entropy

Article

Analysis of Deep Convolutional Neural Networks Using Tensor Kernels and Matrix-Based Entropy

Kristoffer K. Wickstrøm [1,*], Sigurd Løkse [1], Michael C. Kampffmeyer [1,2], Shujian Yu [1,3,4], José C. Príncipe [3] and Robert Jenssen [1,2,5]

1 Machine Learning Group, Department of Physics and Technology, UiT The Arctic University of Norway, NO-9037 Tromsø, Norway; sigurd.lokse@uit.no (S.L.); michael.c.kampffmeyer@uit.no (M.C.K.); s.yu3@vu.nl (S.Y.); robert.jenssen@uit.no (R.J.)
2 Norwegian Computing Center, Department of Statistical Analysis and Machine Learning, 114 Blindern, NO-0314 Oslo, Norway
3 Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA; principe@cnel.ufl.edu
4 Department of Computer Science, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands
5 Department of Computer Science, University of Copenhagen, Universitetsparken 1, 2100 Copenhagen, Denmark
* Correspondence: kristoffer.k.wickstrom@uit.no

Abstract: Analyzing deep neural networks (DNNs) via information plane (IP) theory has gained tremendous attention recently to gain insight into, among others, DNNs' generalization ability. However, it is by no means obvious how to estimate the mutual information (MI) between each hidden layer and the input/desired output to construct the IP. For instance, hidden layers with many neurons require MI estimators with robustness toward the high dimensionality associated with such layers. MI estimators should also be able to handle convolutional layers while at the same time being computationally tractable to scale to large networks. Existing IP methods have not been able to study truly deep convolutional neural networks (CNNs). We propose an IP analysis using the new matrix-based Rényi's entropy coupled with tensor kernels, leveraging the power of kernel methods to represent properties of the probability distribution independently of the dimensionality of the data. Our results shed new light on previous studies concerning small-scale DNNs using a completely new approach. We provide a comprehensive IP analysis of large-scale CNNs, investigating the different training phases and providing new insights into the training dynamics of large-scale neural networks.

Keywords: information theory; deep learning; information plane; kernels methods

check for updates

Citation: Wickstrøm, K.K.; Løkse, S.; Kampffmeyer, M.C.; Yu, S.; Príncipe, J.C.; Jenssen, R. Analysis of Deep Convolutional Neural Networks Using Tensor Kernels and Matrix-Based Entropy. *Entropy* **2023**, *25*, 899. https://doi.org/10.3390/e25060899

Academic Editor: Jerry D. Gibson

Received: 25 April 2023
Revised: 31 May 2023
Accepted: 2 June 2023
Published: 3 June 2023

1. Introduction

Although deep neural networks (DNNs) are at the core of most state-of-the art systems in computer vision, the theoretical understanding of such networks is still not at a satisfactory level [1]. In order to provide insight into the inner workings of DNNs, the prospect of utilizing the mutual information (MI), a measure of dependency between two random variables, has recently garnered a significant amount of attention [1–8]. Given the input variable X and the desired output Y for a supervised learning task, a DNN is viewed as a transformation of X into a representation that is favorable for obtaining a good prediction of Y. By treating the output of each hidden layer as a random variable T, one can model the MI $I(X;T)$ between X and T. Likewise, the MI $I(T;Y)$ between T and Y can be modeled. The quantities $I(X;T)$ and $I(T;Y)$ span what is referred to as the information plane (IP). Several works have unveiled interesting properties of the training dynamics through IP analysis of DNNs [4,7,9–11]. Figure 1, produced using our proposed measure, illustrates one such insight that is similar to the observations of [1], where training can be separated into two distinct phases: the fitting phase and the compression phase.

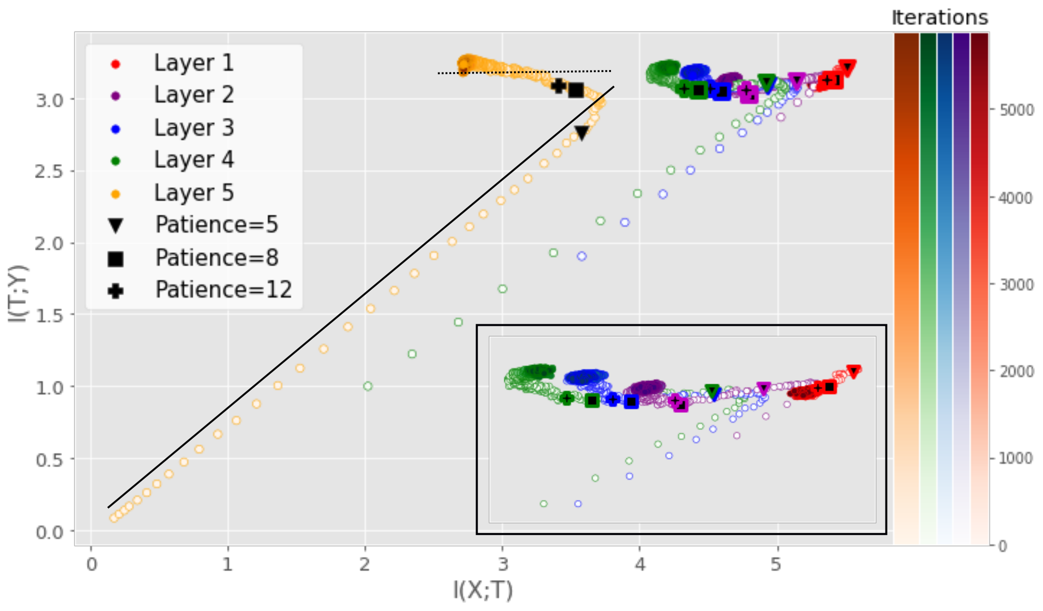

Figure 1. IP obtained using our proposed measure for a small DNN averaged over 5 training runs. The solid black line illustrates the fitting phase while the dotted black line illustrates the compression phase. The iterations at which early stopping would be performed assuming a given patience parameter are highlighted. Patience denotes the number of iterations that need to pass without progress on a validation set before training is stopped to avoid overfitting. For low patience values, training will stop before the compression phase. For the benefit of the reader, a magnified version of the first four layers is also displayed.

The claim of a fitting and compression phase has been highly debated as subsequent research has linked the compression phase to the saturation of neurons [5] or clustering of the hidden representations [9]. Recent studies [1,5,7,9] have been focused on small-scale networks or non-convolutional networks, as the current MI estimators cannot tackle the tensor representations produced by convolutional layers or do not scale well to convolutional layers with many filters [6]. This severely limits the scope of IP analysis, as most real-world applications rely on large-scale CNNs. In this work, we continue the IP line of research through a new matrix-based entropy functional [6,7,12]. We provide insights on this functional by linking the functional to well-understood measures from the kernel literature and propose a new kernel tensor-based approach to the matrix-based entropy functional. Furthermore, we give a new formulation of the matrix-based entropy that is closely connected to measures from quantum information theory. Using the proposed estimator, we provide an analysis of large-scale DNNs and give a new information theoretic understanding of the training procedure of DNNs. Using the proposed measure, we investigate the claim of [3] that the entropy $H(X) \approx I(T;X)$ and $H(Y) \approx I(T;Y)$ in high dimensions (in which case MI-based analysis would be meaningless). The contributions of this work can be summarized as:

- We propose a kernel tensor-based approach to the matrix-based entropy functional that is designed for measuring MI in large-scale convolutional neural networks (CNNs).
- We provide new insights on the matrix-based entropy functional by showing its connection to well-known quantities in the kernel literature such as the kernel mean embedding and maximum mean discrepancy. Furthermore, we show that the matrix-based entropy functional is closely linked with the von Neuman entropy from quantum information theory.
- Our results indicate that the compression phase is apparent mostly for the training data and less so for the test data, particularly for more challenging datasets. When using a technique such as early stopping to avoid overfitting, training tends to stop before the compression phase occurs (see Figure 1).

2. Related Work

Analyzing DNNs in the IP was first proposed by [13] and later demonstrated by [1]. Among other results, the authors studied the evolution of the IP during the training process of DNNs and noted that the process was composed of two different phases. First, there is an initial fitting phase where $I(T;Y)$ increases, which is followed by a phase of compression where $I(X;T)$ decreases. These results were later questioned by [5], who argued that the compression phase is not a general property of the DNN training process but rather an effect of different activation functions. However, a recent study by [4] seems to support the claim of a compression phase regardless of the activation function. The authors argue that the base estimator of MI utilized by [5] might not be accurate enough and demonstrate that a compression phase does occur, but the amount of compression can vary between different activation functions. Another recent study by [10] also reported a compression phase but highlighted the importance of adaptive MI estimators. They also showed that when L2 regularization was included in the training, compression was observed regardless of the activation function. In addition, some recent studies have discussed the limitations of the IP framework for analysis and optimization for particular types of DNN [14,15]. Furthermore, ref. [16] investigated similarities between hidden layers and between hidden layers of different networks, but they did so only for the representation obtained after the networks were fully trained. The dynamics of large-scale DNNs was investigated [17] using MINE [18]. A number of information plane-related studies have also been discussed in [19].

On a different note, Ref. [3] proposed an evaluation framework for DNNs based on the IP and demonstrated that MI can be used to infer the capability of DNNs to recognize objects for an image classification task. Furthermore, the authors argue that when the number of neurons in a hidden layer grows large, $I(T;X)$ and $I(Y;T)$ barely change and are, using [3] terminology, approximately deterministic, i.e., $I(T;X) \approx H(X)$ and $I(T;Y) \approx H(Y)$. Therefore, they only model the MI between X and the last hidden layer—that is, the output of the network—and the last hidden layer and Y.

Ref. [7] investigated a matrix-based measure of MI for analyzing different data processing inequalities in feed-forward stacked autoencoders (SAEs), concluding that the compression phase in the IP of SAEs is determined by the values of the SAE bottleneck layer size and the intrinsic dimensionality of the given data. In follow-up work, a simplistic analysis of small convolutional neural networks (CNNs) was provided in [6]; however, it was based on a multivariate extension [20] of matrix-based Renyi entropy that does not scale well numerically or computationally in the number of feature maps. The interested reader can find additional information on the multivariate extension by [20] in Appendix A.

Recently, the information plane of quantized neural networks was modeled [8], which allowed for an exact analysis of its dynamics. In addition, log-determinant entropy has been used for information plane analysis [11]. Lastly, information plane analysis has also been used to improve the understanding of graph convolutional neural networks [21].

In this paper, we continue the recent trend of leveraging the new matrix-based measures of entropy. We contribute both new insight to the definition of these measures (see Section 3.2.1), and importantly, we extend matrix-based measures of entropy to exploit tensor kernels to enable the first IP analysis of large-scale CNNs by treating feature maps as tensors (Section 3.3).

3. Materials and Methods

3.1. Preliminaries on Matrix-Based Information Measures

For the benefit of the reader, we review in this section first the theory underlying the recent matrix-based measures of entropy and mutual information. Thereafter, we contribute a new special case interpretation of the definition of matrix-based Renyi entropy (Section 3.2.1) and give new insights on the link to other measures in the kernel literature before presenting our new tensor-based approach.

3.1.1. Matrix-Based Entropy and Mutual Information

The matrix-based measure of entropy, originally proposed by [12], is built on kernel matrices obtained from raw data, involving no explicit density estimation or binning procedure:

Definition 1 ([12])**.** *Let $\mathbf{x}_i \in \mathcal{X},\ i = 1, 2, \dots, N$ denote data points and let $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be an infinitely divisible positive definite kernel [22]. Given the kernel matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ with elements $(\mathbf{K})_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and the matrix $\mathbf{A},\ (\mathbf{A})_{ij} = \frac{1}{N}\frac{(\mathbf{K})_{ij}}{\sqrt{(\mathbf{K})_{ii}(\mathbf{K})_{jj}}}$, the matrix-based Rényi's α-order entropy is given by*

$$
\begin{aligned}
S_\alpha(\mathbf{A}) &= \frac{1}{1-\alpha}\log_2\Big(\operatorname{tr}(\mathbf{A}^\alpha)\Big) \\
&= \frac{1}{1-\alpha}\log_2\left[\sum_{i=1}^{N}\lambda_i(\mathbf{A})^\alpha\right].
\end{aligned} \tag{1}
$$

Here, $\lambda_i(\mathbf{A})$ denotes the ith eigenvalue of the matrix $\mathbf{A}$. Equation (1) is a measure of an entropy-like quantity that satisfies Renyi's axiomatic characterization of entropy [23], which is referred to as matrix-based Renyi entropy. In addition to the matrix-based entropy, ref. [12] also defined the matrix-based joint entropy between $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ as

$$
S_\alpha(\mathbf{A}_\mathcal{X}, \mathbf{A}_\mathcal{Y}) = S_\alpha\left(\frac{\mathbf{A}_\mathcal{X} \circ \mathbf{A}_\mathcal{Y}}{\operatorname{tr}(\mathbf{A}_\mathcal{X} \circ \mathbf{A}_\mathcal{Y})}\right), \tag{2}
$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ are two different representations of the same object and $\circ$ denotes the Hadamard product. Finally, the MI is, similar to Shannon's formulation, defined as

$$
I_\alpha(\mathbf{A}_\mathcal{X}; \mathbf{A}_\mathcal{Y}) = S_\alpha(\mathbf{A}_\mathcal{X}) + S_\alpha(\mathbf{A}_\mathcal{Y}) - S_\alpha(\mathbf{A}_\mathcal{X}, \mathbf{A}_\mathcal{Y}). \tag{3}
$$

The properties of these quantities were analyzed in detail in [12], but we want to highlight some important properties and provide links with other measures in the kernel literature.

Information theoretic measures are developed in such a way that they satisfy certain axioms. The measures presented above satisfy the axioms put forth by [23] and are closely connected with quantum information theory [24]. In quantum statistical mechanics, the Von Neumann's entropy [24] is defined as

$$
S(\rho) = -\operatorname{tr}(\rho \log \rho), \tag{4}
$$

where ρ is a density matrix that described a quantum mechanical system. If ρ is written in terms of its eigenvalues, $\lambda_i(\rho)$, then Equation (4) can also be formulated as

$$
S(\rho) = -\sum_{i=1}^{N}\lambda_i(\rho)\log[\lambda_i(\rho)]. \tag{5}
$$

In addition, the quantum extensions of Renyi's entropy [25] that is defined as

$$
S_\alpha(\rho) = \frac{1}{1-\alpha}\log[\operatorname{tr}(\rho^\alpha)], \tag{6}
$$

bears a close resemblance to the matrix-based definition of entropy in Equation (1). While some properties of Equations (4) and (6) can also be extended to Equation (1), it is important to note that the two approaches are very different since the matrix-based framework is built around kernel matrices obtained directly from raw data.

3.1.2. Bound on Matrix-Based Entropy Measure

Not all measures of entropy have the same properties. Many of the estimators used and developed for Shannon suffer from the curse of dimensionality [26]. In contrast, Renyi's entropy measures have the same functional form of the statistical quantity in a reproducing kernel Hilbert space (RKHS), thus capturing properties of the data population. Essentially, we are projecting marginal distribution to an RKHS in order to measure entropy and MI. This is similar to the approach of maximum mean discrepancy and the kernel mean embedding [27,28]. The connection with the data population can be shown via the theory of covariance operators. The covariance operator $G : \mathcal{H} \to \mathcal{H}$ is defined through the bilinear form

$$\begin{aligned}\mathcal{G}(f,g) = \langle f, Gg\rangle &= \int_{\mathcal{X}} \langle f, \psi(\mathbf{x})\rangle\langle\psi(\mathbf{x}), g\rangle d\mathbb{P}_{\mathcal{X}}(\mathbf{x}) \\ &= E_{\mathcal{X}}\{f(X), g(Y)\}\end{aligned} \tag{7}$$

where $\mathbb{P}_{\mathcal{X}}$ is a probability measure and $f, g \in \mathcal{H}$. Based on the empirical distribution $\mathbb{P}_N = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_i}(\mathbf{x})$, the empirical version $\hat{G}$ of G obtained from a sample $\mathbf{x}_i$ of size N is given by:

$$\begin{aligned}\langle f, \hat{G}_{Ng}\rangle = \hat{\mathcal{G}}(f,g) &= \int_{\mathcal{X}} \langle f, \psi(\mathbf{x})\rangle\langle\psi(\mathbf{x}), g\rangle d\mathbb{P}_{\mathcal{X}}(\mathbf{x}) \\ &= \frac{1}{N}\sum_{i=1}^{N}\langle f, \psi(\mathbf{x}_i)\rangle\langle\psi(\mathbf{x}_i), g\rangle\end{aligned} \tag{8}$$

By analyzing the spectrum of $\hat{G}$ and G, Ref. [12] showed that the difference between $\mathrm{tr}(G)$ and $\mathrm{tr}(\hat{G})$ can be bounded, as stated in the following proposition:

Proposition 2. *Let $\mathbb{P}_N = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_i}(\mathbf{x})$ be the empirical distribution. Then, as a consequence of Proposition 6.1 in [12], $tr[\hat{G}_N^{\alpha}] = tr\left[\left(\frac{1}{N}\mathbf{K}\right)^{\alpha}\right]$. The difference between $\mathrm{tr}(G)$ and $\mathrm{tr}(\hat{G})$ can be bounded under the conditions of Theorem 6.2 in [12] and for $\alpha > 1$, with probability 1-δ*

$$\left|\mathrm{tr}(G^{\alpha}) - \mathrm{tr}(\hat{G}_N^{\alpha})\right| \leq \alpha C\sqrt{\frac{2\log\frac{2}{\delta}}{N}} \tag{9}$$

where C is a compact self-adjoint operator.

3.2. Analysis of Matrix-Based Information Measures

In this section, we present new theoretical insights into the Matrix-Based Information Measures.

3.2.1. A New Special-Case Interpretation of the Matrix-Based Renyi Entropy Definition

In previous works, the α in Equation (1) has been ad hoc set to a value of 1.01 in order to approximate Shannon's entropy [6,7]. For $\alpha = 1$, both the denominator and the numerator become zero, so Equation (1) cannot be used directly in this case. However, as a contribution to the matrix-based Renyi entropy theory, we show here that for the case $\alpha \to 1$, Equation (1) can be expressed similarly to the matrix-based Von Neumann's entropy [24], resembling Shannon's definition over probability states and expressed as

$$\lim_{\alpha\to 1} S_{\alpha}(\mathbf{A}) = -\sum_{i=1}^{N}\lambda_i(\mathbf{A})\log_2[\lambda_i(\mathbf{A})]. \tag{10}$$

Equation (1) can be proved using L'Hôpital's rule as follows:

Proof.

$$\lim_{\alpha\to 1} S_{\alpha}(\mathbf{A}) = \lim_{\alpha\to 1}\frac{1}{1-\alpha}\log_2\left(\sum_{i=1}^{n}\lambda_i^{\alpha}\right) \to \frac{0}{0}, \tag{11}$$

since $\sum_{i=1}^{N} \lambda_i = \text{tr}(\mathbf{A}) = 1$. L'Hôpital's rule yields

$$\begin{aligned} \lim_{\alpha \to 1} S_\alpha(\mathbf{A}) &= \lim_{\alpha \to 1} \frac{\frac{\partial}{\partial \alpha} \log_2[\sum_{i=1}^{n} \lambda_i(\mathbf{A})^\alpha]}{\frac{\partial}{\partial \alpha}(1-\alpha)} \\ &= -\frac{1}{\ln 2} \lim_{\alpha \to 1} \frac{\sum_{i=1}^{n} \lambda_i(\mathbf{A})^\alpha \ln[\lambda_i(\mathbf{A})]}{|\sum_{i=1}^{n} \lambda_i(\mathbf{A})^\alpha|} \\ &= -\sum_{i=1}^{n} \lambda_i(\mathbf{A}) \log_2[\lambda_i(\mathbf{A})]. \end{aligned} \tag{12}$$

□

3.2.2. Link to Measures in Kernel Literature and Validation on High-Dimensional Synthetic Data

An interesting aspect of the matrix-based measure of entropy is the special case connection with the theory of maximum mean discrepancy and Hilbert–Schmidt norms via covariance operators [29]. Let G be the covariance operator (see [28] for details), then, for the particular case of $\alpha = 2$, the empirical trace of the covariance operator, $\text{tr}(G^2)$, is given by $\text{tr}(\mathbf{A}^2)$. Furthermore,

$$\begin{aligned} \text{tr}(G^2) &= \left\langle \int_{\mathcal{X}} \kappa(\cdot, \mathbf{x}) d\mathbb{P}_{\mathcal{X}}(\mathbf{x}), \int_{\mathcal{X}} \kappa(\cdot, \mathbf{y}) d\mathbb{P}_{\mathcal{X}}(\mathbf{y}), \right\rangle \\ &= ||\mu_{\mathcal{X}}||^2_{\mathcal{K}}, \end{aligned} \tag{13}$$

where $\mu_{\mathcal{X}} = \int_{\mathcal{X}} \kappa(\cdot, \mathbf{x}) d\mathbb{P}_{\mathcal{X}}(\mathbf{x})$ is the *kernel mean map* [30], i.e., an embedding of the probability measure $\mathbb{P}_{\mathcal{X}}(\mathbf{x})$ in a reproducing kernel Hilbert space (RKHS). Thus, the matrix $\mathbf{A}$ can be related to an empirical covariance operator on embeddings of probability distributions in an RKHS. Moreover, ref. [12] showed that under certain conditions, Equation (1) converges to the trace of the underlying covariance operator, as shown in Proposition 2 in Section 3.1.2. Notice that the dimensionality of the data does not appear in Proposition 2. This means that $S_\alpha(\mathbf{A})$ captures properties of the distribution with a certain robustness with respect to high-dimensional data. This is a beneficial property compared to KNN and KDE-based information measures used in previous works [5,10], which have difficulties handling high-dimensional data [26]. Some measures of entropy developed for measuring the Shannon entropy suffer from the curse of dimensionality [26]. In addition, there is no need for any binning procedure utilized in previous works [1], which are known to struggle with the ReLU activation function commonly used in DNNs [5]. While Equation (13) is not explicitly used in the remainder of our manuscript, we believe that these insights provides a deeper understanding of the inner workings of the matrix-based entropy measure.

To examine the behavior of the matrix-based measures described in Section 3.1, we have conducted a simple experiment on measuring entropy and mutual information in high-dimensional data following a normal distribution with known mean and covariance matrix. In the particular case of the normal distribution, the entropy and mutual information can be calculated analytically. The entropy can be calculated as:

$$H(\mathcal{N}_0) = \frac{1}{2} \log\Big((2\pi e)^d \det(\mathbf{\Sigma}_0)\Big), \tag{14}$$

where $\mathcal{N}_0$ denotes a normal distribution with mean vector $\boldsymbol{\mu}_0$ covariance matrix $\mathbf{\Sigma}_0$, and dimensionality d. For mutual information, we use the experimental setup considered in [31]. Let Z have a $d+1$ dimensional Gaussian distribution with covariance matrix $\mathbf{\Sigma}_z$. Next, let $X = (Z_1, \ldots, Z_d)$ and $Y = Z_{d+1}$. Then, their mutual information satisfies:

$$I(X, Y) = I(Z_1, \ldots, Z_{d+1}) - I(Z_1, \ldots, Z_d) \tag{15}$$

$$= -\frac{1}{2} \log\left(\frac{\det(\mathbf{\Sigma}_z)}{\det(\mathbf{\Sigma}_x)}\right), \tag{16}$$

where $\mathbf{\Sigma}_x$ is the covariance matrix of X. We consider five cases of $(d+1)$-dimensional Gaussian distributions with mean zero, unit variance, and an increasing amount of dependence. The unit variance covariance matrices for the five cases are given as follows:

$$\begin{cases} \mathbf{\Sigma}_A(i,j) = 0.1, & \text{for } i \neq j, \\ \mathbf{\Sigma}_B(i,j) = 0.25, & \text{for } i \neq j, \\ \mathbf{\Sigma}_C(i,j) = 0.5, & \text{for } i \neq j, \\ \mathbf{\Sigma}_D(i,j) = 0.75, & \text{for } i \neq j, \\ \mathbf{\Sigma}_E(i,j) = 0.9, & \text{for } i \neq j. \end{cases}$$

The left plot in Figure 2 displays the entropy of a 100-dimensional normal distribution with zero mean and isotropic covariance matrix, which is calculated using Equation (14). The right plot in Figure 2 displays the estimated entropy using Equation (1) on 500 randomly drawn samples from $\mathcal{N}_0$, which are computed on all 500 samples and in batches of 100. The results show how the estimated entropy follows the same trend as the analytically computed entropy. We quantitativly evaluate the correlation between the analytic quantity and the estimated quantity by calculating Pearson's correlation coefficient and find that that both the full data and batch-wise estimation are highly correlated with the analytic calculation (correlation ≈ 0.99, p-value ≤ 0.01). For mutual information, we generate 500 samples from a 100 + 1 dimensional Gaussian distribution and compare the exact and estimated mutual information, which are both based on all 500 samples and in the batch-wise setting. The left plot of Figure 3 shows the mutual information between X and Y calculated using Equation (16) for the five cases described above. The right part of Figure 3 shows the estimated mutual information using Equation (3), which is computed on all 500 samples and in batches of 100. Again, the result shows how the estimated values follow the same trend as the exact mutual information values. Similarly as with the entropy, we quantitativly evaluate the correlation between the analytic quantity and the estimated quantity by calculating Pearson's correlation coefficient and find that that both the full data and batch-wise estimation are highly correlated with the analytic calculation (correlation ≈ 0.99, p-value ≤ 0.01). Note that the exact value of both the entropy and mutual information is different between the exact and estimated quantities. This is expected, as the matrix-based entropy estimators measure information theoretic quantities in RKHS without the need for explicit PDF but with similar properties as common information theoretic measures. The kernel width was selected by taking the median distance between all samples.

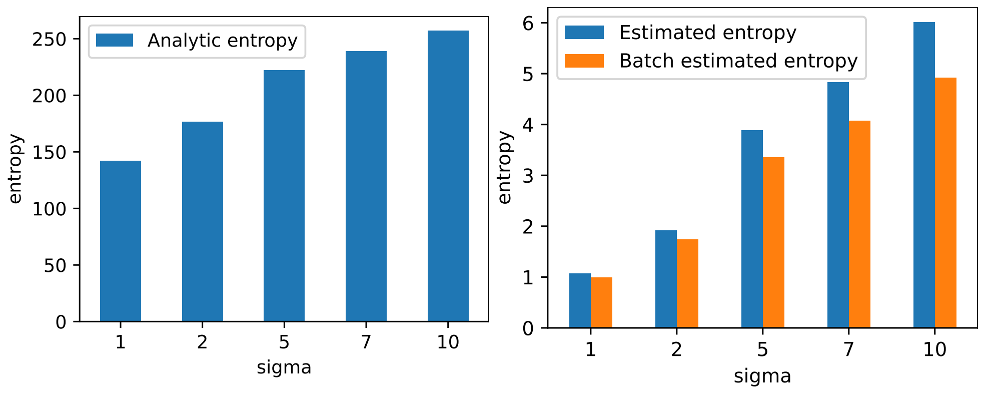

Figure 2. The leftmost plot shows the entropy calculated using Equation (14) of a 100-dimensional normal distribution with zero mean and an isotropic covariance matrix for different variances. The variances are given along the x-axis. The rightmost plot shows the entropy estimated using Equation (1) for the same distribution. The plots illustrated that the analytically computed entropy and the estimated quantity follow the same trend.

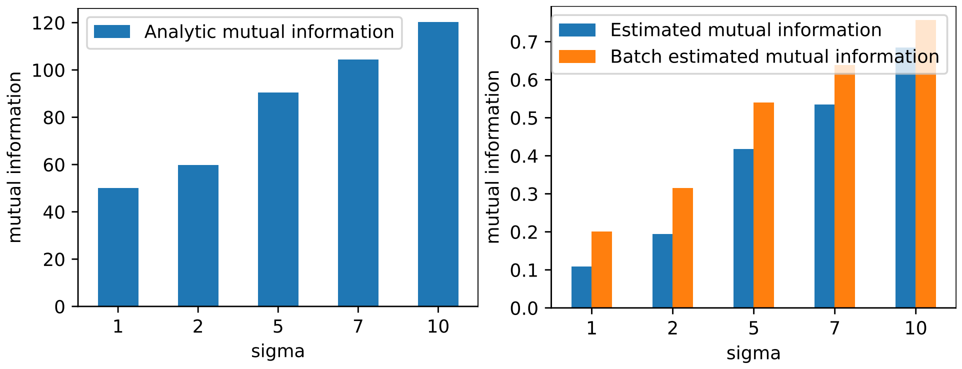

Figure 3. The leftmost plot shows the mutual information calculated using Equation (16) between a standard 100-dimensional normal distribution and a normal distribution with a mean vector of all ones and an isotropic covariance matrix with different variances. The variances are given along the x-axis. The rightmost plot shows the mutual information estimated using Equation (3) for the same distributions. The plots illustrated that the analytically computed mutual information and the estimated quantity follow the same trend.

3.3. Novel Tensor-Based Matrix-Based Renyi Information Measures

To invoke information theoretic quantities of features produced by convolutional layers and to address the limitations discussed above, we introduce in this section our novel tensor-based approach for measuring entropy and MI in DNNs. This enables for the first time an IP analysis of large-scale CNNs.

3.4. Tensor Kernels for Measuring Mutual Information

The output of a convolutional layer is represented as a tensor $\mathbb{X}_i \in \mathbb{R}^C \otimes \mathbb{R}^H \otimes \mathbb{R}^W$ for a data point i. As discussed above, the matrix-based Rényi's α-entropy cannot include tensor data without modifications. To handle the tensor-based nature of convolutional layers, we propose to utilize tensor kernels [32] to produce a kernel matrix for the output of a convolutional layer. A tensor formulation of the radial basis function (RBF) kernel can be stated as

$$\kappa_{\text{ten}}(\mathbb{X}_i, \mathbb{X}_j) = e^{-\frac{1}{\sigma^2}\|\mathbb{X}_i - \mathbb{X}_j\|_F^2}, \tag{17}$$

where $\|\cdot\|_F$ denotes the Hilbert–Frobenius norm [32] and σ is the kernel width parameter. In practice, the tensor kernel in Equation (17) can be computed by reshaping the tensor into a vectorized representation while replacing the Hilbert–Frobenius norm with a Euclidean norm. We compute the MI in Equation (3) by replacing the matrix $\mathbf{A}$ with

$$\begin{aligned}(\mathbf{A}_{\text{ten}})_{ij} &= \frac{1}{N}\frac{(\mathbf{K}_{\text{ten}})_{ij}}{\sqrt{(\mathbf{K}_{\text{ten}})_{ii}(\mathbf{K}_{\text{ten}})_{jj}}} \\ &= \frac{1}{N}\kappa_{\text{ten}}(\mathbb{X}_i, \mathbb{X}_j).\end{aligned} \tag{18}$$

While Equation (17) provides the simplest and most intuitive approach for using kernels with tensor data, it does have its limitations. Namely, a tensor kernel that simply vectorizes the tensor ignores the inter-component structures within and between the respective tensor [32]. For simple tensor data, such structures might not be present and a tensor kernel as described above can suffice; however, other tensor kernels do exist, such as for instance the matricization-based tensor kernels [32]. In this work, we have chosen the tensor kernel defined in Equation (17) for its simplicity and computational benefits, which come from the fact that the entropy and joint entropy are computed batch-wise by finding the eigenvalues of a kernel matrix, or the eigenvalues of the Hadamard product of two kernel matrices, and utilizing Equation (1). Nevertheless, exploring structure-preserving kernels can be an interesting research path in future works. In Appendix C, we have included a simple example toward this direction, where the tensor kernel described in this paper is compared to a matricization-based tensor kernel.

3.4.1. Choosing the Kernel Width

With methods involving RBF kernels, the choice of the kernel width parameter, σ, is always critical. For supervised learning, one might choose this parameter by cross-validation based on validation accuracy, while in unsupervised problems, one might use a rule of thumb [33–35]. However, in the case of measuring MI in DNNs, the data are often high dimensional, in which case unsupervised rules of thumb often fail [34].

In this work, we choose σ based on an optimality criterion. Intuitively, one can make the following observation: a good kernel matrix should reveal the class structures present in the data. This can be accomplished by maximizing the so-called *kernel alignment* loss [36] between the kernel matrix of a given layer, $\mathbf{K}_\sigma$, and the label kernel matrix, $\mathbf{K}_y$. The kernel alignment loss is defined as

$$A(\mathbf{K}_a, \mathbf{K}_b) = \frac{\langle \mathbf{K}_a, \mathbf{K}_b \rangle_F}{\|\mathbf{K}_a\|_F \|\mathbf{K}_b\|_F}, \tag{19}$$

where $\| \cdot \|_F$ and $\langle \cdot, \cdot \rangle_F$ denote the Frobenius norm and inner product, respectively. Thus, we choose our optimal σ as

$$\sigma^* = \arg\max_\sigma A(\mathbf{K}_\sigma, \mathbf{K}_y).$$

To stabilize the σ values across mini batches, we employ an exponential moving average, such that in layer ℓ at iteration t, we have

$$\sigma_{\ell,t} = \beta \sigma_{\ell,t-1} + (1-\beta)\sigma^*_{\ell,t},$$

where $\beta \in [0,1]$ and $\sigma_{\ell,1} = \sigma^*_{\ell,1}$.

4. Results

We evaluate our approach by comparing it to previous results obtained on small networks by considering the MNIST dataset and a Multilayer Perceptron (MLP) architecture that was inspired by [5]. We further compare to a small CNN architecture similar to that of [4] before considering large networks, namely VGG16, and a more challenging dataset, namely CIFAR-10. Note that unless stated otherwise, we use CNN to denote the small CNN architecture. Details about the MLP and the CNN utilized in these experiments can be found in Appendix D. All MI measures were computed using Equations (2), (3) and (10) and the tensor approach described in Section 4, which amounts to setting $\alpha = 1$. Furthermore, the MI estimates showed in all plots are averages across multiple training runs. Code is available online (https://github.com/Wickstrom/InformationTheoryExperiment, accessed on 1 June 2023).

Since the MI is computed at the mini-batch level, a certain degree of noise is present. To smooth the MI measures, we employ a moving average approach where each sample is averaged over k mini-batches. For the MLP and CNN experiments, we use $k = 10$, and for the VGG16, we use $k = 50$. We use a batch size of 100 samples and determine the kernel width using the kernel alignment loss defined in Equation (19). For each hidden layer, we chose the kernel width that maximizes the kernel alignment loss in the range 0.1 and 10 times the mean distance between the samples in one mini-batch. Initially, we sample 75 equally spaced values for the kernel width in the given range for the MLP and CNN and 300 values for the VGG16 network. During training, we dynamically reduce the number of samples to 50 and 100, respectively, to reduce computational complexity, which is motivated by the fact that the kernel width remains relatively stable during the latter part of training (illustrated in Section 5). We chose the ranges 0.1 and 10 times the mean distance between the samples in one mini-batch to avoid the kernel width becoming too small and to ensure that we cover a wide enough range of possible values. For the input kernel width, we empirically evaluated values in the range 2–16 and found consistent results for values in the range 4–12. All our experiments were conducted with an input kernel width of 8. For the label kernel matrix, we want a kernel width that is as small as possible to approach an ideal kernel matrix while at the same time large enough to avoid numerical instabilities. For all our experiments, we use a value of 0.1 for the kernel width of the label kernel matrix.

Comparison to previous approaches First, we study the IP of the MLP similar to the one examined in previous works on DNN analysis using information theory [4,5]. We utilize stochastic gradient descent, a cross-entropy loss function, and repeat the experiment 5 times. Figure 1 displays the IP of the MLP with a ReLU activation function in each hidden layer. MI was measured using the training data of the MNIST dataset. A similar experiment was performed with the tanh activation function, obtaining similar results. The interested reader can find these results in Appendix E.

From Figure 1, one can clearly observe a fitting phase, where both $I(T;X)$ and $I(Y;T)$ increase rapidly, followed by a compression phase where $I(T;X)$ decrease and $I(Y;T)$ remains unchanged. In addition, note that $I(Y;T)$ for the output layer (layer 5 in Figure 1) stabilizes at an approximate value of $\log_2(10)$. The following analysis shows that this is to be expected. When the network achieves approximately 100% accuracy, $I(Y;\hat{Y}) \approx S(Y)$, where $\hat{Y}$ denotes the output of the network, since Y and $\hat{Y}$ will be approximately identical and the MI between a variable and itself is just the entropy of the variable. The entropy of Y is measured using Equation (10), which requires the computation of the eigenvalues of the label kernel matrix $\frac{1}{N}\mathbf{K}_y$. For the ideal case, where $(\mathbf{K}_y)_{ij} = 1$ if $y_i = y_j$ and zero otherwise, $\mathbf{K}_y$ is a rank K matrix, where K is the number of classes in the data. Thus, $\frac{1}{N}\mathbf{K}_y$ has K non-zero eigenvalues which are given by $\lambda_k(\frac{1}{N}\mathbf{K}_y) = \frac{1}{N}\lambda_k(\mathbf{K}_y) = \frac{N_{c_k}}{N}$, where N_{c_k} is the number of datapoints in class k, $k = 1, 2, \ldots, K$. Furthermore, if the dataset is balanced, we have $N_{c_1} = N_{c_2} = \ldots = N_{c_K} \equiv N_c$. Then, $\lambda_k\left(\frac{1}{N}\mathbf{K}_y\right) = \frac{N_c}{N} = \frac{1}{K}$, which gives us the entropy measure

$$\begin{aligned} S\left(\frac{1}{N}\mathbf{K}_y\right) &= -\sum_{k=1}^{K} \lambda_k\left(\frac{1}{N}\mathbf{K}_y\right) \log_2\left[\lambda_k\left(\frac{1}{N}\mathbf{K}_y\right)\right] \\ &= -\sum_{k=1}^{K} \frac{1}{K} \log_2\left[\frac{1}{K}\right] \\ &= \log_2[K]. \end{aligned} \tag{20}$$

Next, we examine the IP of a CNN, similar to that studied by [4], with a similar experimental setup as for the MLP experiment. Figure 4 displays the IP of the CNN with a ReLU activation function in all hidden layers. A similar experiment was conducted using the tanh activation function and can be found in Appendix F. While the output layer behaves similarly to that of the MLP, the preceding layers show much less movement. In particular, no fitting phase is observed, which we hypothesize is a result of the convolutional layers being able to extract the necessary information in very few iterations. Note that the output layer is again settling at the expected value of $\log_2(10)$, similar to the MLP, as it also achieves close to 100% accuracy.

Increasing DNN size We analyze the IP of the VGG16 network on the CIFAR-10 dataset with the same experimental setup as in the previous experiments. To our knowledge, this is the first time that the full IP has been modeled for such a large-scale network. Figures 5 and 6 show the IP when measuring the MI for the training dataset and the test dataset, respectively. For the training dataset, we can clearly observe the same trend as for the smaller networks, where layers experience a fitting phase during the early stages of training and a compression phase in the later stage. Note that the compression phase is less prominent for the testing dataset. Note also the difference between the final values of $I(Y;T)$ for the output layer measured using the training and test data, which is a result of the different accuracy achieved on the training data (≈100%) and test data (≈90%). Ref. [3] claims that $I(T;X) \approx H(X)$ and $I(Y;T) \approx H(Y)$ for high-dimensional data, and they highlight particular difficulties with measuring the MI between convolutional layers and the input/output. However, this statement is dependent on their particular measure for the MI, and the results presented in Figure 5 and 6 demonstrate that neither $I(T;X)$ nor $I(Y;T)$ is deterministic for our proposed measure. Furthermore, other measures of MI have also demonstrated that both $I(T;X) \approx H(X)$ and $I(Y;T) \approx H(Y)$ evolve during training [4,18].

Another type of widely used DNNs is residual networks [37]. While these networks typically have less parameters than the VGG16, they usually have more layers. This increase in the number of layers is enabled by skip-connections that allow data to flow through the network without loss of information. This complicates the information theoretic analysis, as the dynamics between the layers change and information do not need to decrease in between the layers. While our proposed estimator is computationally capable of handling residual networks, an extensive analysis would be required to understand the added complexity that is introduced by the lossless flow of information in these networks. We consider such an analysis as outside the scope of this paper but an interesting avenue of future research.

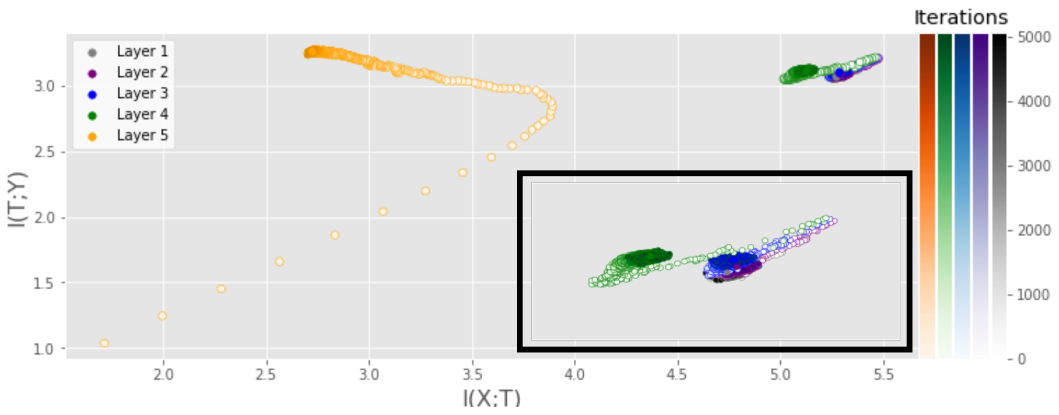

Figure 4. IP of a CNN consisting of three convolutional layers with 4, 8 and 12 filters and one fully connected layer with 256 neurons and a ReLU activation function in each hidden layer. MI was measured using the training data of the MNIST dataset and averaged over 5 runs.

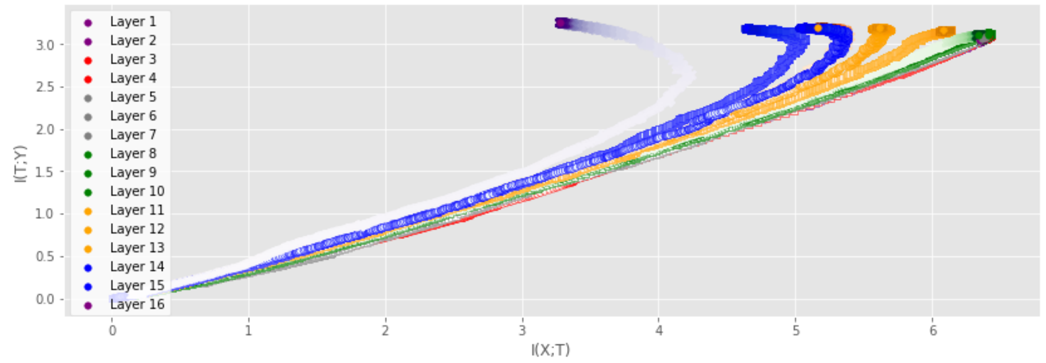

Figure 5. IP of the VGG16 on the CIFAR-10 dataset. MI was measured using the training data and averaged over 2 runs. Color saturation increases as training progresses. Both the fitting phase and the compression phase is clearly visible for several layers.

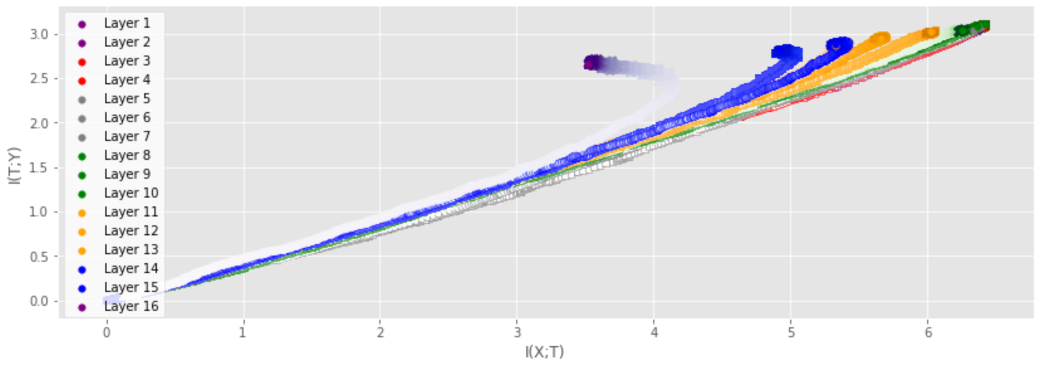

Figure 6. IP of the VGG16 on the CIFAR-10 dataset. MI was measured using the test data and averaged over 2 runs. Color saturation increases as training progresses. The fitting phase is clearly visible, while the compression phase can only be seen in the output layer.

Effect of early stopping We also investigate the effect of using early stopping on the IP described above. Early stopping is a regularization technique where the validation accuracy is monitored and training is stopped if the validation accuracy does not increase for a set number of iterations, which is often referred to as the patience hyperparameter. Figure 1 displays the results of monitoring where the training would stop if the early stopping procedure was applied for different values of patience. For a patience of five iterations, the network training would stop before the compression phase takes place for several of the layers. For larger patience values, the effects of the compression phase can be observed before training is stopped. Early stopping is a procedure intended to prevent the network from overfitting, which may imply that the compression phase observed in the IP of DNNs can be related to overfitting. However, recent research on the so-called double descent phenomenon has shown that longer training might be necessary for good performance for overparameterized DNNs [38,39]. In such settings, early stopping might not be as applicable. We describe the double descent phenomenon and investigate its possible connection with the IP in Appendix H.

Data processing inequality The data processing inequality (DPI) is a concept in information theory which states that the amount of information cannot increase in a chain of transformations. A good information theoretic estimator should tend to uphold the DPI. DNN consists of a chain of mappings from the input through the hidden layers and to the output. One can interpret a DNN as a Markov chain [1,7] that defines an information path [1], which should satisfy the DPI [40]:

$$I(X;T_1) \geq I(X;T_2) \geq \ldots \geq I(X;T_L), \tag{21}$$

where L is the number of layers in the network. An indication of a good MI measure is that it tends to uphold the DPI. Figure 7 illustrates the mean difference in MI between two subsequent layers in the MLP and VGG16 networks. Positive numbers indicate that MI decreases, thus indicating compliance with the DPI. We observe that our measure complies with the DPI for all layers in the MLP and all except one in the VGG16 network. Furthermore, we also model the DPI for a simple MLP using the EDGE MI estimator, which has shown encouraging results on several MI estimation tasks [4]. The DPI for the EDGE estimator is shown in Figure A4 of Appendix G, which shows that the EDGE estimator also upholds the DPI. This agrees with our results with regard to the information flow in neural networks. However, a limitation of the EDGE estimator is that it is not differentiable, which can be beneficial if MI estimates are to be included in the training [41].

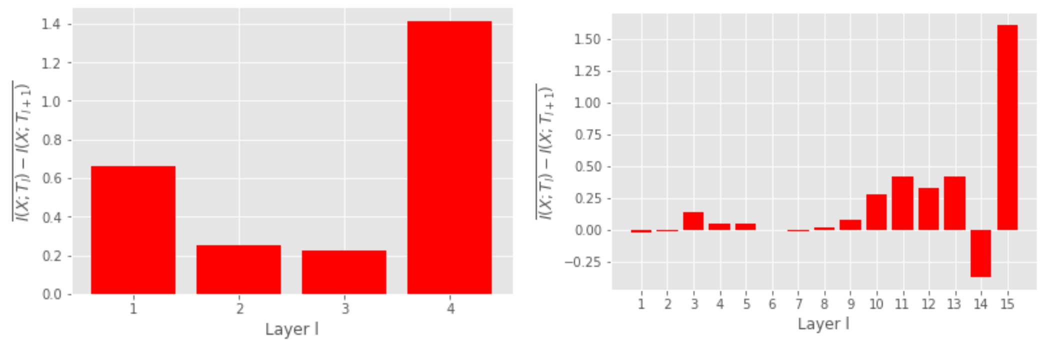

Figure 7. Mean difference in MI of subsequent layers ℓ and $\ell + 1$. Positive numbers indicate compliance with the DPI. MI was measured on the MNIST training set for the MLP and on the CIFAR-10 training set for the VGG16.

5. Kernel Width Sigma

We further evaluate our dynamic approach of finding the kernel width σ. Figure 8 shows the variation of σ in each layer for the MLP, the small CNN and the VGG16 network. We observe that the optimal kernel width for each layer (based on the criterion in Section 3.4.1) stabilizes reasonably quickly and remains relatively constant during training. This illustrates that decreasing the sampling range is a useful approach to decreasing computational complexity.

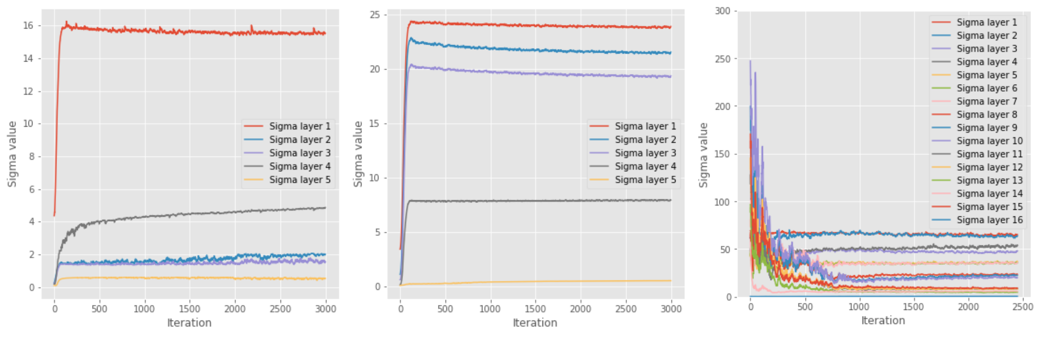

Figure 8. Evolution of kernel width as a function of iteration for the three networks that we considered in this work. From left to right, plots shows the kernel width for the MLP, CNN, and VGG16. The plots demonstrate how the optimal kernel width quickly stabilizes and stays relatively stable throughout the training.

6. Discussion and Conclusions

In this work, we propose a novel framework for analyzing DNNs from an information theoretic perspective using a tensor-based measure of the matrix-based approach of [12]. Our experiments illustrate that the proposed approach scales to large DNNs, which allows us to provide insights into the training dynamics. We observe that the compression phase in neural network training tends to be more prominent when MI is measured on the training set and that commonly used early-stopping criteria tend to stop training before or at the onset of the compression phase. This could imply that the compression phase is linked to overfitting. However, recent research on the double descent phenomenon has shown that a longer training time might be beneficial for generalization [38,39]. In Appendix H, we perform a preliminary study that examines a potential connection between the compression phase and the recent epoch-wise double descent phenomenon. Furthermore, we showed that for our tensor-based approach, the claim that $H(X) \approx I(T;X)$ and $H(Y) \approx I(T;Y)$ does not hold. We believe that our proposed approach can provide new insight and facilitate a more theoretical understanding of DNNs.

Author Contributions: Conceptualization, K.K.W., S.L., M.C.K., S.Y., J.C.P. and R.J.; Formal analysis, K.K.W.; Funding acquisition, R.J.; Investigation, K.K.W., S.L., M.C.K., S.Y., J.C.P. and R.J.; Methodology, K.K.W., S.L., M.C.K., S.Y., J.C.P. and R.J.; Software, K.K.W.; Supervision, M.C.K., J.C.P. and R.J.; Visualization, K.K.W.; Writing—original draft, K.K.W.; Writing—review and editing, K.K.W., S.L., M.C.K., S.Y., J.C.P. and R.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by The Research Council of Norway (RCN) through its Centre for Research-based Innovation funding scheme (grant number 309439) and Consortium Partners; RCN FRIPRO (grant number 315029); RCN IKTPLUSS (grant number 303514) and the UiT Thematic Initiative.

Data Availability Statement: We used the publicly available MNIST (http://yann.lecun.com/exdb/mnist/, accessed on accessed on 1 January 2023) and CIFAR 10 (https://www.cs.toronto.edu/~kriz/cifar.html, accessed on 1 January 2023) datasets in our analysis.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Preliminaries on Multivariate Matrix-Based Renyi's Alpha-Entropy Functionals

The matrix-based Rényi's α-order entropy functional is not suitable for estimating the amount of information of the features produced by a convolutional layer in a DNN as the output consists of C feature maps, each represented by their own matrix, that characterize different properties of the same sample. Ref. [20] proposed a multivariate extension of the matrix-based Rényi's α-order entropy, which computes the joint entropy among C variables as

$$S_\alpha(\mathbf{A}_1, \ldots, \mathbf{A}_C) = S_\alpha\left(\frac{\mathbf{A}_1 \circ \ldots \circ \mathbf{A}_C}{\operatorname{tr}(\mathbf{A}_1 \circ \ldots \circ \mathbf{A}_C)}\right), \tag{A1}$$

where

$$(\mathbf{A}_1)_{ij} = \kappa_1(\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(1)}), \ldots, (\mathbf{A}_C)_{ij} = \kappa_C(\mathbf{x}_i^{(C)}, \mathbf{x}_j^{(C)}). \tag{A2}$$

Ref. [6] also demonstrated how Equation (A1) could be utilized for analyzing the synergy and redundancy of convolutional layers in DNN, but they noted that this formulation can encounter difficulties when the number of feature maps increases, such as in more complex CNNs. Difficulties arise due to the Hadamard products in Equation (A1), given that each element of $\mathbf{A}_c,\ c \in \{1, 2, \ldots, C\}$ takes on a value between 0 and $\frac{1}{N}$, and the product of C such elements thus tends toward 0 as C grows.

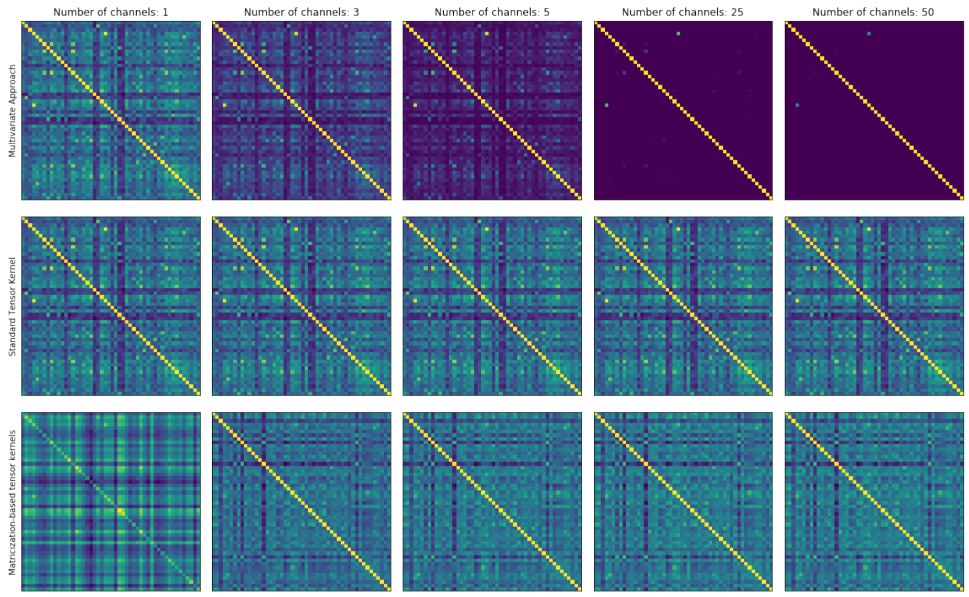

Figure A1. Different approaches for calculating kernels based on tensor data. The first row shows results when using the multivariate approach of [20], the second row depicts the tensor kernel approach used in this paper, and the third row displays the kernel obtained using matricization-based tensor kernels [32] that preserve the structure between channels. Bright colors indicate high values, while dark values indicate low values in all the kernel matrices.

Appendix B. Tensor-Based Approach Contains Multivariate Approach as Special Case

Let $\mathbf{x}_i(\ell) \in \mathbb{R}^{HWC}$ denote the vector representation of data point i in layer ℓ and let $\mathbf{x}_i^{(c)}(\ell) \in \mathbb{R}^{HW}$ denote its representation produced by filter c. In the following, we omit the layer index for ease of notation but assume it is fixed. Consider the case when $\kappa_c(\cdot,\cdot)$ is an RBF kernel with kernel width parameter σ_c. That is, $\kappa_c(\mathbf{x}_i^{(c)}, \mathbf{x}_j^{(c)}) = e^{-\frac{1}{\sigma_c^2}\|\mathbf{x}_i^{(c)} - \mathbf{x}_j^{(c)}\|^2}$. In this case, $\mathbf{A}_c = \frac{1}{N}\mathbf{K}_c$ and

$$\begin{aligned}\frac{\mathbf{A}_1 \circ \ldots \circ \mathbf{A}_C}{\mathrm{tr}(\mathbf{A}_1 \circ \ldots \circ \mathbf{A}_C)} &= \frac{\frac{1}{N}\mathbf{K}_1 \circ \ldots \circ \frac{1}{N}\mathbf{K}_C}{\mathrm{tr}(\frac{1}{N}\mathbf{K}_1 \circ \ldots \circ \frac{1}{N}\mathbf{K}_C)} \\ &= \frac{1}{N}\mathbf{K}_1 \circ \ldots \circ \mathbf{K}_C,\end{aligned} \tag{A3}$$

since $\mathrm{tr}(\mathbf{K}_1 \circ \ldots \circ \mathbf{K}_C) = N$. Thus, element (i,j) is given by

$$\begin{aligned}\left(\frac{\mathbf{A}_1 \circ \ldots \circ \mathbf{A}_C}{\mathrm{tr}(\mathbf{A}_1 \circ \ldots \circ \mathbf{A}_C)}\right)_{ij} &= \frac{1}{N}\prod_{c=1}^{C}(\mathbf{K}_c)_{ij} \\ &= \frac{1}{N}e^{-\sum_{c=1}^{C}\frac{1}{\sigma_c^2}\|\mathbf{x}_i^{(c)} - \mathbf{x}_j^{(c)}\|^2}.\end{aligned} \tag{A4}$$

If we let $\sigma = \sigma_1 = \sigma_2 = \ldots = \sigma_C$, this expression is reduced to

$$\begin{aligned}\frac{1}{N}e^{-\frac{1}{\sigma^2}\sum_{c=1}^{C}\|\mathbf{x}_i^{(c)} - \mathbf{x}_j^{(c)}\|^2} &= \frac{1}{N}e^{-\frac{1}{\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2} \\ &= \frac{1}{N}\kappa_{\mathrm{ten}}(\mathbb{X}_i, \mathbb{X}_j).\end{aligned} \tag{A5}$$

Accordingly, $S_\alpha(\mathbf{A}_{\mathrm{ten}}) = S_\alpha(\mathbf{A}_1, \ldots, \mathbf{A}_C)$, implying that the tensor method is equivalent to the multivariate matrix-based joint entropy when the width parameter is equal within a given layer, assuming an RBF kernel is used. However, the tensor-based approach eliminates the effect of numerical instabilities one encounters in layers with many filters, thereby enabling the training of complex neural networks.

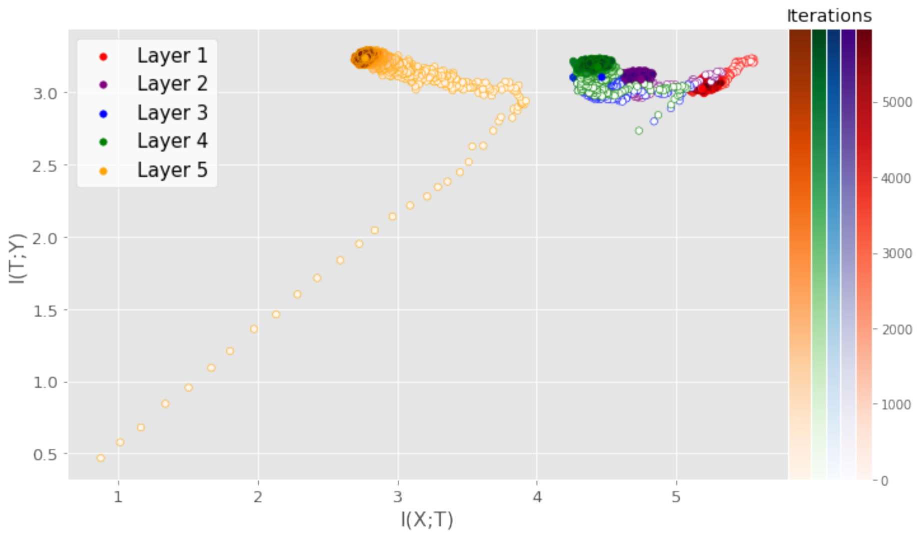

Figure A2. IP of an MLP consisting of four fully connected layers with 1024, 20, 20, and 20 neurons and a tanh activation function in each hidden layer. MI was measured using the training data of the MNIST dataset and averaged over 5 runs.

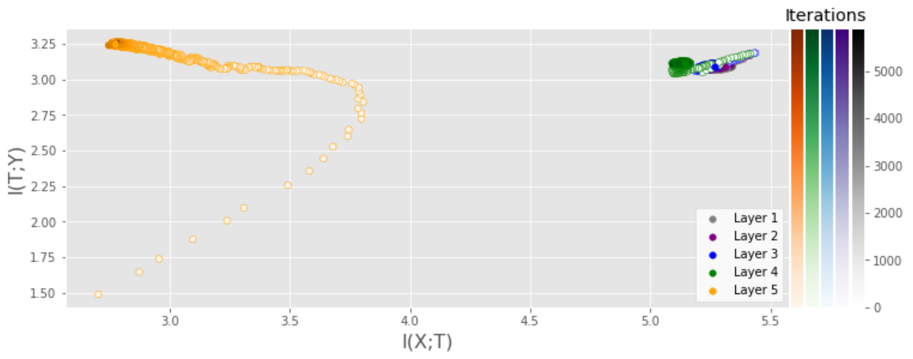

Figure A3. IP of a CNN consisting of three convolutional layers with 4, 8 and 12 filters and one fully connected layer with 256 neurons and a tanh activation function in each hidden layer. MI was measured using the training data of the MNIST dataset and averaged over 5 runs.

Appendix C. Structure Preserving Tensor Kernels and Numerical Instability of Multivariate Approach

As explained in Appendix A, the multivariate approach of [20] Equation (A1) struggles when the number of channels in an image tensor becomes large as a result of the Hadamard products in Equation (A1). To illustrate this instability, we have conducted a simple example. A subset of 50 samples is extracted from the MNIST dataset. Then, each image is duplicated (plus some noise) C times along the channel dimension of the same image, i.e., going from a grayscale image of size $(1, 1, 28, 28)$ to a new image of size $(1, C, 28, 28)$. Since the same image is added along the channel dimension, the kernel matrix should not change dramatically. Figure A1 displays the results of the experiment just described. The first row of Figure A1 shows the kernel matrices based on the multivariate approach proposed by [20]. When the tensor data only have one channel (first column), the kernel obtained is identical to the results obtained using the tensor kernel described in this paper. However, as the number of channels increases, the off-diagonal quickly vanishes and the kernel matrix tends toward a diagonal matrix. This is a result of vanishing Hadamard products, as described in Appendix A. Theoretically, the multivariate approach should yield the same kernel as with the tensor kernel approach, as explained in Section 3.3, but the off-diagonal elements decrease so quickly that they fall outside numerical precision. The second row of Figure A1 depicts the kernel matrices obtained using the tensor kernel approach described in Section 3.3. The kernel matrices in this row are almost unchanged as the number of channels increases, which is to be expected. Since the same image is added along the channel dimension, the similarity between the samples should not change drastically, which is what this row demonstrates. The third row of Figure A1 displays the kernel matrices obtained using so-called matricization-based tensor kernels [32], which are tensor kernels that preserve structure between the channels of the tensor. In this case, this approach produces similar results to the tensor kernel used in this paper, which is to be expected. Since the same image is added along the channel dimension, there is little information to extract between the channels. We hypothesize that for small images with centered objects, such as with MNIST and CIFAR10, the structured tensor kernel does not capture much more information than the tensor kernel described in Section 3.3. However, for more complex tensor data, exploring the potential of such structure-preserving tensor kernels is an interesting avenue for future studies.

Appendix D. Detailed Description of Networks from Section 4

We provide a detailed description of the architectures utilized in Section 4 of the main paper. Weights were initialized according to [42] when the ReLU activation function was applied and initialized according to [43] for the experiments conducted using the tanh activation function. Biases were initialized as zeros for all networks. A learning rate of 0.09 was used for the gradient descent algorithm. All networks were implemented using the deep learning framework Pytorch [44].

Appendix D.1. Multilayer Perceptron Used in Section 4

The MLP architecture used in our experiments is inspired by the architecture utilized in previous studies on the IP of DNN [4,5] but with batch normalization [45] included after the activation function of each hidden layer and an extra hidden layer. Specifically, the MLP in Section 5 includes (from input to output) the following components:

1. Fully connected layer with 784 inputs and 1024 outputs.
2. Activation function.
3. Batch normalization layer.
4. Fully connected layer with 1024 inputs and 20 outputs.
5. Activation function.
6. Batch normalization layer.
7. Fully connected layer with 20 inputs and 20 outputs.
8. Activation function.
9. Batch normalization layer.
10. Fully connected layer with 20 inputs and 20 outputs.
11. Activation function.
12. Batch normalization layer.
13. Fully connected layer with 784 inputs and 10 outputs.
14. Softmax activation function.

Appendix D.2. Convolutional Neural Network Used in Section 4

The CNN architecture in our experiments is a similar architecture to the one used by [4]. Specifically, the CNN in Section 5 includes (from input to output) the following components:

1. Convolutional layer with 1 input channel and 4 filters, filter size 3×3, stride of 1 and no padding.
2. Activation function.
3. Batch normalization layer.

4. Convolutional layer with 4 input channels and 8 filters, filter size 3×3, stride of 1 and no padding.
5. Activation function.
6. Batch normalization layer.
7. Max pooling layer with filter size 2×2, stride of 2 and no padding.
8. Convolutional layer with 8 input channels and 16 filters, filter size 3×3, stride of 1 and no padding.
9. Activation function.
10. Batch normalization layer.
11. Max pooling layer with filter size 2×2, stride of 2 and no padding.
12. Fully connected layer with 400 inputs and 256 outputs.
13. Activation function.
14. Batch normalization layer.
15. Fully connected layer with 256 inputs and 10 outputs.
16. Softmax activation function.

Appendix E. IP of MLP with Tanh Activation Function from Section 4

Figure A2 displays the IP of the MLP described above with a tanh activation function applied in each hidden layer. Similarly to the ReLU experiment in the main paper, a fitting phase is observed, where both $I(T;X)$ and $I(Y;T)$ increase rapidly, which is followed by a compression phase where $I(T;X)$ decreases and $I(Y;T)$ remains unchanged. In addition, note that similar to the ReLU experiment, $I(Y;T)$ stabilizes close to the theoretical maximum value of $\log_2(10)$.

Appendix F. IP of CNN with Tanh Activation Function from Section 4

Figure A3 displays the IP of the CNN described above with a tanh activation function applied in each hidden layer. Just as for the CNN experiment with the ReLU activation function in the main paper, no fitting phase is observed for the majority of the layers, which might indicate that the convolutional layers can extract the essential information after only a few iterations.

Appendix G. Data Processing Inequality with EDGE MI Estimator

Figure A4 displays the DPI of a simple MLP using the EDGE estimator as described by [4]. Results indicate that the EDGE estimator upholds the DPI. This agrees with our results with regard to the information flow in neural networks.

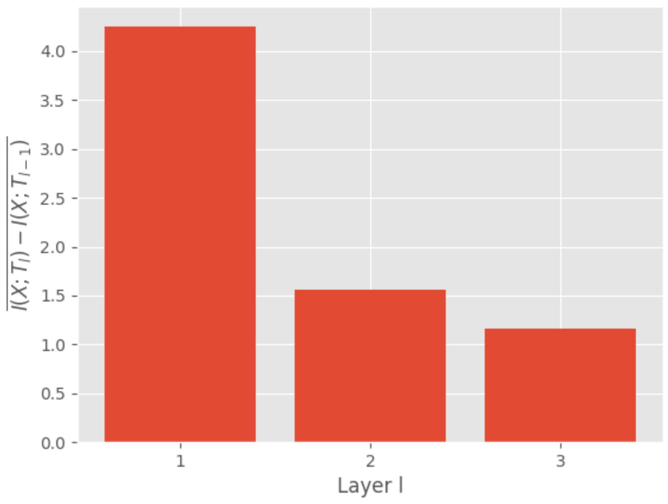

Figure A4. Mean difference in MI of subsequent layers ℓ and $\ell + 1$. Positive numbers indicate compliance with the DPI. MI was measured on the MNIST training using the EDGE MI estimator on a simple MLP [4].

Appendix H. Connection with Epoch-Wise Double Descent

One of the fundamental questions of deep learning is how heavily overparameterized models generalize well to unseen data. Recent studies have empirically demonstrated that having many more parameters than training points can be beneficial for generalization, which have been analyzed from the perspective of the double descent phenomenon [38,39]. The double descent phenomenon was proposed to qualitatively describe the behavior of complex models when the number of parameters is much larger than the number of samples, which is the heavily overparametrized regime that DNNs often operate in. Moreover, ref. [39] showed that overparametrized models can also exhibit epoch-wise double descent. Epoch-wise double descent refers to a phenomenon where the test error initially decreases and then increases before finally decreasing again and stabilizing. Such a behavior bears resemblance to the information flow discussed in this paper, where the information initially increases before decreasing.

To investigate the potential connection between our information theoretic analysis of DNNs and the epoch-wise double descent phenomenon, we train a heavily overparametrized neural network on a subset of the MNIST dataset (10,000 samples) inspired by [38]. The overparametrized network consists of one hidden layer and the output layer, where the hidden layer contains 50,000 neurons with a ReLU activation function. The training and analysis of this network is carried out in a similar manner as all the other reported experiments. Figure A5 displays training and test loss together with the MI between the input and each layer of the network. First, notice that the epoch-wise double descent phenomenon is clearly visible, as the test loss initially decreases, which is followed by an increase, and toward the end, there is another period of decrease. Simultaneously, the start of compression in $MI(X,\hat{Y})$ seems to coincide with the increase of the test loss, and it ends when the test loss stabilizes after going down again. These results suggests that there might be a link with the compression phase of DNNs and the epoch-wise double descent phenomenon, and that information theory can be used to gain new insights into the generalization capabilities of DNNs.

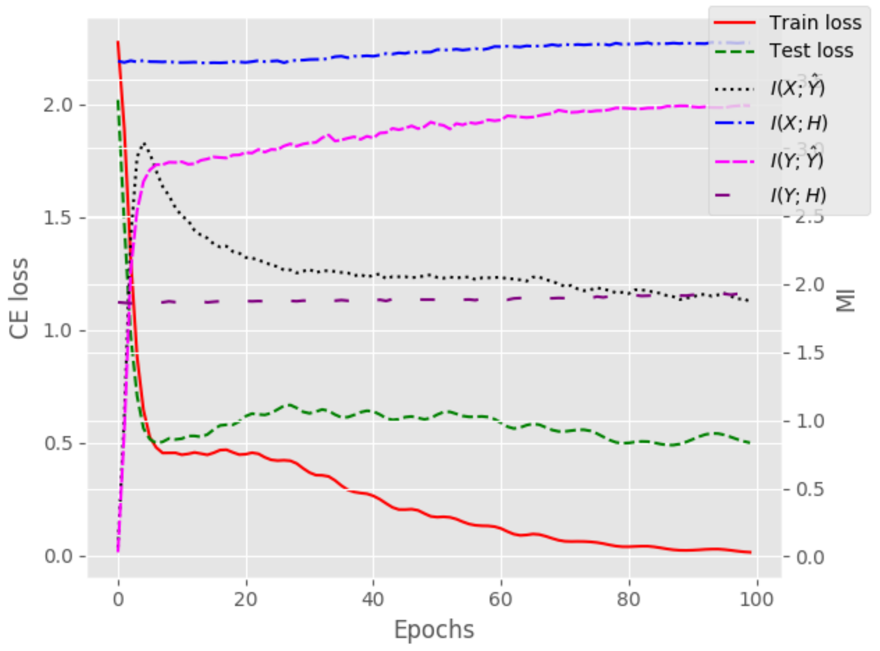

Figure A5. Training and test loss of neural network with one hidden layer with 50,000 neurons on a subset of the MNIST dataset. The figure also shows the MI between the input/labels and the hidden/output layer. The epoch-wise double descent phenomenon is visible in the test loss, and it seems to coincide with the start of the compression of MI between the input and output layer. Notice the different labels on the left and right y-axis. Curves represent the average over 3 training runs.

References

1. Shwartz-Ziv, R.; Tishby, N. Opening the Black Box of Deep Neural Networks via Information. *arXiv* **2017**, arXiv:1703.00810. [CrossRef]
2. Geiger, B.C. On Information Plane Analyses of Neural Network Classifiers—A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 7039–7051. [CrossRef]
3. Cheng, H.; Lian, D.; Gao, S.; Geng, Y. Evaluating Capability of Deep Neural Networks for Image Classification via Information Plane. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 168–182.
4. Noshad, M.; Zeng, Y.; Hero, A.O. Scalable Mutual Information Estimation Using Dependence Graphs. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, UK, 12–17 May 2019; pp. 2962–2966.
5. Saxe, A.M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B.D.; Cox, D.D. On the information bottleneck theory of deep learning. *J. Stat. Mech. Theory Exp.* **2019**, *2019*, 124020. [CrossRef]
6. Yu, S.; Wickstrøm, K.; Jenssen, R.; Príncipe, J.C. Understanding Convolutional Neural Network Training with Information Theory. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 435–442. [CrossRef]
7. Yu, S.; Principe, J.C. Understanding autoencoders with information theoretic concepts. *Neural Netw.* **2019**, *117*, 104–123. [CrossRef] [PubMed]
8. Lorenzen, S.S.; Igel, C.; Nielsen, M. Information Bottleneck: Exact Analysis of (Quantized) Neural Networks. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
9. Goldfeld, Z.; Van Den Berg, E.; Greenewald, K.; Melnyk, I.; Nguyen, N.; Kingsbury, B.; Polyanskiy, Y. Estimating Information Flow in Deep Neural Networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 2299–2308.
10. Chelombiev, I.; Houghton, C.; O'Donnell, C. Adaptive Estimators Show Information Compression in Deep Neural Networks. *arXiv* **2019**, arXiv:1902.09037. [CrossRef]
11. Zhouyin, Z.; Liu, D. Understanding Neural Networks with Logarithm Determinant Entropy Estimator. *arXiv* **2021**, arXiv:2105.03705. [CrossRef]
12. Sanchez Giraldo, L.G.; Rao, M.; Principe, J.C. Measures of Entropy From Data Using Infinitely Divisible Kernels. *IEEE Trans. Inf. Theory* **2015**, *61*, 535–548. [CrossRef]
13. Tishby, N.; Zaslavsky, N. Deep learning and the information bottleneck principle. In Proceedings of the 2015 IEEE Information Theory Workshop (ITW), Jerusalem, Israel, 26 April–1 May 2015.
14. Kolchinsky, A.; Tracey, B.D.; Kuyk, S.V. Caveats for information bottleneck in deterministic scenarios. *arXiv* **2019**, arXiv:1808.07593. [CrossRef]
15. Amjad, R.A.; Geiger, B.C. Learning Representations for Neural Network-Based Classification Using the Information Bottleneck Principle. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, 42, 2225–2239. [CrossRef]
16. Kornblith, S.; Norouzi, M.; Lee, H.; Hinton, G. Similarity of Neural Network Representations Revisited. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 3519–3529.
17. Jónsson, H.; Cherubini, G.; Eleftheriou, E. Convergence Behavior of DNNs with Mutual-Information-Based Regularization. *Entropy* **2020**, *22*, 727. [CrossRef] [PubMed]
18. Belghazi, M.I.; Baratin, A.; Rajeshwar, S.; Ozair, S.; Bengio, Y.; Courville, A.; Hjelm, D. Mutual Information Neural Estimation. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; Volume 80, pp. 531–540.
19. Geiger, B.C.; Kubin, G. Information Bottleneck: Theory and Applications in Deep Learning. *Entropy* **2020**, *22*, 1408. [CrossRef] [PubMed]
20. Yu, S.; Sanchez Giraldo, L.G.; Jenssen, R.; Principe, J.C. Multivariate Extension of Matrix-based Renyi's α-order Entropy Functional. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *42*, 2960–2966. [CrossRef]
21. Landsverk, M.C.; Riemer-Sørensen, S. Mutual information estimation for graph convolutional neural networks. In Proceedings of the 3rd Northern Lights Deep Learning Workshop, Tromso, Norway, 10–11 January 2022; Volume 3.
22. Bhatia, R. Infinitely Divisible Matrices. *Am. Math. Mon.* **2006**, *113*, 221–235. [CrossRef]
23. Renyi, A. On Measures of Entropy and Information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*; University of California Press: Berkeley, CA, USA, 1961; pp. 547–561.
24. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: Cambridge, UK, 2011.
25. Mosonyi, M.; Hiai, F. On the Quantum Rényi Relative Entropies and Related Capacity Formulas. *IEEE Trans. Inf. Theory* **2011**, *57*, 2474–2487. [CrossRef]
26. Kwak, N.; Choi, C.H. Input feature selection by mutual information based on Parzen window. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 1667–1671. [CrossRef]
27. Gretton, A.; Borgwardt, K.M.; Rasch, M.J.; Schölkopf, B.; Smola, A. A Kernel Two-sample Test. *J. Mach. Learn. Res.* **2012**, *13*, 723–773.
28. Muandet, K.; Fukumizu, K.; Sriperumbudur, B.; Schölkopf, B. Kernel Mean Embedding of Distributions: A Review and Beyond. In *Foundations and Trends® in Machine Learning*; Now Foundations and Trends: Boston, FL, USA, 2017; pp. 1–141.

29. Fukumizu, K.; Gretton, A.; Sun, X.; Schölkopf, B. Kernel Measures of Conditional Dependence. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2008; pp. 489–496.
30. Smola, A.; Gretton, A.; Song, L.; Schölkopf, B. A Hilbert Space Embedding for Distributions. In Proceedings of the Algorithmic Learning Theory, Sendai, Japan, 1–4 October 2007; Hutter, M., Servedio, R.A., Takimoto, E., Eds.; pp. 13–31.
31. Evans, D. A Computationally Efficient Estimator for Mutual Information. *Proc. Math. Phys. Eng. Sci.* **2008**, *464*, 1203–1215. [CrossRef]
32. Signoretto, M.; De Lathauwer, L.; Suykens, J.A. A kernel-based framework to tensorial data analysis. *Neural Netw.* **2011**, *34*, 861–874. [CrossRef]
33. Shi, J.; Malik, J. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 888–905.
34. Shi, T.; Belkin, M.; Yu, B. Data spectroscopy: Eigenspaces of convolution operators and clustering. *Ann. Stat.* **2009**, *37*, 3960–3984. [CrossRef]
35. Silverman, B.W. *Density Estimation for Statistics and Data Analysis*; CRC Press: Boca Raton, FL, USA, 1986; Volume 26.
36. Cristianini, N.; Shawe-Taylor, J.; Elisseeff, A.; Kandola, J.S. On kernel-target alignment. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2002; pp. 367–373.
37. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
38. Belkin, M.; Hsu, D.; Ma, S.; Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 15849–15854. [CrossRef] [PubMed]
39. Nakkiran, P.; Kaplun, G.; Bansal, Y.; Yang, T.; Barak, B.; Sutskever, I. Deep Double Descent: Where Bigger Models and More Data Hurt. *arXiv* **2020**, arXiv:1912.02292. [CrossRef]
40. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; Wiley Series in Telecommunications and Signal Processing; Wiley-Interscience: New York, NY, USA, 2006.
41. Yu, X.; Yu, S.; Príncipe, J.C. Deep Deterministic Information Bottleneck with Matrix-Based Entropy Functional. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Toronto, ON, Canada, 6–11 June 2021; pp. 3160–3164.
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
43. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
44. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in PyTorch. 2017. Available online: https://pytorch.org (accessed on 1 January 2023).
45. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.