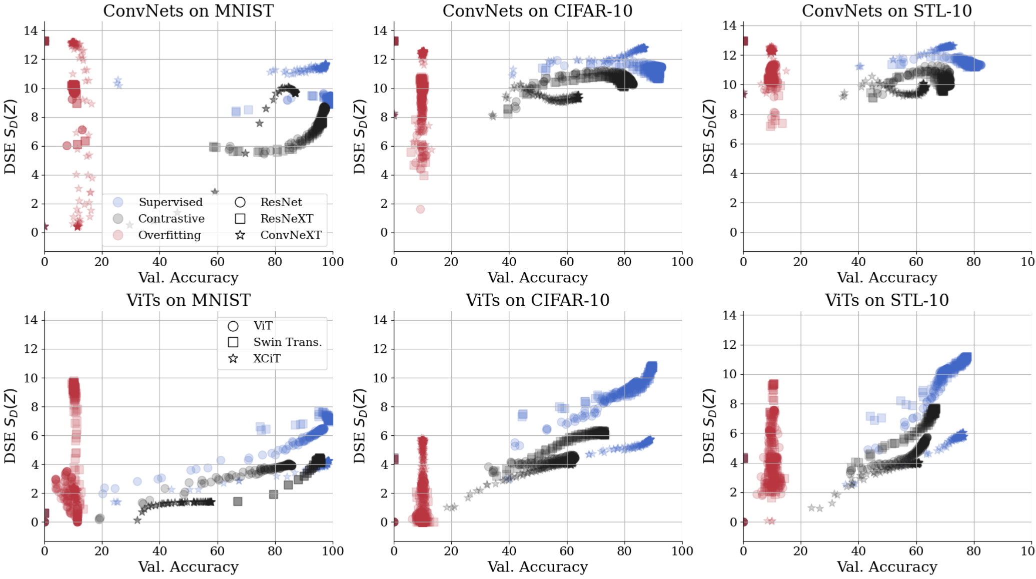

Figure 6: DSE $S_D(Z)$ of embedding vector Z.

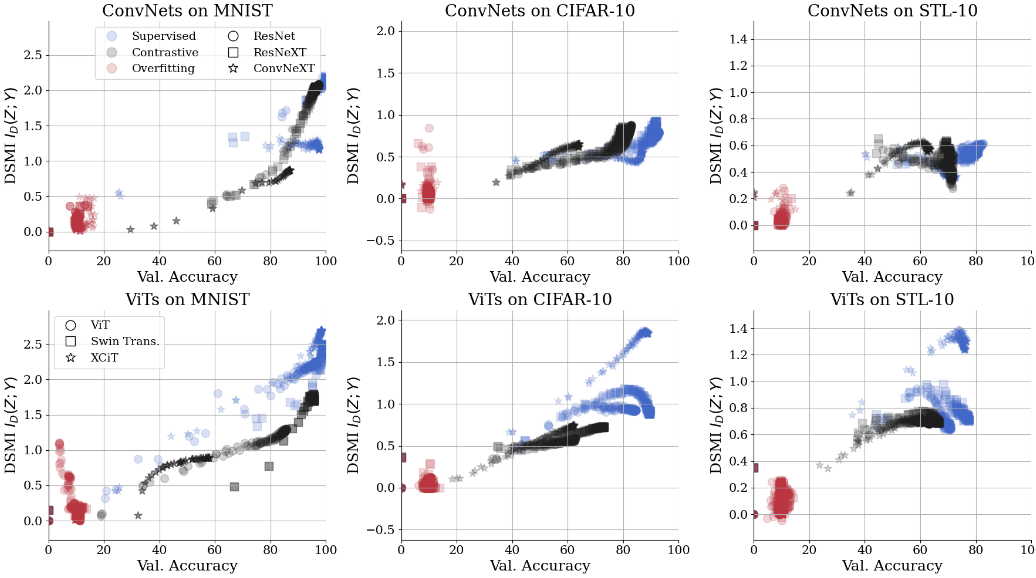

Figure 7: DSMI $I_D(Z;Y)$ between embedding vectors Z and the class label Y.

Finally, we examine the mutual information with the model inputs (Figure 8). The result is quite interesting. **For MNIST, $I_D(Z;X)$ continuously increases, whereas for CIFAR-10 and STL-10, it quickly peaks and then generally goes downward.** This is fascinating because the Information Bottleneck Principle paper suggests that the mutual information between Z and X should decrease, while the ICLR paper [7] refutes this view. We speculate that whether this mutual information should increase or decrease might depend on the dataset itself. For example, MNIST images might have simpler morphological information that is easier to learn (or it might even be a case of overfitting). This is an area that we leave to future researchers to investigate further.

Additionally, $I_D(Z;X)$ increases even in the overfitting experiments. We believe that this indicates that even when overfitting, the model still learns some features of the input images, even if those features are subsequently used to fit incorrect labels. This also makes sense.

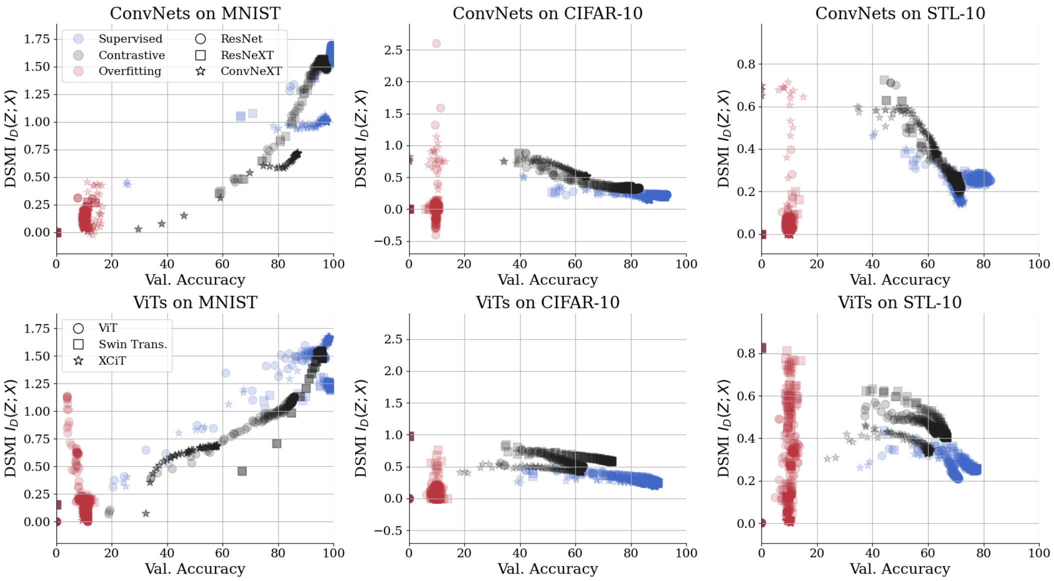

Figure 8: DSMI $I_D(Z;X)$ between Z and input X.

6 What Else Can We Do With DSE and DSMI?

Some might ask, apart from all these analyses, can your method actually do something practical? Some reviewers raised similar questions when we first submitted the paper to NeurIPS. To address this, we have prepared two utility studies to demonstrate the practical applications of our method.

6.1 Application 1: Using DSE to optimize network initialization

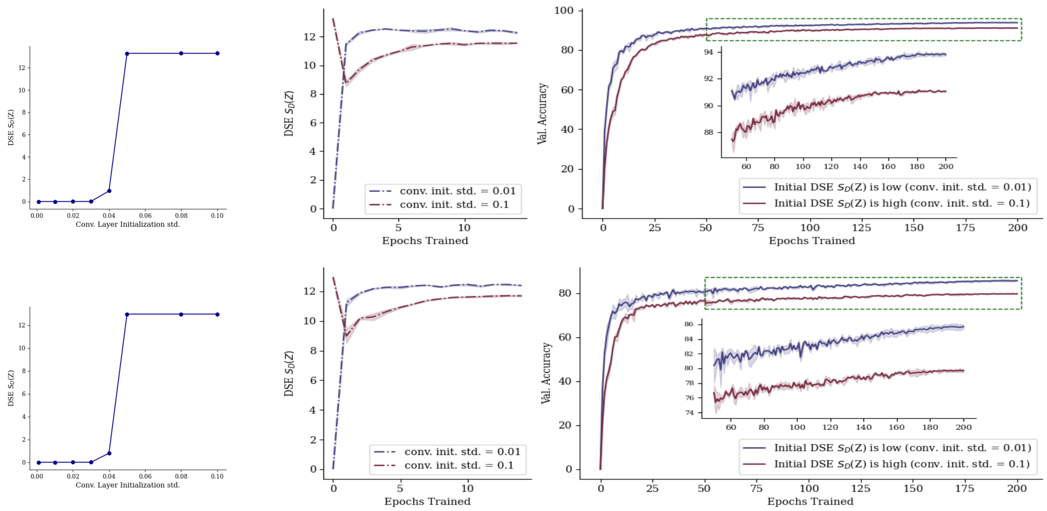

Figure 9: Initializing the network at a low DSE allows for faster convergence and better final performance. Top: CIFAR-10, bottom: STL-10. Shaded areas indicate standard deviation across 3 random seeds.

While analyzing the experimental data, we observed two intriguing phenomena: (1) Even when using identical code for initialization, different models exhibit different initial $S_D(Z)$ values. (2) If the initial $S_D(Z)$ value is very low, it tends to increase steadily throughout the training. Conversely, if the initial value is high, it first decreases and then increases.

Our further experiments revealed that if we initialize the convolutional layers using a Gaussian distribution $\mathcal{N}(0, \sigma)$, then $S_D(Z)$ is affected by the standard deviation σ. The effect varies with different models; for instance, in ResNet50, $S_D(Z)$ increases monotonically as σ (as shown in Figure 9 left panels).

This led us to wonder: if we initialize the model to have a high $S_D(Z)$ or the opposite, would it affect the model's subsequent training performance?

Experimental results indicate that it indeed has an impact. Models initialized with a low $S_D(Z)$ train faster and perform better. We hypothesize that if a model is initialized with high entropy, it might start in a suboptimal high-entropy state. During training, the model first needs to escape this state before moving towards an optimal high-entropy state, causing the $S_D(Z)$ to initially decrease and then increase. Conversely, if the model is initialized with low entropy, it only needs to move towards the optimal high-entropy state, avoiding this detour. Of course, this is just a hypothesis and verifying it in detail might be challenging.

6.2 Application 2: Using DSMI to predict model performance

Next, we considered whether DSE and DSMI could serve as general metrics across different models, rather than just studying their changes within the same model. Could they potentially be used for cross-model comparisons?

To investigate this, we downloaded 963 pretrained models on ImageNet from the torch image model (timm) [18] library and examined whether DSE and DSMI correlate with these models' performance on downstream tasks.

Specifically, to avoid excessive computational demands, we used a typical ImageNet subset (Imagenette + Imagewoof) to calculate DSE and DSMI.

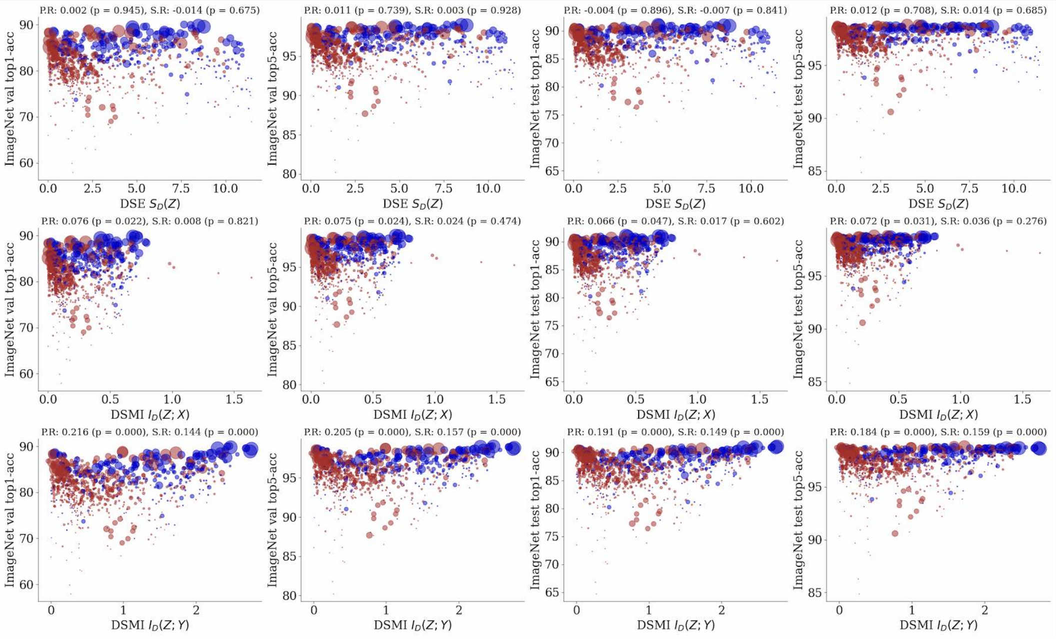

Figure 10: Correlation analysis between DSE $S_D(Z)$, DSMI $I_D(Z;X)$, $I_D(Z;Y)$ and ImageNet accuracy evaluated on 962 pre-trained models. Red circles are ConvNets and blue circles are ViTs. Circle sizes indicate model sizes. $I_D(Z;Y)$ shows a strong positive correlation ($p < 0.001$). P.R: Pearson correlation coefficient, S.R: Spearman correlation coefficient.

The results, shown in Figure 10, were quite interesting: $I_D(Z;Y)$ indeed showed a strong correlation with downstream task performance. This suggests that, at least theoretically, we could use $I_D(Z;Y)$ to select models. For example, if there are 20,000 available models, we could run $I_D(Z;Y)$ on a typical subset of your dataset for each model to roughly filter out a few excellent models. Then, you can run comprehensive tests on these shortlisted models to select the best one. This approach helps avoid the computationally intensive process of running full tests on all available models.

7 Core Code

We intended to open source the code (including scripts for models, experiments, training, and evaluation), but unfortunately due to company policies , we have to remove the codebase from public access (sorry folks, and goodbye GitHub stars). However, for the benefit of future researchers, we decide to keep the DSE/DSMI functions public.

The core code consists of 4 files: `dse.py`, `dsmi.py`, `diffusion.py` and `information_utils.py`, which can be found in `https://github.com/ChenLiu-1996/DiffusionSpectralEntropy/`.

8 Citations

Please do not cite this blog as this blog is not a formal research article. You are very welcome to cite the following papers (the “DSE2024” version preferred) instead if you find this blog helpful.

```
@inproceedings{DSE2024,
  title={Assessing Neural Network Representations During Training Using Noise-
      Resilient Diffusion Spectral Entropy},
  author={Liao, Danqi and Liu, Chen and Christensen, Benjamin W and Tong,
      Alexander and Huguet, Guillaume and Wolf, Guy and Nickel, Maximilian and
      Adelstein, Ian and Krishnaswamy, Smita},
  booktitle={2024 58th Annual Conference on Information Sciences and Systems (
      CISS)},
  pages={1--6},
  year={2024},
  organization={IEEE}
}

@inproceedings{DSE2023,
  title={Assessing Neural Network Representations During Training Using Data
      Diffusion Spectra},
  author={Liao, Danqi and Liu, Chen and Christensen, Benjamin W and Tong,
      Alexander and Huguet, Guillaume and Wolf, Guy and Nickel, Maximilian and
      Adelstein, Ian and Krishnaswamy, Smita},
  booktitle={International Conference on Machine Learning (ICML) Workshop on
      Topology, Algebra, and Geometry in Machine Learning},
  year={2023},
}

@article{DSE2024_arxiv,
  title={Assessing Neural Network Representations During Training Using Noise-
      Resilient Diffusion Spectral Entropy},
  author={Liao, Danqi and Liu, Chen and Christensen, Benjamin W and Tong,
      Alexander and Huguet, Guillaume and Wolf, Guy and Nickel, Maximilian and
      Adelstein, Ian and Krishnaswamy, Smita},
  journal={arXiv preprint arXiv:2312.04823},
  year={2023}
}
```

References

[1] Danqi Liao, Chen Liu, Benjamin W Christensen, Alexander Tong, Guillaume Huguet, Guy Wolf, Maximilian Nickel, Ian Adelstein, and Smita Krishnaswamy. Assessing neural network representations during training using noise-resilient diffusion spectral entropy. In *2024 58th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2024.

[2] Danqi Liao, Chen Liu, Benjamin W Christensen, Alexander Tong, Guillaume Huguet, Guy Wolf, Maximilian Nickel, Ian Adelstein, and Smita Krishnaswamy. Assessing neural network representations during training using noise-resilient diffusion spectral entropy. *arXiv preprint arXiv:2312.04823*, 2023.

[3] Danqi Liao, Chen Liu, Benjamin W Christensen, Alexander Tong, Guillaume Huguet, Guy Wolf, Maximilian Nickel, Ian Adelstein, and Smita Krishnaswamy. Assessing neural network representations during training using data diffusion spectra. In *International Conference on Machine Learning (ICML) Workshop on Topology, Algebra, and Geometry in Machine Learning*, 2023.

[4] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[5] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2020.

[6] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[7] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018.

[8] Dave Gershgorn. New theory cracks open the black box of deep learning, 2017.

[9] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[10] Manik Kuchroo, Marcello DiStasio, Eric Song, Eda Calapkulu, Le Zhang, Maryam Ige, Amar H Sheth, Abdelilah Majdoubi, Madhvi Menon, Alexander Tong, et al. Single-cell analysis reveals inflammatory interactions driving macular degeneration. *Nature Communications*, 14(1):2589, 2023.

[11] Chen Liu, Matthew Amodio, Liangbo L Shen, Feng Gao, Arman Avesta, Sanjay Aneja, Jay Wang, Lucian V Del Priore, and Smita Krishnaswamy. Cuts: A deep learning and topological framework for multigranular unsupervised medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2024.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[14] Chen Liu, Ke Xu, Liangbo L Shen, Guillaume Huguet, Zilong Wang, Alexander Tong, Danilo Bzdok, Jay Stewart, Jay C Wang, Lucian V Del Priore, et al. Imageflownet: Forecasting multiscale trajectories of disease progression with irregularly-sampled longitudinal medical images. *arXiv preprint arXiv:2406.14794*, 2024.

[15] Xingzhi Sun, Danqi Liao, Kincaid MacDonald, Yanlei Zhang, Guillaume Huguet, Guy Wolf, Ian Adelstein, Tim GJ Rudner, and Smita Krishnaswamy. Geometry-aware generative autoencoders for metric learning and generative modeling on data manifolds. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024.

[16] Greg Ver Steeg. Non-parametric entropy estimation toolbox (npeet). *Non-parametric entropy estimation toolbox (NPEET)*, 2000.

[17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[18] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.