INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 20, Issue: 1, Month: February, Year: 2025
Article Number: 6904, https://doi.org/10.15837/ijccc.2025.1.6904

CCC Publications

Transfer Entropy in Deep Neural Networks

R. Andonie, A. Caţaron, A. Moldovan

Răzvan Andonie
1. Department of Computer Science
Central Washington University, USA
2. Department of Electronics and Computers
Transilvania University of Braşov, Romania
*Corresponding author: razvan.andonie@cwu.edu

Angel Caţaron
1. Department of Electronics and Computers
Transilvania University of Braşov, Romania
2. Siemens Research and Predevelopment, Siemens SRL, Braşov, Romania
cataron@unitbv.ro

Adrian Moldovan
Siemens Research and Predevelopment, Siemens SRL, Braşov, Romania
adrian.moldovan@siemens.com

Abstract

This paper explores the application of Transfer Entropy (TE) in deep neural networks as a tool to improve training efficiency and analyze causal information flow. TE is a measure of directed information transfer that captures nonlinear dependencies and temporal dynamics between system components. The study investigates the use of TE in optimizing learning in Convolutional Neural Networks and Graph Convolutional Neural Networks. We present case studies that demonstrate reduced training times and improved accuracy. In addition, we apply TE within the framework of the Information Bottleneck theory, providing an insight into the trade-off between compression and information preservation during the training of deep learning architectures. The results highlight TE's potential for identifying causal features, improving explainability, and addressing challenges such as oversmoothing in Graph Convolutional Neural Networks. Although computational overhead and complexity pose challenges, the findings emphasize the role of TE in creating more efficient and interpretable neural models.

Keywords: Transfer entropy, causality, deep learning, neural network explainability

1 Introduction

Causality refers to a direct relationship in which one event or variable directly influences or produces changes in another. In the classical philosophical view of David Hume, causality is characterized by [1]: Temporal precedence (cause precedes effect); Constant conjunction (cause and effect regularly occur together); Counterfactual dependence (without cause, the effect would not occur).

Transfer Entropy (TE), introduced by Thomas Schreiber [2], is a nonparametric unilateral information measure between two random processes X and Y. It measures how much the past of X helps predict the future of Y beyond what the past of Y itself tells us. TE was commonly used to quantify the degree of coherence between events, usually those represented as time series. It was sometimes related to Granger's causality, where the process X is said to be the Granger-cause of process Y if future realizations of Y can be better explained using past information from X and Y rather than Y alone.

It should be noted that TE is not a general measure of causality since it implies only statistical relationships and information flow, not necessarily causal relationships. A high TE value from X to Y means that X provides predictive information about Y, but does not prove that X causes Y. In addition, TE is usually applied to time series data, while causality often requires controlled experiments or additional assumptions to establish. On the bright side, TE can detect indirect and nonlinear relationships that might be missed by traditional causal analysis methods.

There is a growing interest in applying TE in quantifying the effective connectivity between artificial neurons. TE can measure the strength of a connection between neurons and we can use this measure as feedback to amplify the connection.

The focus of this paper is on how to use TE to improve training and interpretability in massive neural models. Deep learning neural architectures are hard to train due to their increasing complexity and the size of the datasets used. Our objective is to design more efficient training algorithms that utilize, if possible, causal relationships inferred from neural networks. Meanwhile, we are also interested in using TE to better understand and interpret the decision-making process of neural networks.

The potential applications of TE in neural networks are very promising. We could use TE to identify input features that causally affect the output and possibly reduce dimensionality. We could also analyze the information flow in biologically inspired neural networks to compare artificial and biological systems. Not to omit that TE can reveal how perturbations in inputs propagate through the network, identifying fragile pathways or spurious correlations. A challenging application is to identify which neurons or layers contribute the most to decision making, helping to explain AI models.

We face several challenges in this approach. Estimation of TE in neural networks is far from trivial and implies significant computational overhead. The connection between TE and causality is not always well understood, and it is very difficult to achieve complete interpretability of a neural model from its TE calculations.

Our current contribution is a comprehensive description and interpretation of TE applications used in deep learning. We present an overview of our research results for optimizing the learning phase of deep learning models: Convolutional Neural Networks (CNNs) and Graph Convolutional Neural Networks (GCNs). We also focus on the information flow quantified by TE in neural networks, including in Information Plane (IP) analysis. IP is a visualization technique used to understand the trade-off between compression and information preservation in the context of Information Bottleneck (IB) framework.

The paper proceeds as follows. Section 2 analyzes TE in the context of causality and interpretability. Section 3 describes our practical method for calculating TE in neural networks. In Section 4 we present several case studies using CNNs and GCNs. Section 5 contains our final remarks.

2 TE and Causal Information Flow in Neural Networks - Concepts and Previous Work

Using TE to study causality in neural networks involves quantifying the flow of information between different components of the network, such as neurons, layers, or even inputs and outputs. TE is particularly well-suited for neural networks because it accounts for nonlinear dependencies and temporal dynamics, which are inherent in many neural systems. This section discusses the causal interpretation of TE in neural networks.

Causality is a fundamental concept that is used to describe the relationship between causes and their effects. Definitions of causality can vary depending on the context (philosophy, physics, etc.), but the core idea remains the same: causality describes how one event (the cause) directly influences another event (the effect).

Within any causality model, there is a correlation function [3]. However, causal analysis goes one step further than statistical analysis, as it aims to infer not only the likelihood of events under static conditions but also the dynamics of events under changing conditions [4]. Practically, it is very difficult to establish the causality between two correlated events. In contrast, it is relatively easy to establish a statistically significant correlation. Human intuition has evolved in such a way that it has learned to identify causality through correlation. This is due to the inability to detect a time lag between a cause and effect, which is a prerequisite for causality [3].

There are three key criteria for inferring a cause-effect relationship, defined as early as 1882 by Mill [5]: (1) the cause preceded the effect, (2) the cause was related to the effect, and (3) we cannot find any plausible alternative explanation for the effect other than the cause. These criteria match Hume's characterization of causality from 1739 [1], which uses philosophical terminology.

In the transmission theory of causality, causal information is transmitted from the cause to the effect [3]. This was described in detail by several authors, including Salmon [6]. In the context of neural networks, Salmon's definition of causality involves drawing parallels between his concepts of causal processes, causal interactions, and the computational mechanisms of neural networks. In Salmon's framework, a causal process is an entity that transmits causal influence.

In neural networks, the output of neurons represents the flow of information or influence within the network. Salmon's "mark transmission criterion" states that a causal process must be able to carry a change that persists. In neural networks, such a change can be thought of as a perturbation in the input data (e.g., a slight change in pixel values of an image) that causes detectable changes in the output activations. If the network is causal in Salmon's sense, the input change will propagate through the layers and influence the final output. Causal interactions occur when the outputs of previous neurons are combined. This summation and activation represent a causal interaction in which multiple processes influence the result. This ability to propagate changes through the network may demonstrate that the neural network embodies Salmon's concept of causal processes.

Explainable machine learning is a very hot research area, especially for deep neural models [7, 8, 9, 10]. Neural networks are complex and often lack transparency, making it difficult to trace clear causal pathways. Currently, several advanced techniques are used to make neural networks more interpretable by identifying causal relationships:

- *Saliency maps* [11], where a heatmap on an image can show which pixels are "causally" relevant for a classification decision.
- *Shapley values* [12], where a Shapley value in a neural network context represents the contribution of each individual input feature to the final prediction, calculated considering how the prediction changes when that feature is added or removed from a hypothetical set of features.
- *Counterfactual analysis* [13], where in the context of neural networks, we use the model to explore "what-if" scenarios by generating hypothetical alternative inputs, essentially asking "what would the prediction have been if this input value was different?".
- *Causal graphs*, where neural networks can be augmented with causal graph models that explicitly represent causal relationships between input features and outputs [14].
- *Semiotic superization* [15, 16], a semiotic aggregation interpretation of inference mechanisms in neural networks.

These methods identify how input features (causal processes) propagate their influence through the network and contribute to the final output. This aligns with Salmon's emphasis on understanding causal mechanisms rather than just correlations, as these techniques show the pathways of causal influence in the network. During training, the weights of a neural network represent the mechanism that governs causal interactions between neurons. Backpropagation can be viewed as the identification and adjustment of these causal pathways to better align the network behavior with the desired outcomes.

Therefore, Salmon's definition provides a useful lens for understanding neural networks as systems where "causal processes" correspond to the flow of activations, "causal interactions" happen when neurons aggregate and transform inputs, and "propagation of marks" corresponds to how changes in inputs (or intermediate states) propagate and influence the outputs. This perspective reinforces the interpretability and mechanistic understanding of neural networks, especially when exploring their internal causal structures.

Causality in the context of time-series data is often posed using two major frameworks: Granger causality and the information-theoretical approach (e.g., based on the Kullback-Leibler divergence or TE).

The Granger[1] causality test is a statistical hypothesis test to determine whether one time series is useful to forecast another. According to Granger, causality could be reflected by measuring the ability to predict the future values of a time series using past values of another time series. The Granger test is based on linear regression modeling of stochastic processes. More complex extensions to nonlinear cases exist, but these extensions are more difficult to apply in practice. A weakness of Granger's causality is that it can make one infer causality when the reason is that the two variables (time series) just happen to have common cause [3].

TE was introduced by Schreiber [2] not as a causality indicator but as a *information transfer measure* used to quantify statistical coherence between time series. In general, information transfer refers to a directional signal or the communication of dynamic information from a source to a destination. The TE is capable of distinguishing driving and responding elements and detecting asymmetry in the interaction of time series. For example, in the financial market, based on the TE concept, Kwon *et al.* [17] found that the amount of information transfer from index to stock is greater than from index to index. This indicates that the market index plays an important driving force for the individual stock. An excellent introduction to TE with applications is [18].

Later, TE was related to Granger causality. Barnett *et al.* [19] proved that the Granger causality and the TE causality measure are equivalent for time series which have a Gaussian distribution. Hlaváčková-Schindler generalized this result in 2011 [20].

However, there are differences between the general concept of causality (not necessarily Granger causality) and information transfer. Causality is typically related to whether interventions on a source can be identified to have an effect on the target, rather than whether observations of the source can help predict state transitions on the target.

In conclusion, TE is a dynamic directional measure of predictive information, rather than a measure of the flow of causal information from a source to a destination. In our paper, we use the information transfer, measured by TE, to quantify causal relationships *only* between the information sources and a given destination.

TE was recently used to quantify and interpret causal relationships in neural networks [21, 22, 23, 24, 25]. Kim *et al.* [26] predicted the direction of stock prices using TE and several machine learning methods. Another application, combining TE and deep learning, is [27], where faulty sensor data were recovered for building air conditioning systems.

Only a few attempts were made to use TE to improve the learning mechanism of neural networks, our main focus here [28, 29, 30, 31, 32, 33]. The explanation and interpretation of the decisions made by a neural model is another hot research area. Féraud *et al.* [34] explained the classification obtained by a multilayer perceptron by introducing the concept of "causal importance" and defining a saliency measurement that allows the selection of relevant variables.

3 How To Calculate Transfer Entropy in Neural Networks

When applied to neural networks, TE can reveal the causal information flow between individual neurons, network layers, or inputs and outputs. This section explains in detail how TE can be used to quantify the information flow in neural networks.

[1]Clive Granger, recipient of the 2003 Nobel Prize in Economics.

3.1 TE Definition in Neural Networks

We start with the mathematical definition of TE, an extension of Shannon's entropy that quantifies how much information of time series J influences time series I [2]:

$$TE_{J \to I} = \sum_{t=1}^{n-1} p(i_{t+1}, i_t, j_t)\, log \frac{p(i_{t+1} \mid i_t, j_t)}{p(i_{t+1} \mid i_t)} \tag{1}$$

During training neural networks, each neuron produces an activation at a given time t. TE between two neurons can quantify how much information neuron j's activation contributes to predicting the future activation of neuron i.

In a multilayer perceptron (MLP) network, at the layer level, we can measure both *Forward Causality* (how much information flows from one layer to the next) and *Backward Causality* (how much feedback from deeper layers affects earlier layers). Backward causality was recently introduced as Backward TE to measure effect-cause relationships [35].

TE can also be used to evaluate the influence of specific input features on the network output: for a feature x, we calculate $TE_{x \to y}$, where y is the prediction of the network. This helps identify causal features, distinguishing them from irrelevant or spurious features.

In our approach, we compute the TE between individual directly connected neurons, knowing that only a few connections are truly relevant. This gives us more flexibility in inferring important causal relationships since we use a finer granularity. In a backpropagation-trained feedforward neural network, we add the factor $(1 - te^l_{j,i})$ to the weight update formula of the backpropagation algorithm:

$$\Delta w^l_{ij} = -\eta \frac{\partial C}{\partial w^l_{ij}} (1 - te^l_{j,i}), \tag{2}$$

where C is the loss function and te is the computed value of TE for the neuron pair j to i in layer l.

If we attempt to interpret the TE values, a lower value compared to the median implies that the activations of the neuron pairs involved in the TE are similar (for classification tasks) and therefore not immediately useful.

Our aim is to provide a *corrective* update of the weights, especially when we record *disruptive* TE values. Such disruptive values are related to significant discrepancies between the activations of connected neurons. By negating the TE values in term $(1 - te^l_{j,i})$, we apply a prominent update to the weights corresponding to a pair of neurons with a large TE. We do not introduce the derivative of TE in Equation 2 since the updates of the TE values are usually small compared to the gradient values.

The range of TE values changes from large to small during training and also within an epoch. Implementing Equation 2 in the backpropagation algorithm provides improvements in the validation accuracy of a MLP. Using TE yields good results under the assumption that modified weights can positively *steer* the network performance. However, using Equation 2 in deep learning architectures and for large training datasets is less efficient, and a different approach is needed.

We will describe in detail how we construct and use the time series for the TE computation. The length of the time series is determined empirically. Since computing TE is a highly resource-demanding task, we first evaluate various settings that maximize performance gain while minimizing TE computations. For MLP architectures, we found that computing TE for all epochs and all neuron pairs yields the best results. This is computationally feasible as long as the number of MLP's neurons is relatively small.

During neural network training, we collect neuron activations from all pairs of neurons from successive layers. To construct the time series needed for TE, we binarize the collected activations into time series I and J, as illustrated in Fig. 1, where g represents the binarization threshold. For MLP architectures, the network is trained in two stages: *a)* the activations are binarized into time series over epochs for the entire dataset, and *b)* TE is computed using the recorded time series, resulting in a single TE value for each pair of neurons. Subsequently, another network with identical characteristics is trained, maintaining the same order of input patterns. This time, Equation 2 is used, incorporating the computed TE values for all updates.

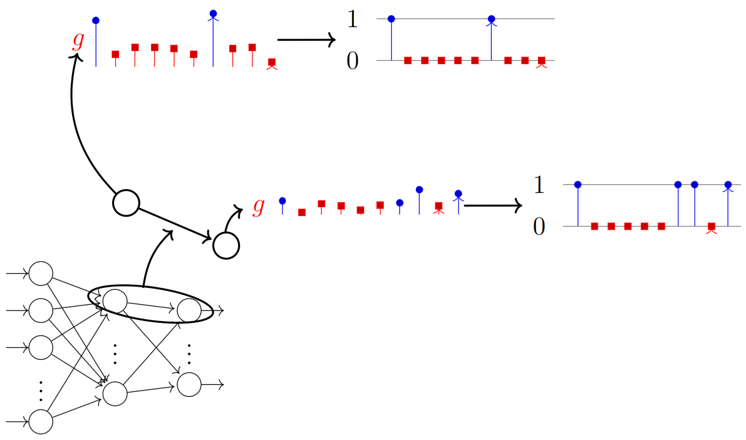

Figure 1: Binarizing the activations from a pair of neurons in two adjacent layers. A fixed thresholding parameter g is used to control how many zeroes and ones are produced during binarization. This parameter is empirically determined and varies for each architecture and dataset. The value of g is selected to ensure that it produces fewer 1s than 0s. The distribution of binary values changes during the training process, directly affecting the calculated TE values [30].

For CNN architectures, we restrict the length of the time series to keep computational costs manageable. Using only the last two layers of the CNN yields the best results with minimal computations, confirming our findings from [32].

The number of neurons in the fully connected layers of a CNN is significantly larger than in regular MLPs. To obtain meaningful TE values, we need to balance the length of the time series with the computational cost. The length of the time series is similar to the length of the context, which makes this parameter closely related to the training mode. In batch training, regularization and gradient updates typically target the batch size. For large datasets, longer time series tend to produce TE values close to 0, in the range of 10^{-4} to 10^{-6}, making them ineffective. Due to the nature of batch normalization, updates in the backpropagation mechanism do not yield sufficient improvements over an epoch for small TE values.

Our best results were achieved by computing the TE over a batch of inputs, where the TE batch size matched the CNN batch parameter specified in the mini-batch gradient descent algorithm [36]. Fig. 2 illustrates the process of constructing the time series using a sliding-window mechanism applied to a series of training patterns. The size of the sliding window is the same as the gradient descent's parameter batch size.

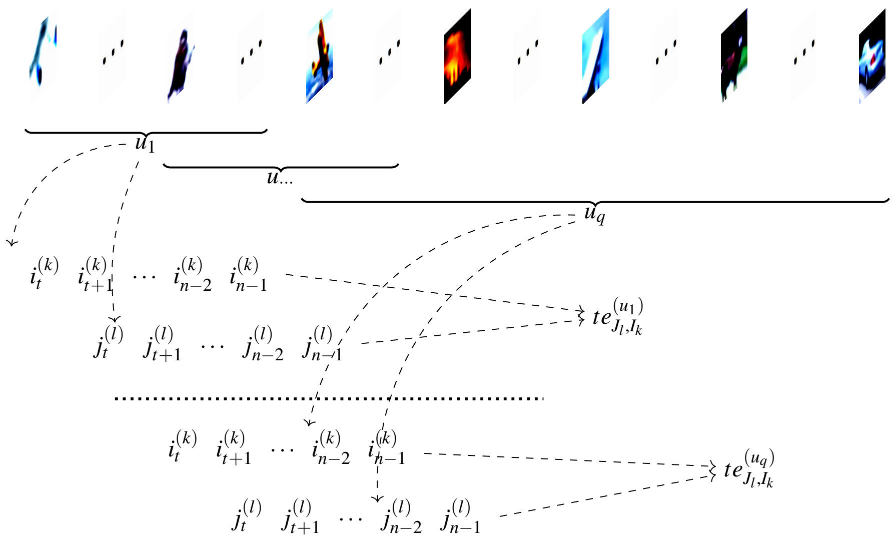
u_1
$u_{\dots}$
u_q
$i_t^{(k)}$ $i_{t+1}^{(k)}$ $\cdots$ $i_{n-2}^{(k)}$ $i_{n-1}^{(k)}$
$j_t^{(l)}$ $j_{t+1}^{(l)}$ $\cdots$ $j_{n-2}^{(l)}$ $j_{n-1}^{(l)}$
$te_{J_l,I_k}^{(u_1)}$
$i_t^{(k)}$ $i_{t+1}^{(k)}$ $\cdots$ $i_{n-2}^{(k)}$ $i_{n-1}^{(k)}$
$j_t^{(l)}$ $j_{t+1}^{(l)}$ $\cdots$ $j_{n-2}^{(l)}$ $j_{n-1}^{(l)}$
$te_{J_l,I_k}^{(u_q)}$

Figure 2: How we calculate TE in a CNN [31].

For larger and deeper architectures, Equation 2 does not yield a stable and consistently positive accuracy improvement. Our experiments in [31] focused on supervised image classification tasks. In these experiments, we computed the TE between the last two layers as illustrated in Fig. 3. After computing TE from the last two successive fully connected layers, we applied TE using the factor $(\mathbb{I} - (\boldsymbol{te})^\top)$ in the delta learning rule.

3.2 How to Estimate TE

Accurately estimating entropy-based measures, including TE, can be challenging. In the following, we will describe our practical implementation of TE estimation.

Initially, Schreiber estimated TE using generalized correlation integrals, but there is no consensus on an optimal approach to estimating TE from a data set. Instead, an alternative approach was proposed where transition probabilities are obtained by kernel estimation [2].

The fixed partition histogram estimation approach is the most widely used technique. This method is simple and efficient but is not scalable for more than three scalars. In addition, it is sensitive to the choice of bin size. Other entropy estimation methods have also been used for computing TE, such as kernel density estimation, nearest-neighbor techniques, and neural network-based approaches.

Estimating TE using variables extracted from neural network components is difficult for several reasons:

- TE must be calculated multiple times during a single training epoch of a neural network, which can lead to increased bias estimation error due to repetitive computation.
- The performance of TE estimation is influenced by the number of input variables, as the computational cost to compute TE is linearly correlated with the size of TE's inputs.
- The data sets used to approximate the TE should be large enough to produce reliable estimates.
- TE estimation techniques should be able to capture a broad range of TE distribution values, and these values should align with the distributions of the weights or gradients of the neural network where TE is actually used.

In [37] we used two common methods for estimating probabilities in TE: the binning method and the box-kernel method. The binning method involves the discretization of the continuous time-series data into a finite number of equally sized intervals (bins) and assigning each value to a bin. In this approach, the probabilities are the relative frequencies of the data points within each bin. However, the binning method is sensitive to the choice of bin size and may lead to loss of resolution if the bins are too large or to excessive fragmentation if bins are too small. The box-kernel method, a variant of Parzen window estimation, uses a nonparametric approach, estimating probabilities based on the proportion of data points within a defined radius or kernel around each target value. This method is more flexible and provides smoother probability distributions, particularly for small datasets or irregularly distributed data. In contrast to the binning method, it is computationally more demanding. The experiments demonstrated that both methods yield consistent results in estimating TE and Transfer Information Energy (TIE), with differences in computational efficiency and smoothness, depending on the dataset and method chosen. Both methods can be used to calculate the joint and conditional probabilities, which are essential to derive the TE values.

In [30], we estimated the probabilities in the TE equation by constructing time series from the discretized output of neurons during the neural network training process. These continuous neuron outputs were discretized using a binning process, where a threshold was applied to categorize the outputs into binary values. The probabilities were approximated as relative frequencies derived from the occurrence of specific combinations of binary states in the time series. The computed TE values were integrated into a modified version of the backpropagation algorithm, where TE was used as feedback to adjust the weights between neurons. Although a higher number of discrete levels can yield more accurate TE approximations, it requires longer time series and increases computational complexity.

The probabilities estimation we used in [30] was further extended in [31] by constructing time series from the activations of CNN neurons. To balance accuracy and computational overhead, we limited the TE computation to the last two fully connected layers, where classification decisions are made. The weight updates were adjusted by incorporating the factor $(1-te^{l}_{j,i})$ in the modified version of the backpropagation algorithm, ensuring that the weights associated with high information transfer connections are preserved, which accelerates convergence while improving the stability of the training.

3.3 Observations

Compared with the standard backpropagation, our training algorithm introduces two additional parameters: the binarization threshold and the length of the time series. A key question is whether these parameters could lead to overfitting and reduced generalization performance.

Computing TE for the updated backpropagation algorithm can introduce an overhead of up to several times compared to the original training process. This overhead is increased linearly with the product between the time series length and the number of neuron pairs for which the TE is calculated.

For large datasets, computing the TE for all pairs of neurons is prohibitive. However, it is not necessary to compute the TE for every pair of neurons. Selecting the right pairs of neurons for the computation of TE (as shown in [33]) produces very good results, comparable to those obtained when computing TE for all pairs of neurons.

Adding the TE feedback parameter generally accelerates the training process by reducing the number of epochs required to achieve the same accuracy. In MLP networks, for certain datasets, 7-10 times fewer epochs were needed to obtain the same accuracy. For CNNs, this approach consistently improved both the accuracy and the number of epochs required to achieve a target accuracy [31].

Using TE in the training mechanism of the MLP and CNN architectures offers several advantages. TE captures nonlinear dependencies, making it suitable for highly nonlinear systems like neural networks. Unlike correlation, TE identifies the direction of information flow (e.g., whether neuron i influences neuron j, or vice versa). In addition, TE does not require prior knowledge of the network structure.

Significant challenges remain. Neural networks often contain thousands or even millions of neurons, making it computationally expensive to estimate TE for all possible pairs. Accurate TE estimation requires large amounts of data to reliably compute probability distributions.

In terms of interpretability, we observe that high TE values indicate strong causal relationships but do not provide direct insights into the network's learned representations. This underscores the need to not only quantify causality, but also explain the results in a meaningful way.

4 Case Studies

This section summarizes, with new interpretations, our applications of TE in deep learning. Details can be found in our published papers [30, 31, 32, 33].

4.1 TE in CNNs

CNNs represent a major component of the deep learning landscape. In the search for increasing their precision, researchers have produced complex architectures that were difficult to improve. In [31], we proposed a generic mechanism that improves the accuracy of CNNs. Our results confirmed the findings of [32, 38, 39, 40].

In Section 3, we show how TE can be calculated in a CNN. We aim to give details on how TE can be used in the CNN learning mechanism. Our core idea is that the most influential layers in the fine-tuning stage of CNN training are the ones closest to the output. We computed TE between the last pair of layers that comprise the last fully connected layers for various CNN architectures, which are, in fact, the pre-softmax and softmax layers. The softmax layer transforms its input into probabilistic outputs [41].

Using the real-valued time series outputs from the neural network layer in the TE formula is not feasible due to computational burden. Binarizing the real-valued inputs simplifies the probabilities computation. We used a fixed threshold to binarize the inputs used in TE for the TE estimation techniques, then algebraically computed the probabilities of the members of the TE formula.

In a CNN, it is not practical to compute the TE for all layers, regardless of the estimation techniques used. Each layer and each dataset require its own threshold, as each architecture produces different distributions of neuron activations. The size of the time-series window is another hyperparameter that can influence two different behaviors: if it is too large, it can smoothen the TE values, diminishing any returns. If the window is too small, the computed TE will not selectively capture the distribution of the inputs. We observed that computing and using TE for an additional set of layers improved performance.

Our approach creates conditions for further information transfer *interpretation* within deep learning models. The statistical aspects of information transfer between CNN layers can provide insight into the feature abstraction process.

Essentially, our idea was to use TE as a momentum factor in the backward step of the error backpropagation and update the weights according to the unidirectional amount of information transferred between pairs of neurons (see Fig. 3). An example of how the TE is constructed between a pair of layers for the USPS dataset and architecture is shown in Fig. 4.

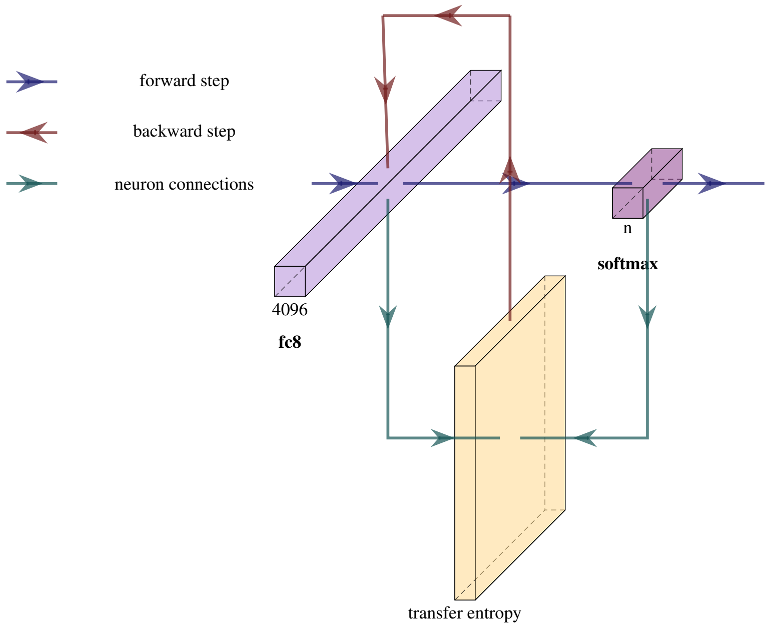

Figure 3: We collect activations and transform these into time series I and J. The green arrows indicate the flow of inputs that construct the TE layer which is later backpropagated into the network [31].

4.2 Information Adaptation Analysis in CNNs via TE

Information adaptation in neural networks refers to the process by which neural models modify or refine their internal representations to effectively handle changing input data. This adaptability is critical for generalization, robustness, and transfer learning.

Information *inflation* occurs when the representation of data is expanded, often to richer feature spaces. This allows the network to disentangle and separate complex features that are not linearly separable in the input space. In contrast, information *deflation* is the process of compressing the input data's representation, where irrelevant, or less useful, information is discarded while retaining essential features. This transformation helps the network generalize, focus on relevant patterns, and reduce computational complexity. This two-stage process is illustrated in Fig. 5.

Inflation-deflation duality was recently translated into the *fitting* - *compression* duality in IB theory. The IB principle posits that an effective neural network should compress input data into internal representations that retain only the most relevant information necessary for predicting the output. It

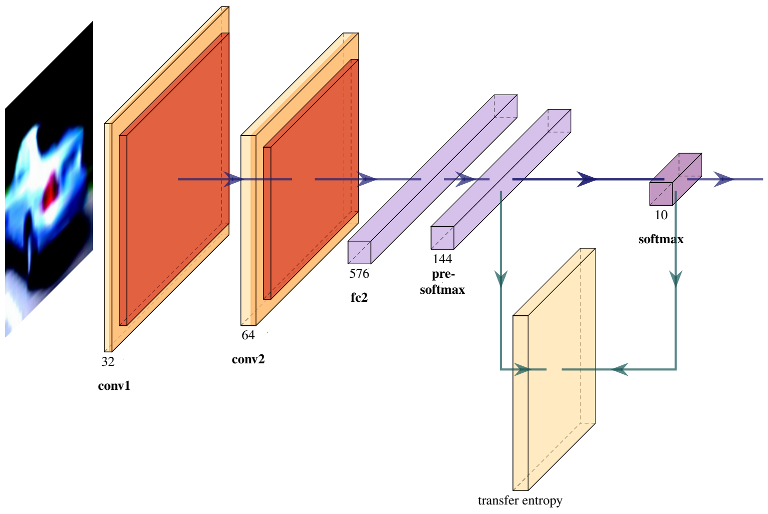

Figure 4: How we compute and use TE for the USPS dataset. The green arrows show the contributing neurons for TE calculus, during the feed-forward step. Each of the TE input layers required different threshold values for binarization which were determined empirically [31].

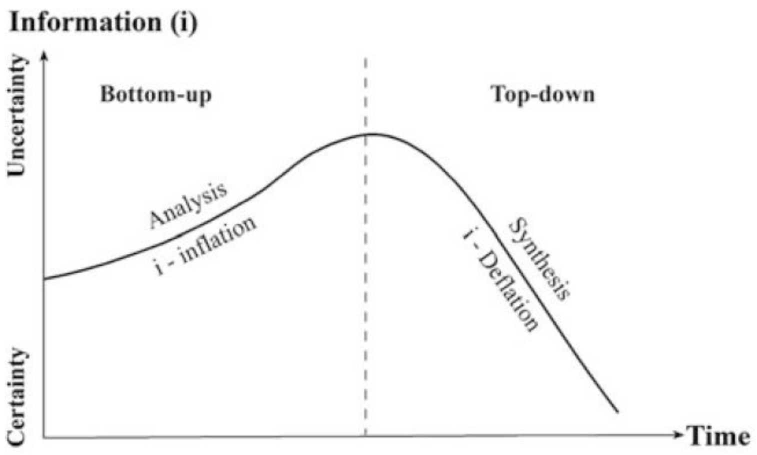

Figure 5: The 2-stages process of information adaptation: It starts with a bottom-up process of information inflation and continues with a top-down process of information deflation until the appropriate quantity of information is adapted to the required task [42].

was introduced by Tishby *et al.* [38, 39, 43] as a theory to explain some of the training dynamics of deep neural architectures. This is formalized by minimizing the objective function $min_{P_{T|X}}(I(X;T) - \beta I(Y;T))$. Based on IB theory, the two distinct phases, fitting and compression, characterize the mutual information (MI) of: *a)* the input X and the internal representation T; and *b)* the internal representation T and the output Y. Additionally, a good internal representation produced by a neural model should maximally compress the input data while preserving sufficient information about the output. The question is how to optimally balance fitting and compression so that we correctly predict an input. We want the most compact network with the best prediction accuracy. In terms of MI, we want to minimize MI between inputs and outputs while minimizing MI between inputs and a compressed distribution T.

Shwartz-Ziv and Tishby [39] used the IP analysis technique to visualize and explain the trade-off between compression and information preservation in the context of the IB method, by plotting the amount of information in the input data against the compressed representation (Fig. 6). The goal was to observe the dynamics of information flow and compression.

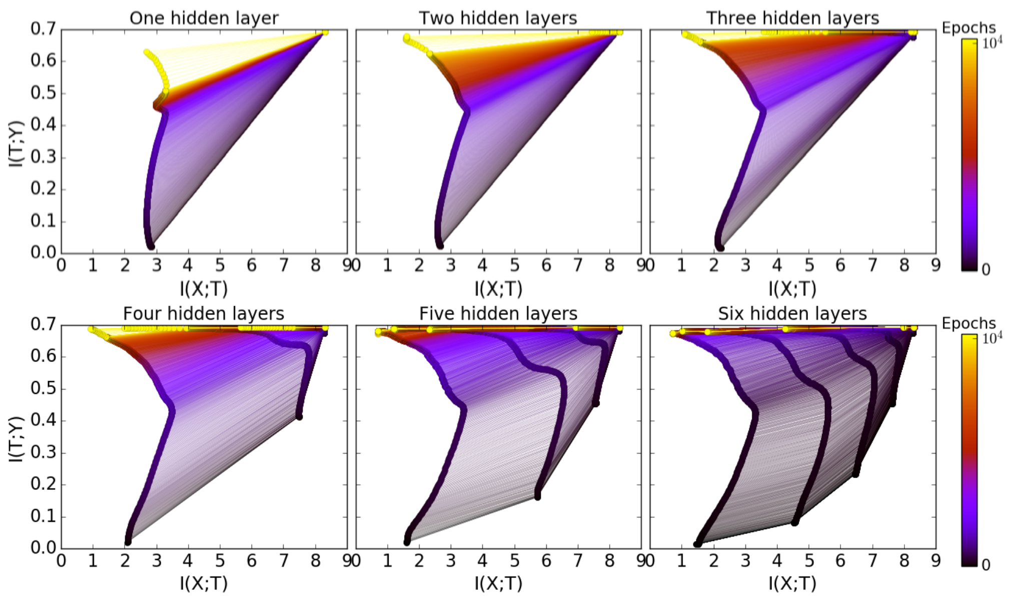

Figure 6: "Information planes" drawn from pairs of MI from adjacent layers. The distribution of the IP has a larger variance for the initial layer. The IP lines show that the differences between $I(X;T)$ and $I(Y;T)$ are minimal in the last layers and last epochs. This confirms that the network is fine-tuning its classification capabilities. T is derived from weights' values using a small random linear addition to the initial variable [44].

In [39] it was observed that during training, neural networks undergo two distinct phases of fitting and compression, with the compression phase crucial for generalization. However, subsequent studies by [40] and [45] challenged these claims, demonstrating that compression does not necessarily correlate with improved generalization and that MI may not be the optimal metric to capture learning dynamics. Although the IB principle suggests that an effective neural network should compress input data into internal representations that retain only the most relevant information to predict output, the evidence supporting a causal link between compression (measured by MI) and generalization has been inconclusive.

Traditionally, MI has been used to quantify the trade-off between input compression and output preservation in neural networks. However, conflicting results on the causal link between MI-based compression and generalization performance have prompted the exploration of alternative measures.

For this reason and because TE can capture nonlinear temporal relationships between variables (which MI does not), we used TE instead of MI as a metric for information compression/preservation [32]. We introduced a novel approach for analyzing and visualizing the information flow within deep neural networks. This method provides fresh insights into the IB principle and IP analysis, with the aim of deepening the understanding of learning dynamics and generalization in neural networks. Our method is an alternative to the MI-based approach in [44].

We calculated TE between adjacent layers of the neural network, overlayed the averaged TE values of each batch in each epoch, and performed IP analysis to reveal the presence of the IB principle. We could experimentally show, for fully connected shallow networks and CNNs, that TE values decrease over training epochs. This decrease can be related to the compression of irrelevant information and aligns with the fitting and compression phases described by the IB theory. The TE values are higher in initial epochs and layers, decreasing as the network abstracts and generalizes features in deeper layers. The observations revealed that TE effectively captures the training dynamics and provides concrete evidence of information compression at the layer level.

Our findings revealed consistent patterns:

- Temporal evolution of TE: TE values are higher during the initial training epochs and gradually decrease as training progresses. This reflects the transition from the fitting phase to the compression phase in the IB framework, as conceptualized by [39]. The decrease in TE indicates that the network is increasingly focusing on relevant features and filtering out irrelevant information. This result was aligned with the findings in [40, 44].

- Layer-wise TE distribution: Higher TE values and variances are observed in the deeper layers of the network, particularly the final fully connected layers in CNNs. This aligns with observations in [40], who noted that significant compression occurs in the later layers of the network. The higher TE in these layers suggests that they are responsible for capturing more abstract representations and that information compression is more pronounced there.

- TE based IP correlation with performance metrics: There is a strong inverse relationship between TE and training accuracy, as well as a close alignment between TE and the loss function. As TE decreases, indicating increased compression, accuracy improves and loss decreases. This supports the notion that efficient information transfer and compression are associated with increased learning performance. IP illustrates each layer's compression proficiency, and TE dynamic visualization (at the layer, epoch, or training batch level) can depict possible hurdles during training.

In summary, TE is a promising tool for analyzing learning dynamics and the relationship between compression and generalization in neural networks. It offers deeper insights than MI by accounting for temporal dependencies. We believe that further developing our TE findings into TE-based loss functions has the potential to enhance the training efficacy of deep neural networks.

4.3 TE in Graph Convolutional Neural Networks

Graph Neural Networks (GNNs) are neural networks that use graph data as inputs. The input features are associated with the values of the nodes, and the data links between the nodes are modeled as edges. Together, these are modeled using different mechanisms and can employ a multitude of tasks, from classification systems to recommendation systems, traffic forecasting, and molecular property predictions. Schematically, a GNN can be simplified to the representation in Fig. 7. When applying operations on a GNN, in particular convolution operations, these use the node relations to select how far from the centered node an operation is applied to the neighboring nodes (Fig. 8). The similarities with the convolutional operators from the classical CNNs can be observed in Fig. 9: the convolutional kernel is applied in the same way as the number of node neighbors are being targeted by the graph convolution operator. Considering the convolutional operator as a filter function applied to graph nodes, we depict in Fig. 10 how the node's features are used to compute the aggregated values to the neighboring nodes.

We integrated TE in a semi-supervised classification method in [33]. Essentially, we improved the Yan *et al.* algorithm [47] to address two challenges in GCNs: *oversmoothing* and the effective use of the relational properties of the nodes, specifically *heterophily* and *homophily*. The latter two negatively affect the generality of the classification algorithms in GCNs. Oversmoothing refers to the phenomenon in which repeated aggregations in GCNs lead to the homogenization of node features, diminishing

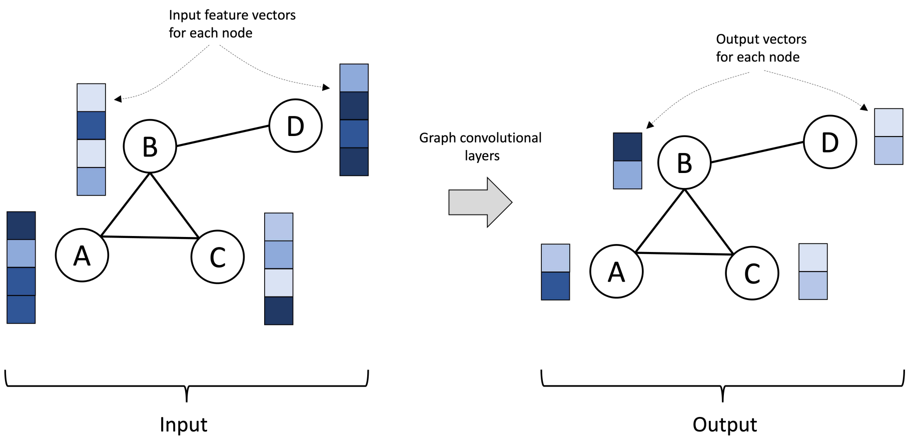

Figure 7: A GCN takes as input a graph together with a set of feature vectors where each node is associated with its own feature vector. The GCN is then composed of a series of graph convolutional layers that iteratively transform the feature vectors at each node. The output is the graph associated with output vectors associated with each node (from: Matthew Bernstein https://mbernste.github.io/posts/gcn/).

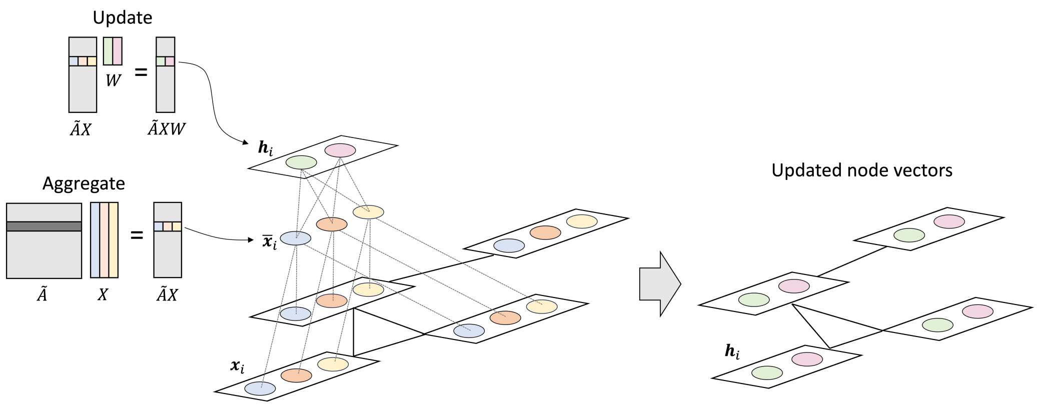

Figure 8: We can visualize the graph convolutional layer at a given node using a network diagram highlighting the neural network architecture (from: Matthew Bernstein https://mbernste.github.io/posts/gcn/).

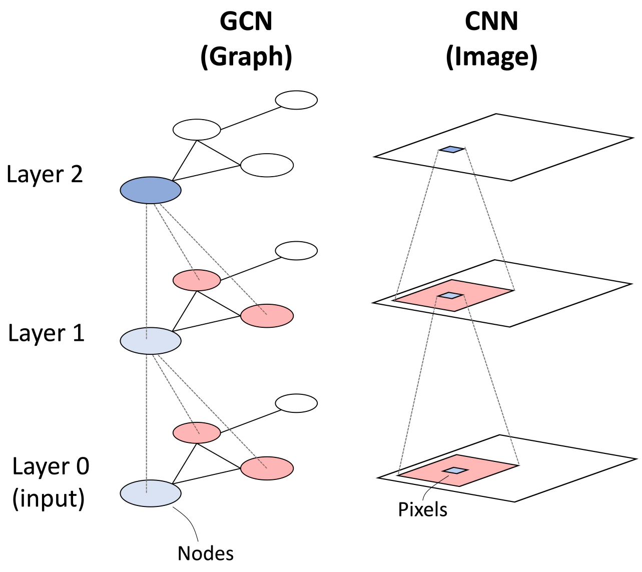

Figure 9: A GCN can be understood as performing a convolution in the same way that traditional CNNs perform a convolution-like operation (from: Matthew Bernstein https://mbernste.github.io/posts/gcn/).

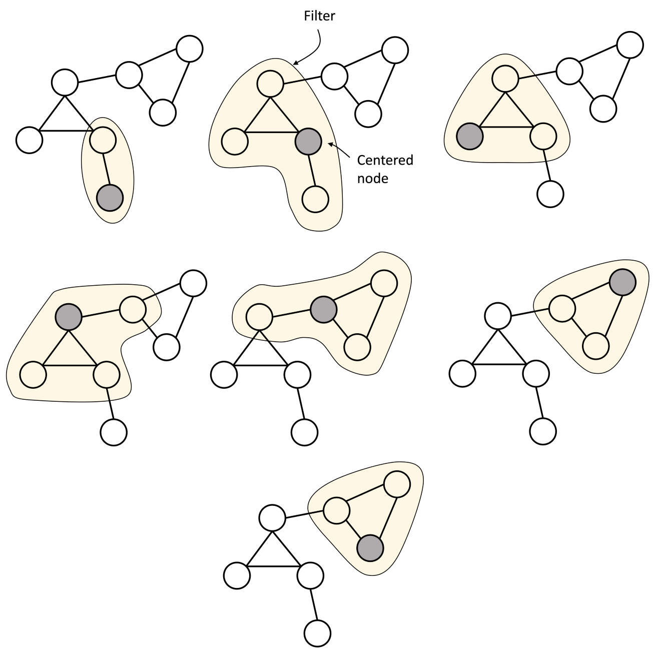

Figure 10: A filter is passed over each node and the values of the neighboring nodes are combined to form the output value at the next layer (from: Matthew Bernstein https://mbernste.github.io/posts/gcn/).

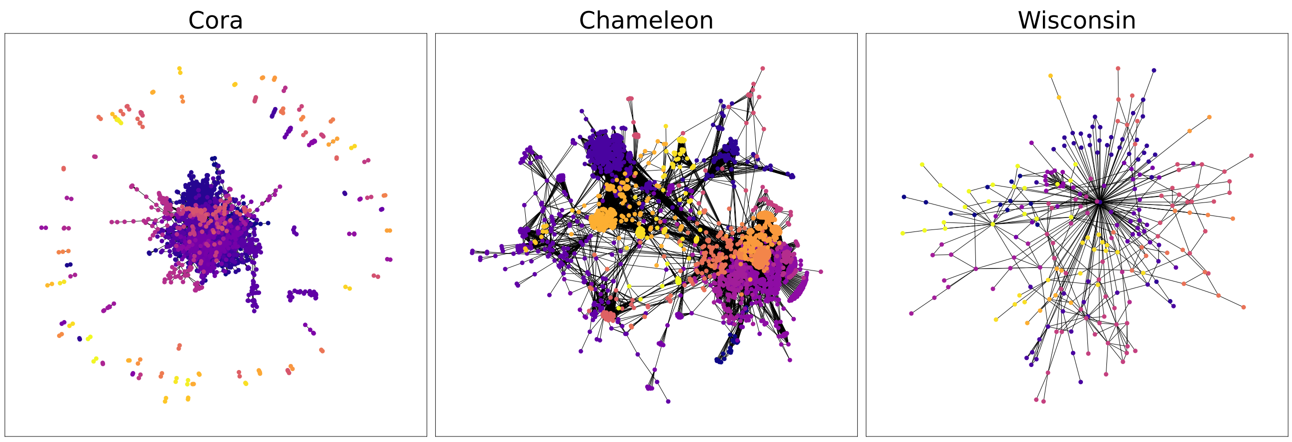

Figure 11: Clustering obtained using Louvain method [46] for three citation network datasets. Colors represent a grouping method based k-nearest neighbor, using the node's internal and external degrees to compute distance metrics. Three different structures from the datasets used in our experiments are illustrated. Observe the completely different organization of the nodes and connections where both homophily and heterophily are present in the same dataset, under different clusters [33].

their discriminative power; this effect is intrinsic to the convolution operations and increases with the number of its applications. Heterophily denotes the tendency of nodes to connect with dissimilar nodes, while homophily refers to connections between similar nodes. Both are significantly impacting GCN performance. The heterogeneity of a graph data structure can be observed in Fig. 11: there is no generic operation that can be easily applied to the entire graph structure to produce consistent results across its clusters, since any operations are limited through their nature and they are inherently relying on the node's relational properties. We motivate the selection of Yan's *et al.* implementation as baseline by the high scores obtained on the citation network datasets, its usage of latest regularization and decay aggregation techniques, its speed, and the fact that it is using a widely adopted technique which weights the inputs along with output aggregation.

In the GGCN algorithm, Yan *et al.*, recalibrated the edge weights using a novel mechanism to calculate the relative degree of a node. The authors then assigned negative signs for heterophilous edges and positive signs for homophilous ones. In our implementation, in addition to the GGCN method, we integrated TE as a control mechanism within the GCN learning process to mitigate these challenges. In the context of GCNs, TE quantifies the information transfer between nodes based on their feature distributions. Our proposed TE-GGCN method modifies the standard GCN architecture by introducing TE-based adjustments after the convolutional layers. Specifically, the method involves the following steps:

1. Calculate the heterophily rate $\mathcal{H}_v$ for each node v, using the formula $\mathcal{H}_v = \frac{1}{|N(v)|}\sum_{u\in N(v)} 1\,(l_u \neq l_v)$, where $N(v)$ is the set of neighbors and l_u is the label of node u.

2. Select nodes with the highest heterophily rates and degrees to compute TE, thus focusing computational resources on the most impactful nodes.

3. Compute TE using node features as input, $TE_{Y\to X}$, where X and Y are feature vectors of pairs of nodes.

4. Adjust the node features $\mathbf{H}_{i,j}$ post-convolution using $\mathbf{H}_{i,j} = \mathbf{H}_{i,j} + max(TE_{Y_j\to X_i})$, thereby enhancing feature differentiation.

In essence, different from the GGCN algorithm, we used TE to compute the similarity of the node values. We used TE to directly modify the features of the GCN nodes. In this way, we controlled the messages exchanged between these nodes and, consequently, the misclassification rate. We computed TE prior to the convolution operations and applied it after the convolution blocks.

To minimize the computational cost of TE, we used the following node selection mechanism (before computing TE): *a)* Select the 5% highest heterophilic nodes; *b)* From these, select 10% from the nodes that have the highest degree. The node selection count was empirically determined after multiple runs while trying to optimally balance computational cost and accuracy.

Experimental evaluations on citation network datasets having various degrees of homophily and heterophily demonstrate that TE-GGCN outperforms the baseline GGCN in classification accuracy. The results are summarized in Tables 1 and 2 and the implementation details can be found in our repository[2].

Our method maintains the original GGCN architecture while effectively addressing oversmoothing and leveraging heterophilic relationships. However, the computational overhead increases with the number of nodes for which TE is computed, posing a trade-off between accuracy and efficiency. As a result, nodes that have many neighbors will require increased computational needs. Despite the added computational cost, the TE-GGCN offers a flexible and generalized approach to improve GCN performance by incorporating TE, suggesting its potential applicability to a wide range of graph-based learning tasks.

Table 1: Datasets characteristics and mean accuracy over 10 runs with ± stdev. The best results are grayed and bolded.

	Texas	Wisconsin	Actor	Squirrel
Hom. level h	0.11	0.21	0.22	0.22
Classes	5	5	5	5
#Nodes	183	251	7,600	5,201
#Edges	295	466	26,752	198,493
TE-GGCN (ours)	**84.86 ± 4.55**	**87.45 ± 3.70**	37.50 ± 1.57	55.04 ± 1.64
GGCN	83.51 ± 3.72	86.47 ± 3.29	**37.56 ± 1.55**	**55.51 ± 2.06**

Table 2: Results continued.

	Chameleon	Cornell	Citeseer	Pubmed	Cora
Hom. level h	0.23	0.3	0.74	0.8	0.81
Classes	5	5	7	3	6
#Nodes	2,277	183	3,327	19,717	2,708
#Edges	31,421	280	4,676	44,327	5,278
TE-GGCN (ours)	**71.14 ± 1.84**	**85.68 ± 6.63**	**77.14 ± 1.45**	89.08 ± 0.37	**87.95 ± 1.05**
GGCN	70.57 ± 1.84	84.32 ± 6.63	76.51 ± 1.45	**89.12 ± 0.32**	84.32 ± 1.05

5 Conclusions

This study underscores TE as a tool to improve the efficiency, interpretability, and analysis of deep learning models. We have demonstrated improvements in training performance for CNNs and GCNs, highlighting its ability to capture causal information flow. The application of TE not only reduces training time, but also aids in identifying key causal features, providing a pathway toward more explainable artificial intelligence.

Despite its promise, the integration of TE in neural network optimization faces challenges such as computational overhead and the difficulty of reliably estimating entropy measures for large-scale networks. However, our findings illustrate that targeted applications, such as using TE to analyze layer-level information dynamics or mitigate oversmoothing in GCNs, can strike an effective balance between computational cost and model performance.

Future work should focus on developing scalable methods for TE computation and exploring its integration with other interpretability frameworks. By addressing these challenges, TE could serve as a cornerstone to advance efficient and interpretable deep learning systems, bridging the gap between performance and understanding.

The second law of thermodynamics says that in an isolated system, entropy always increases. This principle explains, for example, why an egg cannot be unscrambled. TE can be conceptually linked

[2]https://github.com/avmoldovan/Heterophily_and_oversmoothing-forked/

to thermodynamics by interpreting the direction of information flow as a proxy for the direction of entropy change between two interacting systems [48].

TE applied to neural networks has certain limitations. TE can measure the information flow between multiple variables with time-series activities in discrete time. However, neural networks involve billions of information flow paths that lead to a decision, and these paths cannot be easily reverse-engineered simply by observing or measuring the neural computations. Consequently, it is not possible to mathematically dissect a machine's decision to uncover the exact paths taken by the neural network to reach that conclusion.

Additionally, entropy, including TE, cannot be reversed in a closed system because entropy always increases over time. It is more accurate to consider the entropy as being generated rather than reduced. As a result, such an information-theoretical lossy method cannot serve as a universal tool for reverse engineering a neural model and interpreting / explaining its decisions with 100% precision.

References

[1] D. Hume, "Of the idea of necessary connection," in *A Treatise of Human Nature.* John Noon, 1739, ch. 14.

[2] T. Schreiber, "Measuring information transfer," *Phys. Rev. Lett.*, vol. 85, pp. 461–464, Jul 2000.

[3] T. Marwala, *Causality, correlation and artificial intelligence for rational decision making.* World Scientific, 2015.

[4] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. New York, NY, USA: Cambridge University Press, 2009.

[5] J. S. Mill, *A System of Logic, Ratiocinative and Inductive: Being a Connected View of the Principles of Evidence and the Methods of Scientific Investigation.* New York: Harper & Brothers, 1882.

[6] W. Salmon, *Scientific Explanation and the Causal Structure of the World.* Princeton University Press, 2020. [Online]. Available: https://books.google.com/books?id=AET_DwAAQBAJ

[7] R. Andonie and B. Kovalerchuk, "Neural networks for data mining: Constrains and open problems," in *Proceedings of the 12th European Symposium on Artificial Neural Networks (ESANN'2004)*, M. Verleysen, Ed., 2004, pp. 449–458.

[8] B. Kovalerchuk, K. Nazemi, R. Andonie, N. Datia, and E. Banissi, *Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery.* Springer, 2022.

[9] B. Kovalerchuk, R. Andonie, N. Datia, K. Nazemi, and E. Banissi, "Visual knowledge discovery with artificial intelligence: Challenges and future directions," in *Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery.* Cham: Springer International Publishing, 2022, pp. 1–27.

[10] B. Kovalerchuk, K. Nazemi, R. Andonie, N. Datia, and E. Banissi, *Artificial Intelligence and Visualization: Advancing Visual Knowledge Discovery.* Springer Cham, 2024.

[11] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization," *CoRR*, vol. abs/1610.02391, 2016. [Online]. Available: http://arxiv.org/abs/1610.02391

[12] E. Borgonovo, E. Plischke, and G. Rabitti, "The many shapley values for explainable artificial intelligence: A sensitivity analysis perspective," *European Journal of Operational Research*, 2024.

[13] G. Morales and J. Sheppard, "Counterfactual explanations of neural network-generated response curves," in *2023 International Joint Conference on Neural Networks (IJCNN).* IEEE, 2023, pp. 01–08.

[14] A. Behnam and B. Wang, "Graph neural network causal explanation via neural causal models," *arXiv preprint arXiv:2407.09378*, 2024.

[15] B. Muşat and R. Andonie, "Semiotic aggregation in deep learning," *Entropy*, vol. 22, no. 12, p. 1365, 2020.

[16] R. Andonie and B. Musat, "Signs and supersigns in deep learning," *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL*, vol. 19, no. 1, 2024.

[17] O. Kwon and J.-S. Yang, "Information flow between stock indices," *EPL (Europhysics Letters)*, vol. 82, no. 6, p. 68003, 2008. [Online]. Available: http://stacks.iop.org/0295-5075/82/i=6/a=68003

[18] T. Bossomaier, L. Barnett, M. Harré, and J. T. Lizier, *An Introduction to Transfer Entropy. Information Flow in Complex Systems.* Springer, 2016.

[19] L. Barnett, A. B. Barrett, and A. K. Seth, "Granger causality and transfer entropy are equivalent for gaussian variables," *Phys. Rev. Lett.*, vol. 103, p. 238701, Dec 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.238701

[20] K. Hlaváčková-Schindler, "Equivalence of granger causality and transfer entropy: A generalization," *Appl. Math. Sci.*, vol. 5, no. 73, pp. 3637–3648, 2011.

[21] P. Wollstadt, M. Martínez-Zarzuela, R. Vicente, F. J. Díaz-Pernas, and M. Wibral, "Efficient transfer entropy analysis of non-stationary neural time series," *PloS one*, vol. 9, no. 7, p. e102833, 2014.

[22] P. Bonetti, A. M. Metelli, and M. Restelli, "Causal feature selection via transfer entropy," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024, pp. 1–10.

[23] X. Li and G. Tang, "Multivariate sequence prediction for graph convolutional networks based on esmd and transfer entropy," *Multimedia Tools and Applications*, pp. 1–19, 2024.

[24] J. Zhang, J. Cao, W. Huang, X. Shi, and X. Zhou, "Rutting prediction and analysis of influence factors based on multivariate transfer entropy and graph neural networks," *Neural Networks*, vol. 157, pp. 26–38, 2023.

[25] H. Xu, Y. Huang, Z. Duan, J. Feng, and P. Song, "Multivariate time series forecasting based on causal inference with transfer entropy and graph neural network," *arXiv preprint arXiv:2005.01185*, pp. 1–9, 2020.

[26] S. Kim, S. Ku, W. Chang, and J. W. Song, "Predicting the direction of us stock prices using effective transfer entropy and machine learning techniques," *IEEE Access*, vol. 8, 2020.

[27] H. Wang, D. Li, H. Zhou, C. Guo, and Y. Liu, "Transfer entropy and lstm deep learning-based faulty sensor data recovery method for building air-conditioning systems," *Journal of Building Engineering*, p. 111307, 2024.

[28] O. Obst, J. Boedecker, and M. Asada, "Improving recurrent neural network performance using transfer entropy," in *Proceedings of the 17th International Conference on Neural Information Processing: Models and Applications - Volume Part II*, ser. ICONIP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 193–200. [Online]. Available: http://dl.acm.org/citation.cfm?id=1939751.1939778

[29] S. Herzog, C. Tetzlaff, and F. Wörgötter, "Transfer entropy-based feedback improves performance in artificial neural networks," *CoRR*, vol. abs/1706.04265, 2017. [Online]. Available: http://arxiv.org/abs/1706.04265

[30] A. Moldovan, A. Caţaron, and R. Andonie, "Learning in feedforward neural networks accelerated by transfer entropy," *Entropy*, vol. 22, no. 1, p. 102, 2020.

[31] ——, “Learning in convolutional neural networks accelerated by transfer entropy,” *Entropy*, vol. 23, no. 9, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/9/1218

[32] A. Moldovan, A. Caţaron, and R. Andonie, “Information plane analysis visualization in deep learning via transfer entropy,” in *2023 27th International Conference Information Visualisation (IV)*, 2023, pp. 278–285.

[33] ——, “Transfer entropy in graph convolutional neural networks,” in *2024 28th International Conference Information Visualisation (IV)*, 2024, pp. 278–285.

[34] R. Féraud and F. Clérot, “A methodology to explain neural network classification,” *Neural Networks*, vol. 15, no. 2, pp. 237 – 246, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608001001277

[35] S. Ito, “Backward transfer entropy: Informational measure for detecting hidden markov models and its interpretations in thermodynamics, gambling and causality,” *Scientific reports*, vol. 6, no. 1, p. 36831, 2016.

[36] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: primal estimated sub-gradient solver for svm,” *Mathematical Programming*, vol. 127, pp. 3–30, 2011.

[37] A. Caţaron and R. Andonie, “Transfer information energy: A quantitative indicator of information transfer between time series,” *Entropy*, vol. 20, no. 5, 2018. [Online]. Available: https://www.mdpi.com/1099-4300/20/5/323

[38] N. Tishby and N. Zaslavsky, “Deep learning and the information bottleneck principle,” in *2015 IEEE Information Theory Workshop (ITW)*, 2015, pp. 1–5.

[39] R. Shwartz-Ziv and N. Tishby, “Opening the black box of deep neural networks via information,” *CoRR*, vol. abs/1703.00810, 2017. [Online]. Available: http://arxiv.org/abs/1703.00810

[40] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, “On the information bottleneck theory of deep learning,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124020, dec 2019. [Online]. Available: https://dx.doi.org/10.1088/1742-5468/ab3985

[41] D. McFadden, “Conditional logit analysis of qualitative choice behavior,” in *Frontiers in Econometrics*, P. Zarembka, Ed. Academic Press, 1972, pp. 105–142.

[42] H. Haken and J. Portugali, *Information adaptation: the interplay between Shannon information and semantic information in cognition.* Springer, 2014.

[43] O. Shamir, S. Sabato, and N. Tishby, “Learning and generalization with the information bottleneck,” *Theoretical Computer Science*, vol. 411, no. 29-30, pp. 2696–2711, 2010.

[44] R. Shwartz Ziv and Y. LeCun, “To compress or not to compress—self-supervised learning and information theory: A review,” *Entropy*, vol. 26, no. 3, 2024. [Online]. Available: https://www.mdpi.com/1099-4300/26/3/252

[45] B. C. Geiger, “On information plane analyses of neural network classifiers—a review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7039–7051, 2022.

[46] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1088/1742-5468/2008/10/P10008

[47] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, and D. Koutra, “Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks,” in *IEEE International Conference on Data Mining (ICDM)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2022, pp. 1287–1292.

[48] M. Prokopenko, J. T. Lizier, and D. C. Price, “On thermodynamic interpretation of transfer entropy,” *Entropy*, vol. 15, no. 2, pp. 524–543, 2013. [Online]. Available: https://www.mdpi.com/1099-4300/15/2/524

Cite this paper as:

Andonie, R.; Caţaron, A.; Moldovan, A. (2025). Transfer Entropy in Deep Neural Networks, *International Journal of Computers Communications & Control*, 20(1), 6904, 2025.
https://doi.org/10.15837/ijccc.2025.1.6904