Productive Simplification: Complexity Production by Multiple Reductive Simplification Procedures for Interpreting Multi-Layered Neural Networks

Ryotaro Kamimura
Tokai University
Kitakaname, Hiratsuka, Kanagawa, Japan
e-mail: ryotarokami@gmail.com

***Abstract*—The present paper aims to show that there exists a strong simplification force inside neural networks. However, it is naturally necessary to introduce some complexity into connection weights to make learning possible. This means that simplification must be violated to achieve appropriate learning. To address this issue, we assume that the simplification principle is always maintained, and the only way it can be (seemingly) violated is by combining several different types of simplification procedures. This combination, which adds some complexity, can be called "productive simplification," meaning that the added complexity arises as a result of the interaction among many different simplification procedures. The simplification principle still holds, but it can be modulated by various types of these very simplification processes. To demonstrate productive simplification, we use two types of simplification: local and distributed simplification. Local simplification aims to reduce the number of components, while distributed simplification attempts to simplify the roles of each component as much as possible. These two types of simplification are unified in productive simplification, and in practice, some complexity can thereby be introduced. Productive simplification was applied to an artificial dataset that incorporated both linear and non-linear relations. The results show that the unification of local and distributed simplification appeared in the form of a phase transition, where generalization performance changed significantly in this transition phase. This phase transition demonstrates the existence of a productive unification of different types of simplification, ultimately strengthening the simplification force inside neural networks.**

Keywords-simplification; productive; reductive; local; distributed; diachronic

I. Introduction

This paper aims to demonstrate that the simplification force should be observed at all levels of neural learning, even in situations where some complexity in connection weights is required [1]. This apparent contradiction can be resolved by assuming that different types of simplification forces are combined to produce the effect of complexity in connection weights, though only seemingly. This means that complexity is actually the result of many simplification forces operating inside neural networks.

A. Simplification Principle

The present paper aims to show that there exists a strong simplification force inside neural networks. These simplification forces should be observed internally and in many different ways. Internal simplification lies in the existence of self-organizing forces that simplify internal network configurations as much as possible, and this can be realized even without information from external inputs. Naturally, this is too idealized, and in practice it must take input information into account. Thus, it becomes necessary to add some complexity to connection weights during learning, which seems contradictory to the hypothesis of simplicity. This contradiction can be resolved by supposing that simplification can take many different forms. Although these forms may appear contradictory to each other, their interaction can be used to introduce complexity into the connection weights. This means that complexity is in fact the outcome of the simplification principle, which operates universally in learning.

B. Reductive and Productive Simplification

As mentioned above, it is often necessary to introduce some complexity into connection weights to make learning possible. This implies that simplification must be weakened or seemingly violated for appropriate learning. To address this issue, we assume that the simplification principle always holds, and that the only way it can be (seemingly) violated is by combining several different simplification procedures. This combination, which adds some complexity, can be called "productive simplification," meaning that complexity arises as a consequence of the coexistence of many different simplification procedures. In other words, the simplification principle continues to operate, but it can be modulated by employing different types of simplification.

To emphasize the importance of productive simplification, we distinguish between two types of simplification: reductive and productive. Reductive simplification refers to the usual meaning of simplification: reducing the complexity of network configurations as much as possible, as many previous learning procedures for improving generalization and interpretation have done. In contrast, productive simplification aims to add some complexity into network configurations while still adhering to the simplification principle. This means that in each computational step, the simplification principle must be strictly followed, but the combination of these procedures can result in the addition of complexity necessary for learning. In this sense, productive simplification shows that even when complexity is introduced, the simplification principle continues to prevail.

C. Local and Distributed Simplification

The general simplification principle governs learning processes, but in practice, many different types of simplification forces are likely to be present. This arises from attempts

to achieve simplified configurations from as many different perspectives as possible. Among these, we consider two types of simplification forces as a first approximation in this paper: local and distributed simplification. In general, simplification is assumed to reduce the total strength of connection weights. Local simplification seeks to reduce the number of strong connection weights as much as possible. In this case, only one weight retains significant strength, while the others are forced to become very small. This ultimately reduces the total strength of the connection weights, with only one weight playing an essential role in information processing. On the other hand, distributed simplification seeks to reduce the strength of all connection weights as much as possible. In this case, all weights retain small, evenly distributed strength values. When a small number of weights are known to play important roles, local simplification is appropriate. Conversely, when no prior knowledge is available about which weights are more important, distributed simplification should be used. In productive simplification, these two types of forces are combined, producing an effect that weakens the total simplification force. It should be noted that even this weaker form of simplification still arises from the combination of two distinct simplification strategies.

D. Main Contributions

The main contributions of this paper can be summarized as follows:

- The present paper aims to clarify the simplification forces hidden in multi-layered neural networks.
- Even in the process of adding complexity during learning, the simplification principle continues to operate internally. This can be realized through productive simplification, which combines multiple reductive simplification procedures.
- We propose two types of simplification forces: local and distributed simplification. These two types should be unified, and through their unification, some complexity can be introduced while still adhering to the simplification principle.
- We conducted experiments to demonstrate how this unification of simplification types occurs. The results show that, initially, local simplification dominates, and then a phase transition occurs to unify local and distributed simplification.
- The results indicate that simplification forces exist in neural networks. The main mechanism lies in strengthening simplification through the unification of different simplification forces. This suggests that behind seemingly complexity-adding operations, strong simplification forces are always at work.

E. Paper Outline and Organization

In Section II, we show that our method of productive simplification can be regarded as one of the first attempts to transform conventional reductive simplification procedures into a productive and unified approach. In Section III, we introduce local and distributed potentiality, which are consumed to achieve simplicity. The unification of these two types of potentiality is realized through diachronic potentiality, where unification is controlled in a time-dependent manner. In addition, we provide a brief explanation of computational methods such as compression and ratio potentiality for comparison. In Section IV, we present experimental results obtained using productive potentiality. We first show that simplification can be modulated by introducing productive potentiality control. Then, during learning, a kind of phase transition occurs, demonstrating a compromise between local and distributed simplification. Finally, simplification tends to make networks as linear as possible despite the presence of many nonlinear properties in neural networks. In Section V, we summarize the paper and suggest future work, such as exploring the relationship between productive simplification and the problem of simplicity versus complexity in neuroscience.

II. Related Work

We have stated the existence of multiple types of simplification procedures in the complexity addition processes of productive learning. Almost all conventional methods, however, seem to be confined to reductive simplification. Here, we describe four conventional reductive simplification procedures, emphasizing that all of them aim to address reductive simplicity without sufficient consideration of the productive property. The reductive procedures must be complemented by productive simplification, whereby certain complexities are reintroduced as a result of the simplification itself.

A. Prototype Simplification

The prototype approach has been one of the major simplification procedures in neural networks. Here, we use the prototype to describe the simplest form achievable through the simplification forces inherent in the network. Prototype-based approaches have been studied extensively since the early stages of learning, under names such as vector quantization, competitive learning, and self-organizing maps [2]–[7]. These methods aim to identify a small number of representative vectors for many input patterns. In recent deep learning research, prototype learning has gained renewed attention [8], as the volume of input patterns to be processed has grown larger and increasingly heterogeneous. Simplifying these complex and heterogeneous patterns has become urgent, giving rise to methods like one-shot and few-shot learning, which represent many inputs using a few representative patterns. Similarly, zero-shot learning [9] has been developed to incorporate the abstract and semantic properties of input patterns. These methods exemplify a typical reductive simplification procedure, representing a large number of input patterns with a smaller set of representative inputs and more abstract features. In contrast, our approach focuses on the network itself, aiming to determine which network configuration should be organized when all network components are given before learning. Furthermore, we attempt to unify multiple types of simplification to introduce complexity.

B. Interpretable Simplification

Second, most interpretation methods can be classified as reductive simplification. Among these, a popular approach is localized interpretation, which focuses on specific instances rather than the overall inference mechanism. Unlike global interpretation methods [10]–[12], localized interpretation has proven effective in practical applications due to its simplicity and the urgent demand for explainability. Linear and local simplifications have been widely used in interpretation methods, replacing complex non-linear models with corresponding linear models [13]–[15]. These simplified local models aim to interpret surface networks, which are diverse and heterogeneous. However, because the main characteristics of surface networks lie in their diversity, such diversity hinders interpretation. It is therefore necessary to seek a simpler and unified prototype for interpretation. Our approach seeks to uncover the prototype hidden deep within surface models. Surface models are produced through transformational rules and may be too complex to understand, but the underlying prototype is expected to be much easier to interpret.

C. Distilling Simplification

Third, network compression simplifies neural networks by replacing complex networks with smaller ones [16]–[20]. This type of simplification has become increasingly necessary as networks grow larger to process more input patterns. However, this approach does not allow us to interpret how all components in such networks operate to produce outputs. Most compression methods represent typical external and reductive simplification, replacing the original network with smaller and often unrelated networks. These methods attempt to interpret the smaller "student" networks under the assumption that their inference mechanism is similar to the original network's. However, this assumption may not always hold true. In contrast, the productive simplification we propose compresses the original multi-layered neural network directly, retaining the original network's information. Thus, interpreting and explaining the compressed networks is more directly connected to the original multi-layered networks. Conventional compression is not productive in the sense that it cannot directly relate the original networks to compressed ones for interpretation.

D. Mutual Information-Theoretic Simplification

Fourth, information-theoretic methods also relate to network simplification, though in more abstract ways. Multiple network configurations can be represented by simpler and more abstract information content, and the objective of learning can be considered as necessary information acquisition. In particular, mutual information has played a significant role in neural networks. Well-known examples include maximum mutual information preservation [21]–[24], which aims to retain as much relevant information as possible, and the information bottleneck method [25]–[28], which seeks to maximize necessary information while minimizing unnecessary information.

Despite their utility, mutual information-based methods face difficulties, such as computational complexity and ambiguous interpretations of the results. These issues arise because mutual information involves contradictory operations: entropy maximization and conditional entropy minimization. Recent information bottleneck methods introduce additional mutual information computations to balance compression and relevant information preservation [29], but computational complexity remains a challenge. The main procedure is to reduce information on inputs while retaining appropriate information about targets. This is similar to distributed simplification.

Mutual information maximization can, in fact, be viewed as addressing both local and distributed simplification, as is the case with the productive simplification in this paper. We believe it can be applied to the problem of complexity and simplicity in neuroscience [30]. However, due to the computational complexity inherent to mutual information, its productive property of simplification has not yet been realized [24][31][32]. The present attempt at productive simplification aims to take the first step toward realizing the actual potential of mutual information in neural networks.

III. Theory and Computational Methods

Here, we introduce how to compute two types of simplification: local and distributed. Then, we briefly explain how to combine these simplification forces to add complexity. In addition, we give a short explanation for better understanding of the experimental results and how to compress a neural network into the corresponding prototype. Since our method is an extension of entropy, we also compare it with conventional entropic methods, showing how our approach can enhance learning characteristics in a simplified manner.

A. Theoretical Procedures

1) Simplification Principle: The present paper aims to show how to use the simplification principle to add some complexity to network configurations, which is necessary for realizing appropriate learning processes. Figure 1 shows a process of productive simplification, transforming a network from a surface network into a simpler one. For the surface network, two types of reductive simplification are applied. Local simplification aims to reduce the number of important connection weights as much as possible, whereas distributed simplification aims to reduce the strength of all connection weights. Both procedures are unified in productive simplification, producing an intermediate multi-layered neural network. In addition, this productive simplification is adjusted in a diachronic manner, meaning that the ratio of local to distributed simplification is controlled throughout the course of learning. Finally, the intermediate network is compressed into the simplest form, namely, a prototype network.

2) Simplification Potentiality Consumption: First, we introduce the basic simplification indexes used in this paper. The overall measure of simplification is defined as the strength of absolute connection weights. When this simplification potentiality, or the strength of absolute weights, becomes smaller, the network becomes simpler by definition. For simplicity, we consider only one layer, from the nth layer to the $n+1$th layer,

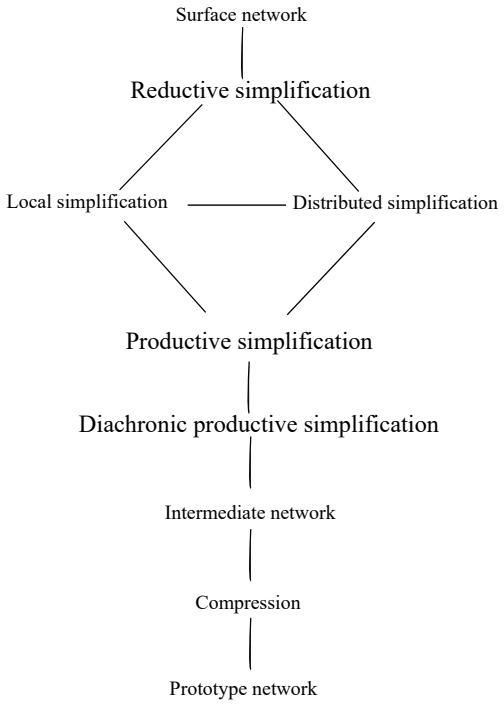


Figure 1. Diachronic productive simplification from a surface network to a prototype network by unifying local and distributed reductive simplification.

denoted as $(n, n+1)$ in Figure 2. The individual simplification potentiality for $(n, n+1)$ is computed as:

$$u_{jk}^{(n)} = \mid w_{jk}^{(n)} \mid, \tag{1}$$

where the notation for the layer is simplified from $(n, n+1)$ to (n), and all potentialities (absolute weights) are assumed to be greater than zero. The simplification potentiality for one layer is obtained by summing all individual simplification potentialities, and by summing across all layers, we obtain the total potentiality:

$$U = \sum_{njk} u_{jk}^{(n)}. \tag{2}$$

As this potentiality becomes larger, the network has a higher capacity to be simplified. The simplification potentiality should be reduced or consumed to achieve the simplest network.

3) Local Potentiality Consumption: As mentioned above, simplification potentiality must generally be reduced or consumed. However, there are several possible ways to achieve

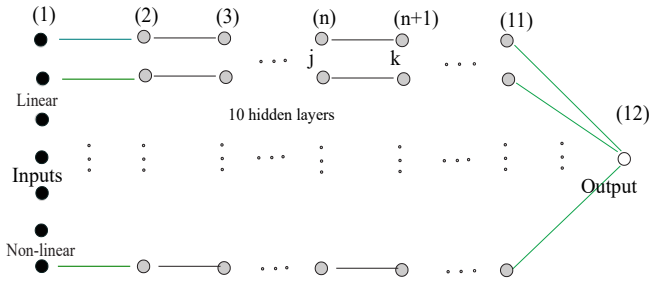


Figure 2. A network architecture with ten hidden layers.

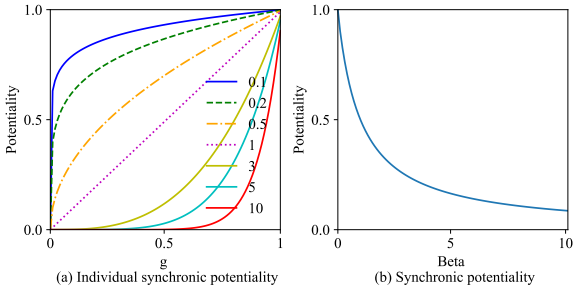


Figure 3. Individual local potentiality (a) and local potentiality (b) as a function of the parameter β_{loc}. All potentialities are normalized for comparison.

this reduction. One popular method can be called "local simplification potentiality." Local simplification potentiality, or simply local potentiality, aims to reduce the strength of all connection weights except a small number of specific ones. Because the majority of weights become smaller, the total simplification potentiality decreases, even though a small number of stronger weights remain.

Now, by normalizing the individual potentiality by its maximum value, we define the relative local potentiality as:

$$g_{jk}^{(n)} = \left[\frac{u_{jk}^{(n)}}{\max_{j'k'} u_{j'k'}^{(n)}} \right]^{\beta_{loc}}, \tag{3}$$

where β_{loc} is a parameter controlling the strength of local potentiality. In particular, when $\beta_{loc} = 1$, the potentiality corresponds to the base case. By summing across all layers, we obtain the local potentiality:

$$G = \sum_{njk} \left[\frac{u_{jk}^{(n)}}{\max_{j'k'} u_{j'k'}^{(n)}} \right]^{\beta_{loc}}. \tag{4}$$

Figure 3(a) illustrates the individual local potentiality for different parameter values. As the parameter β_{loc} increases, the majority of connection weights are forced to decrease, thereby reducing the potentiality. Figure 3(b) shows the local potentiality as a function of β_{loc}. As explained above, the local potentiality decreases gradually as the parameter increases.

Using this local potentiality, the new weights at the $(t+1)$th learning step are obtained by multiplying the weights at the tth step by the corresponding potentiality:

$$w_{jk}^{(n)}(t+1) = g_{jk}^{(n)} w_{jk}^{(n)}(t). \tag{5}$$

4) Distributed Potentiality Consumption: The distributed potentiality is the inverse of the local potentiality and aims to make the strength of all connection weights as small as possible. By normalizing the inverse of individual potentiality by the maximum value, we can define the relative distributed potentiality as:

$$h_{jk}^{(n)} = \left[\frac{\max_{j'k'} u_{j'k'}^{(n)} - u_{jk}^{(n)}}{\max_{j'k'} u_{j'k'}^{(n)}} \right]^{\beta_{dis}}, \tag{6}$$

where β_{dis} is a parameter controlling the strength of the potentiality. By summing all the relative potentialities, we obtain the final form of distributed potentiality:

$$H = \sum_{njk} \left[\frac{\max_{j'k'} u^{(n)}_{j'k'} - u^{(n)}_{jk}}{\max_{j'k'} u^{(n)}_{j'k'}} \right]^{\beta_{dis}}. \quad (7)$$

Using this distributed potentiality, new weights at the $(t+1)$th learning step are obtained by multiplying the weights at the tth step by the corresponding potentiality:

$$w^{(n)}_{jk}(t+1) = h^{(n)}_{jk} w^{(n)}_{jk}(t). \quad (8)$$

Finally, we note that the distributed potentiality is simply the inverse of local potentiality for the base case ($\beta = 1$). In this case, it can be obtained by taking the inverse of the local potentiality.

5) Productive Potentiality Consumption: After defining the local and distributed potentiality, we need to combine these two types to produce complexity, namely, productive potentiality control. We combine the two types of potentiality as:

$$P = \alpha G + \bar{\alpha} H, \quad (9)$$

where the parameter α ranges between zero and one and is introduced to control the ratio of local to distributed potentiality. Note that $\bar{\alpha}$ corresponds to $1 - \alpha$. The unification of local and distributed simplification is simply the sum of the two potentialities when treated synchronically. However, the process of unification must be controlled more subtly to achieve better generalization with simplification, since generalization is considered the most important factor in neural networks.

6) Diachronic Productive Potentiality Consumption: This unification can be realized by defining a new type of potentiality called "diachronic". The diachronic potentiality considers the entire sequence of learning steps. The individual diachronic potentiality at the tth learning step (epoch) is defined solely based on the time step t:

$$v_t = t. \quad (10)$$

The individual diachronic potentiality is then defined as

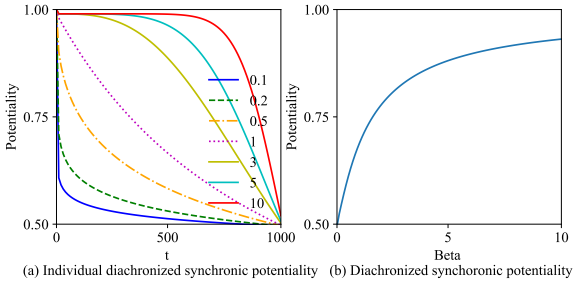
$$z_t = \left[\frac{v_t}{\max_{t'} v_{t'}} \right]^{\beta_{dch}}. \quad (11)$$

We invert this original diachronic potentiality to obtain a distributed type of simplification. The new individual diachronic potentiality is obtained by inverting the original:

$$\bar{z}_t = 1 - z_t. \quad (12)$$

This equation is used to ensure that the sum of the two types of potentiality equals one. In addition, the local potentiality is used at the beginning of learning, followed by the introduction of distributed potentiality. The diachronic potentiality is then used to combine the local and distributed potentiality as

$$w^{(n)}_{jk}(t+1) = \left(\beta_t g^{(n)}_{jk} + \bar{\beta}_t h^{(n)}_{jk} \right) w^{(n)}_{jk}(t), \quad (13)$$



(a) Individual diachronized synchronic potentiality (b) Diachronized synchoronic potentiality

Figure 4. Individual diachronic potentiality (a) and diachronic potentiality (b) as a function of the parameter β_{dia}.

where $\beta = \bar{z}$ and $\bar{\beta} = z$. The effect of diachronic potentiality increases as learning progresses. As the diachronic parameter β_{dch} increases, the effect of local potentiality becomes increasingly mixed with the effect of distributed potentiality.

To explain how diachronic potentiality controls local and distributed potentiality, Figure 4(a) illustrates the diachronic potentiality for different parameter values. As the parameter β_{dia} increases, the majority of individual potentialities increase. This means that the local potentiality becomes more effective throughout learning as the parameter increases. Figure 4(b) shows the inverse of diachronic potentiality as a function of β_{dia}. As explained, the inverse diachronic potentiality increases gradually with increasing parameter strength. By using diachronic potentiality, local potentiality is applied first, followed by the introduction of distributed potentiality. As the parameter increases, the local potentiality remains effective for a longer period. This sequential operation improves the generalization performance of the final neural network.

B. Computational Procedures

1) Compression: To obtain an estimated prototype, we must compress the original multi-layered neural network into the corresponding simplest network without hidden layers. This assumes that all activation functions are linear, although they are actually non-linear. Our simplicity principle aims to make activation functions as linear as possible. If the compressed networks differ from actual linear networks, these differences can be used to specify the effects of non-linearity in neural networks.

Suppose we attempt to compress a twelve-layered neural network, including the input and output layers in Figure 2, into a network without hidden layers. The layers are numbered from No. 1 (input) to No. 12 (output), with ten hidden layers in between. This configuration was used in our experiments, described in the next section. We begin the first compression by:

$$w^{(1,3)}_{ik} = \sum_j w^{(1,2)}_{ij} w^{(2,3)}_{jk}, \quad (14)$$

where layer (1,2) is combined with layer (2,3) to produce the compressed layer (1,3). Repeating this process, we obtain the compressed weights connecting the first and eleventh layers,

denoted $w_{im}^{(1,11)}$. Using these weights, we finally obtain the fully compressed weights for (1,12):

$$w_i^{(1,12)} = \sum_m w_{im}^{(1,11)} w_m^{(11,12)}, \tag{15}$$

where (1,12) indicates that the twelve-layered network is compressed into a two-layered one (1,12).

This compression method differs from conventional approaches [33][34] because it aims to interpret internal representations as much as possible. Our method seeks to understand the inference mechanism of neural networks while preserving the original information in a simplified form.

2) Ratio Potentiality: In addition to the standard potentialities explained above, we introduce the ratio potentiality for comparing estimated and supposed prototypes. Using a twelve-layered network for simplicity, the ratio of compressed individual potentiality of the compressed network to the supposed individual potentiality of the prototype network can be computed by:

$$r_i^{(1,12)} = \frac{g_i^{(1,12)}}{g_i^{(1,2)}}. \tag{16}$$

Here, (1,12) and (1,2) denote the estimated and supposed networks, respectively. For this ratio r_i, we compute the ratio potentiality:

$$v_i^{(1,12)} = \frac{r_i^{(1,12)}}{\max_{i'} r_{i'}^{(1,12)}}. \tag{17}$$

Then,

$$V^{(1,12)} = \sum_i v_i^{(1,12)}. \tag{18}$$

When all ratio potentialities are equal, the ratio potentiality is larger. If only one ratio potentiality is larger than the others, the ratio potentiality is smaller. This metric can be used to assess the similarity between compressed and prototype networks.

3) Entropy and Divergence: Potentiality has been introduced to simplify the entropy function, and it is necessary to compare potentiality with the corresponding entropy. Our entropy and related divergence are defined as follows. The relative potentiality for the supposed prototype network is computed as

$$q_i^{(1,2)} = \frac{u_i^{(1,2)}}{\sum_{i'} u_{i'}^{(1,2)}}. \tag{19}$$

Next, the relative potentiality for the compressed network is

$$p_i^{(1,12)} = \frac{u_i^{(1,12)}}{\sum_{i'} u_{i'}^{(1,12)}}. \tag{20}$$

Entropy is defined as

$$S^{(1,12)} = -\sum_i p_i^{(1,12)} \log\ p_i^{(1,12)}. \tag{21}$$

Entropy decreases when only one weight becomes large while the others are small. Conversely, when entropy is large, all weights are equal. This behavior is similar to the property of potentiality used in this paper. Divergence, as a complementary measure, is computed as

$$D^{(1,12)} = \sum_i p_i^{(1,12)} \log\ \frac{p_i^{(1,12)}}{q_i^{(1,2)}}. \tag{22}$$

Divergence corresponds to ratio potentiality and decreases as the potentialities of the two networks become more similar.

IV. Results and Discussion

The experimental results show that two types of simplification can be unified to produce some complexity. Then, this productive simplification can produce a kind of phase transition in connection weights. Through this transition, the final networks were forced to approach the supposed prototype. This phase transition can also improve generalization, meaning that simplification should be considered a necessary step for improving generalization and interpretation. However, even in cases with better generalization and added complexity, simplification must still be operating inside neural networks.

A. Experiment Outline

The data set was created by imitating an actual business data set used to estimate the bankruptcy of companies [35]. The objective of the experiment is not to improve generalization but to interpret how bankruptcy occurs and what the major causes of bankruptcy are. In more technical terms, we aim to understand relationships between inputs and outputs. In the actual data set, linear and non-linear relationships are naturally mixed, making it seemingly impossible to explicitly understand the relationships between inputs and outputs. To address this issue, we created a data set with both linear and non-linear relationships between inputs and targets artificially. This artificial data set allows us to explicitly examine how neural networks respond to specific inputs to produce outputs. The number of input variables was seven. Of these, input No. 5, input No. 6, and input No. 7 were created non-linearly using exponential and sine functions, while the remaining inputs were linear.

The number of input patterns was 1000, and the number of hidden layers was ten, with ten neurons in each hidden layer. We used the PyTorch program package, with almost all parameter values set to default to ensure easy reproduction of the results presented here. The experiment was designed to make the neural networks as close as possible to the prototype network, which is assumed to be hidden within surface networks. The prototype network is the simplest form, and connection weights in this paper were computed using correlation coefficients between inputs and targets of the training data set.

Figure 5 shows a supposed prototype network computed by correlation coefficients between inputs and outputs from the artificial data set. As shown in the figure, the first four inputs are strongly correlated with outputs, whereas the last three inputs (No. 5 to No. 7) were only weakly correlated with outputs. As already mentioned, these three inputs were created non-linearly, and the linear correlation coefficients could not

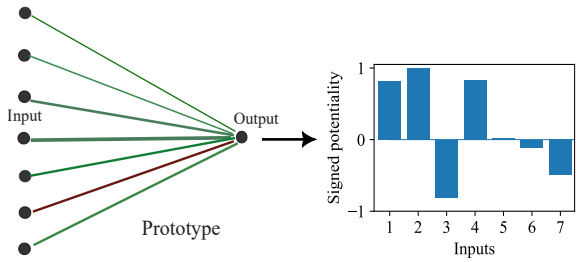


Figure 5. Supposed prototype (left) and the normalized strength of connection weights of the prototype network (right), computed by correlation coefficients between inputs and targets.

capture those relations. The final multi-layered neural network was then compressed into the simplest network without hidden layers. Compression was conducted layer by layer. We then compared the estimated and compressed networks with the assumed simplest prototype network.

A set of potentiality values was used to improve generalization performance, since generalization was considered the most important factor for evaluating network performance. However, we should note that the purpose of this study is to interpret how networks self-organize to achieve the simplest configuration. The parameters β_{loc} and β_{dis} were set to 0.005 to stabilize the learning process. The parameter β_{dch} was set to 2, and we first used local potentiality, gradually introducing distributed potentiality.

Before explaining local and distributed potentiality, it should be noted that they were computed only in the base case for illustrating the figures. In this case, distributed potentiality is actually the inverse of local potentiality, and therefore only the results of local potentiality are shown.

Now, the main findings can be summarized as follows:

- Productive simplification could control local and distributed potentiality forces to produce networks with better generalization.
- Connection weights with better generalization were obtained through a kind of phase transition during learning.
- Those connections with better generalization showed clearer characteristics even in hidden layers.
- By controlling local and distributed potentiality, we could produce the final and estimated prototypes, close to the supposed prototype.
- The experimental results show that complexity added during learning should be based on simplification procedures.

B. Potentiality and Entropy

The method could decrease the strength of total simplification potentiality. In addition, productive potentiality could control the strength of total and local potentiality appropriately for better generalization.

Figure 6 shows the total simplification potentiality (left), local potentiality (middle), and entropy (right). When only local potentiality was used in Figure 6(a), simplification potentiality (left) decreased gradually, and local potentiality (middle) decreased sharply. Similarly, entropy (right) decreased rapidly as the number of learning steps increased. When the diachronic parameter was two, corresponding to the best generalization performance in Figure 6(b), the total simplification potentiality was small but remained nearly constant throughout learning. Local potentiality decreased in the first stage and then increased gradually. This indicates that the effect of distributed potentiality appeared in the later stage of learning. Entropy showed the same trend, though less clearly. Productive simplification was able to control local potentiality while keeping total potentiality nearly unchanged. Figure 6(c) shows the results when distributed potentiality was used. Simplification potentiality decreased very slowly, while local potentiality increased gradually. However, entropy did not show a clear trend. Finally, Figure 6(d) shows the results with the conventional method without potentiality control. Simplification potentiality increased considerably, while local potentiality decreased only slightly, and entropy again did not show a clear trend.

The results show that productive simplification control can adjust the strength of both total simplification potentiality and local potentiality appropriately. Local potentiality can be decreased and then increased to introduce complexity. On the other hand, entropy, although conceptually related to potentiality, could not clearly represent these properties. Our potentiality method, while close to entropy, can more explicitly capture the characteristics of connection weights due to its computational simplicity.

C. Weights for All Layers

Better generalization was obtained through productive potentiality control and by making full use of all hidden layers with explicit characteristics.

Figure 7 shows the weights of all layers for the four methods. When only the local potentiality was used in Figure 7(a), a stronger weight could be seen in most layers. The local potentiality attempted to achieve simplification by weakening all connection weights except for a small number of specific ones. However, in the middle layers, these stronger weights became smaller, indicating that important weights could not be detected there. This means that under local potentiality, hidden layers in the middle could not acquire important information, and only the layers closer to the input and output played important roles in learning. When the diachronic parameter was set to two, yielding the best generalization in Figure 7(b), weights were arranged symmetrically. This indicates that important and non-important weights were explicitly separated, and this tendency was observed even in the middle layers. Better generalization was thus obtained by fully utilizing all hidden layers. When only the distributed potentiality was used, the symmetric arrangement of weights became somewhat blurred, as shown in Figure 7(c). Finally, when the conventional method without any potentialities was applied in Figure 7(d), all weights became obscure, showing almost no regularity.

The results show that better generalization was obtained by using all hidden layers and by explicitly detecting some

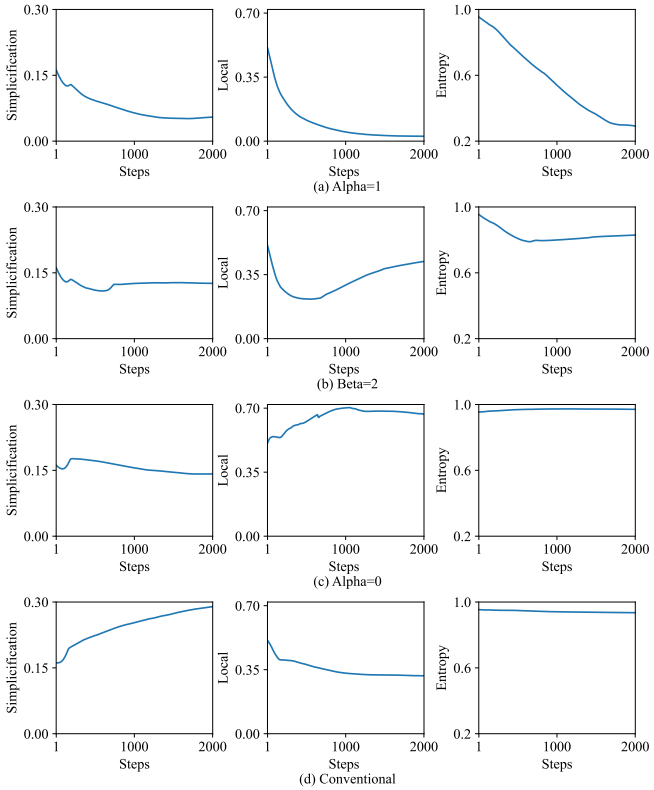


Figure 6. Total simplicity (left), local (middle), and entropy (right) as a function of the number of learning steps (epochs). Local potentiality $\alpha = 1$ (a), diachronic potentiality ($\beta = 2$) with best generalization (b), distributed potentiality $\alpha = 0$ (c), and conventional method without potentiality control (d).

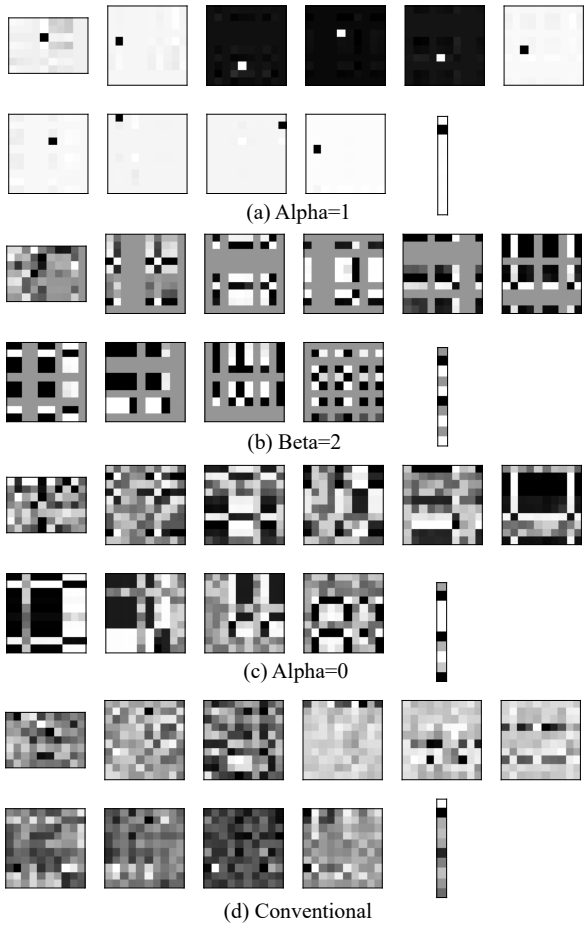


Figure 7. Weights for all layers, when the local potentiality was used (a), the diachronic potentiality was used (b), the distributed potentiality was used (c), and the conventional method was used (d).

characteristics in those layers. When such characteristics could not be detected, better generalization was not obtained.

D. Layer Potentiality

The layer potentiality demonstrated that better generalization could be obtained by considering hidden layers in the middle in addition to the input and output layers. These hidden layers seemed to resemble those obtained with the conventional method. The layer potentiality was computed by summing the individual local potentiality for each layer.

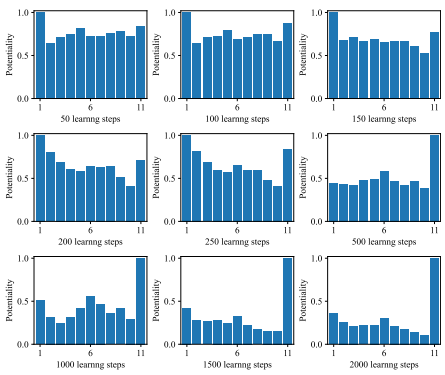
Figure 8(a) shows the results when only the local potentiality was used for learning. As can be seen in the figure, the input layer initially had the largest potentiality value, and the potentialities of subsequent layers gradually decreased, with the output layer having the largest potentiality at the end. Figure 8(b) shows the results when the diachronic parameter was set to two, corresponding to the best generalization. Compared with the results from using local potentiality alone, the sixth layer's potentiality increased in the later stage of learning. This indicates that hidden layers attempted to detect certain characteristics for better generalization. When only the distributed potentiality was used, as shown in Figure 8(c), layer No. 7 had a larger potentiality. Note, however, that better generalization under productive potentiality considered layer No. 6 as important. Finally, when the conventional method was applied, as in Figure 8(d), layer No. 6 became larger.

The results show that productive potentiality could utilize hidden layers in the middle. The characteristics detected by these layers appeared to be similar to those identified by the conventional method. The productive method attempted to enhance the characteristics found by the conventional method.

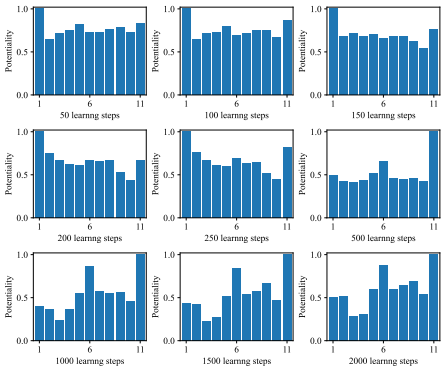
E. Ratio Potentiality

The results confirmed that productive potentiality aimed to increase the ratio potentiality as much as possible. This produced a kind of phase transition, leading to structural changes in connection weights that improved generalization. The ratio potentiality represents the ratio of the individual potentiality of a compressed network to that of the supposed prototype, computed using the correlation coefficients of the data set. When ratio potentiality increases, the similarity between the estimated and supposed prototype increases. The divergence between compressed and supposed prototypes was also computed for comparison. A smaller divergence indicates greater similarity.

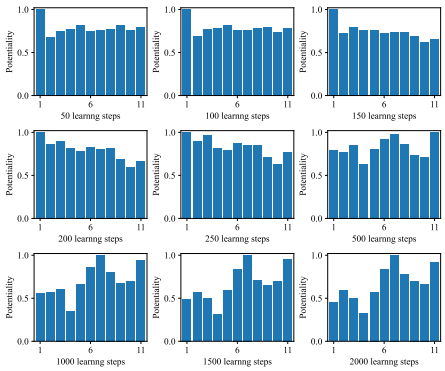
Figure 9 shows the ratio potentiality (left), divergence (middle), and generalization performance (right). When only the local potentiality was used, as in Figure 9(a), the ratio potentiality was initially high but gradually decreased. This



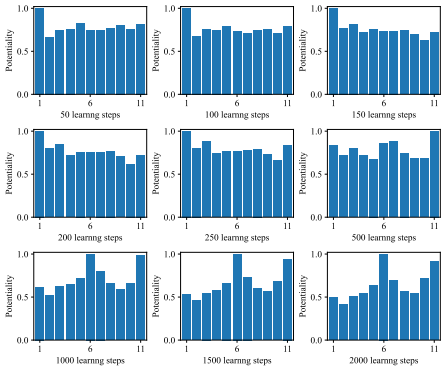
(a) Alpha=1



(b) Beta=2



(c) Alpha=0



(d) Conventional

Figure 8. Layer potentialities for all layers, when the ratio potentiality was used (a), the diachronic potentiality was used (b), the distributed potentiality was used (c), and the conventional method was used (d).

suggests that, at first, local potentiality control attempted to detect the prototype network. By contrast, divergence in the middle and correlation coefficients did not clearly show this tendency. Generalization (right) increased rapidly at first and then decreased slightly toward the end. When the diachronic parameter was set to two with the best generalization in Figure 9(b), ratio potentiality continued to increase as the number of learning steps grew. The estimated prototype was forced to remain closer to the supposed prototype throughout learning. The divergence in the middle also decreased as learning progressed, though it could not be normalized and the tendency was not as clear. The right figure shows that generalization performance increased gradually, with a kind of phase transition occurring in the middle of learning. This indicates that improved generalization performance resulted from structural changes in weights achieved by controlling both local and distributed potentiality. When only the distributed potentiality was applied, as shown in Figure 9(c), ratio potentiality increased only at the beginning of learning, while divergence did not clearly reveal this tendency. Consequently, generalization (right) was lower across the entire learning process. When the conventional method was used, as in Figure 9(d), the ratio potentiality was initially high, but divergence decreased at first and then gradually increased later. Generalization (right), however, remained relatively low.

The results show that productive simplification consistently attempted to detect the prototype network throughout learning, whereas the other methods attempted detection only at the beginning. This persistent attempt by productive simplification seems to produce a kind of phase transition that improves generalization.

F. Interpreting Compressed Weights

The results show that under productive simplification, the weights were close to those of the supposed prototype network, which were obtained by computing the correlation coefficients between inputs and outputs independently. The best generalization performance was obtained by preserving the weights of the supposed prototype network. This means that improved generalization could be achieved through simplification.

Figure 10 shows the weights of compressed networks obtained by four methods. When only the local potentiality was used, as in Figure 10(a), the weights were initially close to those of the prototype, but gradually changed, with only the weight from input No. 3 remaining strong. When generalization was highest and the diachronic parameter was set to two in Figure 10(b), the weights in the initial stage of learning remained almost unchanged until the final stage. Productive simplification attempted to detect the supposed prototype as consistently as possible throughout learning. When only the distributed potentiality was used, as shown in Figure 10(c), the prototype was detected in the early stage, but later the weight from input No. 3 became dominant, with several other weights also relatively large. When the conventional method was used, as in Figure 10(d), the weight from input No. 3 was

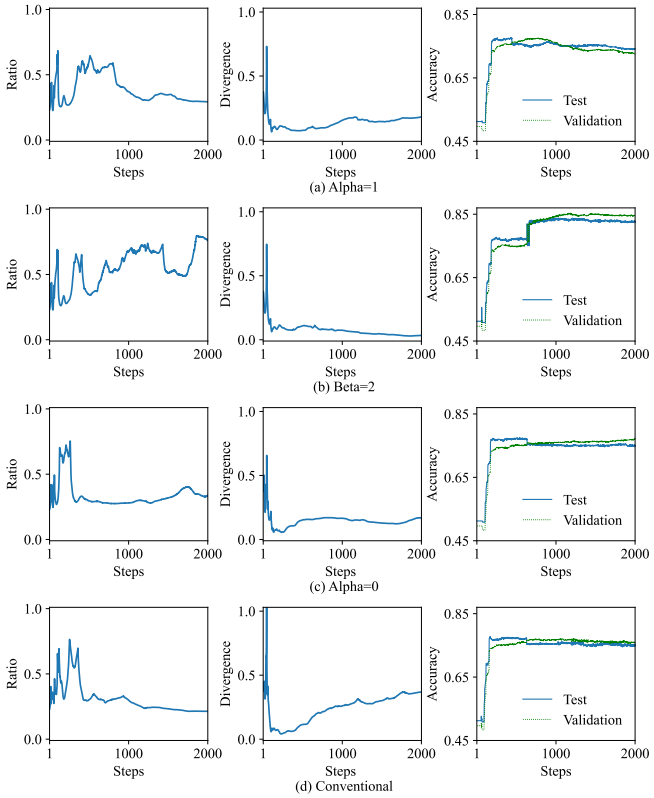


Figure 9. Ratio potentiality (left), divergence (middle), and generalization accuracy (right), when the local potentiality was used (a), the diachronic potentiality was used (b), the distributed potentiality was used (c), and the conventional method was used (d).

the largest, and all other weights were relatively weaker at the end.

The results show that the final connection weights obtained through productive simplification were very similar to those of the prototype network. This indicates that prototype detection or simplification was necessary even to increase generalization.

G. Numerical Summary

These numerical results reconfirmed that generalization was closely related to simplification. This suggests that behind seemingly complicated and nonlinear multilayer neural networks, there exists a simple prototype network that enables a surface network to improve generalization.

Table I summarizes the results, with the upper and lower tables representing the outcomes based on maximum ratio potentiality and generalization, respectively. The upper table shows the results when the ratio potentiality was the largest, where the final compressed networks were closest to the supposed prototype. For example, when the diachronic parameter was two, the ratio was the largest (0.796) with the largest number of learning steps (1857), the smallest simplification potentiality (0.126), and the highest generalization accuracy (0.827). Only the local potentiality was not so small (0.413), adding some complexity to the connection weights. This indicates that the best generalization was not obtained by simply decreasing the local potentiality, but by balancing local and distributed

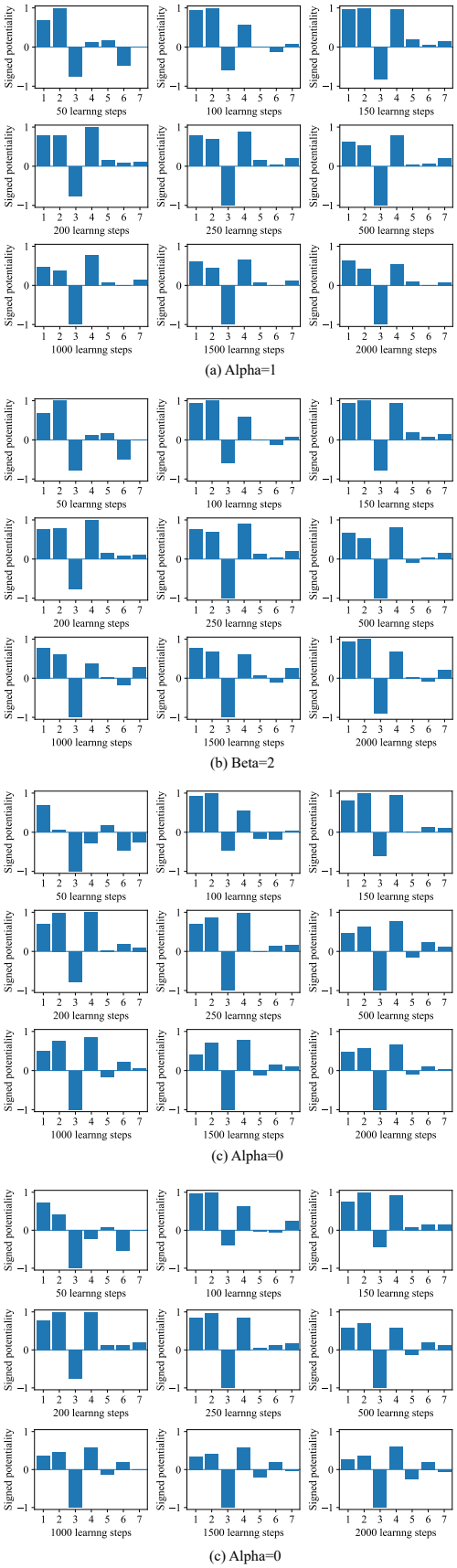


Figure 10. Weights of compressed networks, when the local potentiality was used (a), the diachronic potentiality was used (b), the distributed potentiality was used (c), and the conventional method was used (d).

potentiality. Even in this case, the total simplification was strictly maintained.

The lower table shows the results based on generalization performance. When the diachronic parameter was two, the generalization performance was the highest (0.838), and the ratio potentiality was also the largest (0.698) with the greatest number of learning steps (1032). The simplification potentiality was the second smallest (0.126), and the local potentiality was the second largest (0.301). Even when generalization performance was the criterion for evaluation, the ratio potentiality remained the largest and the total simplification potentiality the second smallest. The best generalization was obtained by controlling both simplification and local potentiality.

The results show that generalization could be improved by making networks as simple as possible, while allowing some complexity in the connection weights. This added complexity did not significantly increase the total simplification potentiality. These findings indicate that simplification can ultimately be controlled in correspondence with the necessary complexity.

TABLE I. SUMMARY OF RESULTS, BASED ON MAXIMUM RATIO POTENTIALITY (UPPER) AND MAXIMUM GENERALIZATION ACCURACY (LOWER).

Method	Ratio	Step	Simp	Local	Testing
Local	0.684	103	0.129	0.330	0.509
β_{dch}=2	**0.796**	**1857**	**0.126**	0.413	**0.827**
Distributed	0.754	260	0.176	**0.245**	0.768
Conventional	0.764	251	0.204	0.401	0.768

Method	Ratio	Step	Simp	Local	Testing
Local	0.277	254	**0.118**	**0.200**	0.776
β_{dch}=2	**0.698**	**1032**	0.126	0.301	**0.838**
Distributed	0.296	498	0.172	0.217	0.774
Conventional	0.341	166	0.195	0.404	0.777

V. CONCLUSION

The present paper aimed to show that there exists a strong simplification force inside all types of neural networks as a model of human cognition. In fact, there are different forms of simplification. For example, the number of components may be reduced in terms of local simplification, while the number of components may also be increased with reduced strength to capture the roles or features of each component. These different types of simplification can be unified productively to create a more powerful simplification force, enabling neural networks to handle a wider variety of new inputs. This productive simplification is naturally based on the simplification principle, but it can eventually produce some complexity in the connection weights. This simplified complexity generated by productive simplification is assumed to prevail in any cognitive system, including neural networks.

The method was applied to an artificial data set with linear and nonlinear inputs. The results show that two types of simplification could be unified productively to produce a more flexible simplification force. This productive unification could be realized through a phase transition, improving generalization performance. This means that productive simplification can drastically and structurally change network configurations for better generalization.

For future work, we should mention four points. First, we dealt only with the linear type of prototype network obtained by our method. Since our data set also contained nonlinear relations between inputs and outputs, we need to consider a nonlinear prototype network. Second, we need to extend our experiments to real-world data sets to examine how effectively our method can reveal the simplification force in neural networks. Third, the present method essentially attempted to simplify the well-known mutual information principle in neural networks, in which two contradictory operations of entropy maximization and minimization coexist. We therefore need to examine more precisely how our method can be used to realize the effect of mutual information maximization [21]. Fourth, we need to compare our simplification model with actual findings in neuroscience, where simplicity and complexity have often been discussed with some confusion. It is possible to extend our method to clarify the problem of simplicity in neuroscience [30], where simplification should play a more significant role.

REFERENCES

[1] R. Kamimura, "Neutralized synchronic and diachronic potentiality for interpreting multi-layered neural networks," in *Proceedings of the Seventeenth International Conference on Advanced Cognitive Technologies and Applications*, IARIA, 2025, pp. 17–25.

[2] T. Kohonen, "The self-organization map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[3] P. Schneider, M. Biehl, and B. Hammer, "Adaptive relevance matrices in learning vector quantization," *Neural Computation*, vol. 21, no. 12, pp. 3532–3561, 2009.

[4] Y. Lu, Y.-M. Cheung, and Y. Y. Tang, "Self-adaptive multiprototype-based competitive learning approach: A k-means-type algorithm for imbalanced data clustering," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1598–1612, 2019.

[5] M. Zidan *et al.*, "Quantum classification algorithm based on competitive learning neural network and entanglement measure," *Applied Sciences*, vol. 9, no. 7, p. 1277, 2019.

[6] T. Shinozaki, "Biologically motivated learning method for deep neural networks using hierarchical competitive learning," *Neural Networks*, vol. 144, pp. 271–278, 2021.

[7] M. G. Mahdy, A. R. Abas, and T. M. Mahmoud, "Multi-phase adaptive competitive learning neural network for clustering big datasets," in *The International Conference on Artificial Intelligence and Computer Vision*, Springer, 2021, pp. 731–741.

[8] M. Biehl, B. Hammer, and T. Villmann, "Prototype-based models in machine learning," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 7, no. 2, pp. 92–111, 2016.

[9] F. Pourpanah *et al.*, "A review of generalized zero-shot learning methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4051–4070, 2022.

[10] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7775–7784, 2018.

[11] C. Yang, A. Rangarajan, and S. Ranka, "Global model interpretation via recursive partitioning," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2018, pp. 1563–1570.

[12] S. M. Lundberg *et al.*, "From local explanations to global understanding with explainable ai for trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 2522–5839, 2020.

[13] G. Alain, "Understanding intermediate layers using linear classifier probes," *arXiv preprint arXiv:1610.01644*, 2016.

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1135–1144.

[15] D. Garreau and U. Luxburg, "Explaining the explainer: A first theoretical analysis of lime," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 1287–1296.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Stat*, vol. 1050, p. 9, 2015.

[17] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang, "Face model compression by distilling knowledge from neurons," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[18] S. Hooker, A. Courville, G. Clark, Y. Dauphin, and A. Frome, "What do compressed deep neural networks forget?" *arXiv preprint arXiv:1911.05248*, 2019.

[19] R. Mishra, H. P. Gupta, and T. Dutta, "A survey on deep neural network compression: Challenges, overview, and solutions," *arXiv preprint arXiv:2010.03954*, 2020.

[20] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[21] R. Linsker, "Self-organization in a perceptual network," *Computer*, vol. 21, no. 3, pp. 105–117, 1988.

[22] ——, "Perceptual neural organization: Some approaches based on network models and information theory," *Annual Review of Neuroscience*, vol. 13, no. 1, pp. 257–281, 1990.

[23] ——, "Local synaptic learning rules suffice to maximize mutual information in a linear network," *Neural Computation*, vol. 4, no. 5, pp. 691–702, 1992.

[24] ——, "Improved local learning rule for information maximization and related applications," *Neural Networks*, vol. 18, no. 3, pp. 261–265, 2005.

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

[26] A. M. Saxe *et al.*, "On the information bottleneck theory of deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124 020, 2019.

[27] Z. Goldfeld and Y. Polyanskiy, "The information bottleneck problem and its applications in machine learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 19–38, 2020.

[28] R. K. Mahabadi, Y. Belinkov, and J. Henderson, "Variational information bottleneck for effective low-resource fine-tuning," *arXiv preprint arXiv:2106.05469*, 2021.

[29] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[30] E. Genç *et al.*, "Diffusion markers of dendritic density and arborization in gray matter predict differences in intelligence," *Nature Communications*, vol. 9, no. 1, p. 1905, 2018.

[31] M. M. V. Hulle, "The formation of topographic maps that maximize the average mutual information of the output responses to noiseless input signals," *Neural Computation*, vol. 9, no. 3, pp. 595–606, 1997.

[32] 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, 2021.

[33] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2006, pp. 535–541.

[34] J. O. Neill, "An overview of neural network compression," *arXiv preprint arXiv:2006.03669*, 2020.

[35] K. Shimizu, *Multivariate Analysis (in Japanese)*. Nikkan Kogyo Shinbun, 2009.