Learning from Emergence: A Study on Proactively Inhibiting the Monosemantic Neurons of Artificial Neural Networks

Jiachuan Wang
The Hong Kong University of Science and Technology
Hong Kong SAR, China
jwangey@connect.ust.hk

Shimin Di*
The Hong Kong University of Science and Technology
Hong Kong SAR, China
sdiaa@connect.ust.hk

Lei Chen†
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
leichen@hkust-gz.edu.cn

Charles Wang Wai Ng†
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
charles.ng@ust.hk

Abstract

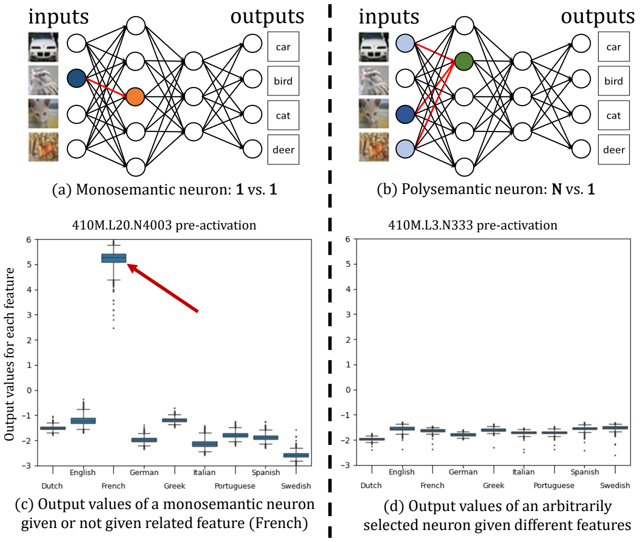
Recently, emergence has received widespread attention from the research community along with the success of large-scale models. Different from the literature, we hypothesize a key factor that promotes the performance during the increase of scale: the reduction of monosemantic neurons that can only form one-to-one correlations with specific features. Monosemantic neurons tend to be sparser and have negative impacts on the performance in large models. Inspired by this insight, we propose an intuitive idea to identify monosemantic neurons and inhibit them. However, achieving this goal is a non-trivial task as there is no unified quantitative evaluation metric and simply banning monosemantic neurons does not promote polysemanticity in neural networks. Therefore, we first propose a new metric to measure the monosemanticity of neurons with the guarantee of efficiency for online computation, then introduce a theoretically supported method to suppress monosemantic neurons and proactively promote the ratios of polysemantic neurons in training neural networks. We validate our conjecture that monosemanticity brings about performance change at different model scales on a variety of neural networks and benchmark datasets in different areas, including language, image, and physics simulation tasks. Further experiments validate our analysis and theory regarding the inhibition of monosemanticity.

CCS Concepts

• **Theory of computation** → **Machine learning theory**; *Mathematical optimization*; • **Computing methodologies** → *Knowledge representation and reasoning*.

Keywords

Deep Learning, Artificial Neural Networks, Emergent Abilities, Monosemanticity



(a) Monosemantic neuron: **1** vs. **1**

(b) Polysemantic neuron: **N** vs. **1**

(c) Output values of a monosemantic neuron given or not given related feature (French)

(d) Output values of an arbitrarily selected neuron given different features

Figure 1: Demonstration of important concepts with statistics: (a) A monosemantic neuron (orange) ideally activates for one specific type of feature. (b) A polysemantic neuron (green) activates for multiple features. (c) The output values of a monosemantic neuron when different features are inputted. Its related feature (French) produces values that significantly stand out from other features. (d) The output values of an arbitrarily selected neuron (layer 3, number 333) given different features. These statistics are obtained by inspecting the Pythia-v0 410M model [4].

ACM Reference Format:
Jiachuan Wang, Shimin Di, Lei Chen, and Charles Wang Wai Ng. 2024. Learning from Emergence: A Study on Proactively Inhibiting the Monosemantic Neurons of Artificial Neural Networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671776

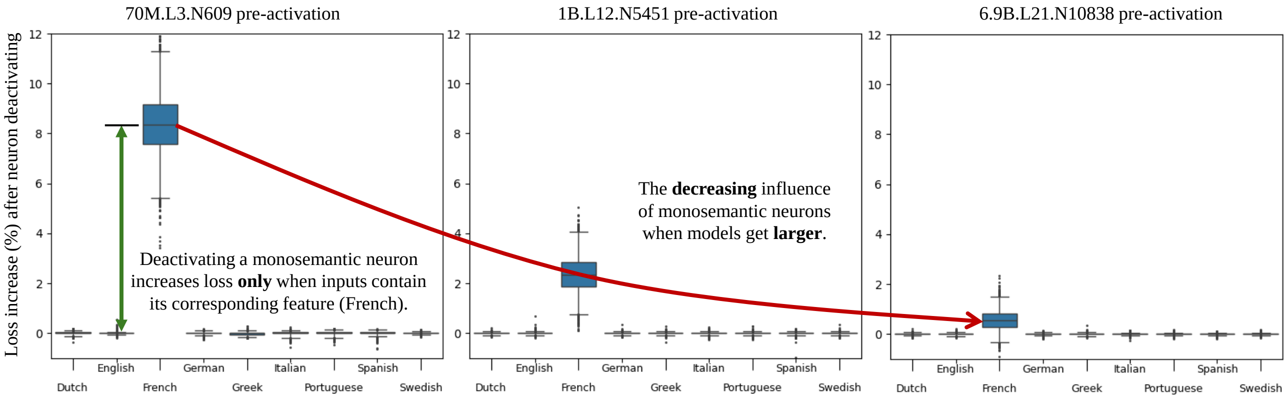
1 Introduction

The activity of artificial neural networks diminished for decades before experiencing great success after the 2010s [17, 18]. One major difference compared to previous models is the increasing scale. In recent years, large neural networks have much larger scales in terms of datasets, model sizes, and training quantities, which have achieved remarkable results in various fields [10, 34]. Given increasing scales, the performance improvements can usually be

[*] The corresponding author.
[†] Also with The Hong Kong University of Science and Technology, Hong Kong SAR, China

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671776



(a) Loss increase in a **smaller-size** (70M) model (b) Loss increase in a **middle-size** (1B) model (c) Loss increase in a **larger-size** (6.9B) model

Figure 2: We detect the monosemantic neurons of "French" following the sparse probing paper [13] and run the experiments on Pythia-v0 [4]. After deactivating a monosemantic neuron for "French", there is an increase in the loss given inputs of different language features (e.g., Dutch and Greek) on Pythia models of different scales: (a) on the 70M Pythia-v0 model, (b) on the 1B Pythia-v0 model, and (c) on the 6.9B Pythia-v0 model. It can be observed that these neurons are typically monosemantic, causing a large increase in loss only when the input contains "French" (see green arrow). However, for larger models, deactivation of these neurons leads to a smaller increase in loss (see red arrow). This gives us a hint that monosemanticity may be negatively related to the scale and performance of larger models.

estimated under the scaling law [16], which follows stable but slow increasing trend of power laws [22]. However, improvements that significantly deviate from estimation occur when the scale increases. Emergence, an intriguing phenomenon in large-scale models, refers to the gradual improvement of model performance before the scale reaching a certain threshold, followed by a rapid enhancement once the threshold is surpassed [39]. Increasing evidence suggests that the surprises may not arise from new module and architecture designs, but rather from the underlying nature of scale changes [42]. A critical question arises: ***people increase the model scale and get better results, but what has changed underlying the process?***

From the perspectives of *monosemantic* and *polysemantic* neurons, some pioneer works try to interpret the performance from small to large-scale models. Through statistical analysis of the relationships between neurons and input features, a neuron is considered monosemantic if it forms a 1-1 correlation with its related input feature [3, 9] (Figure 1(a)). In contrast, polysemantic neurons activate for several features that are weakly correlated [5, 9, 11, 35] (Figure 1(b)). By comparing Figure 1(c) and (d), we can observe that the activation level of a monosemantic neuron for the feature "French" is significantly different from that of the neuron for other features and from that of other non-monosemantic neurons for "French". Researchers find that smaller models incur larger errors [13] when an input's related monosemantic neuron is disabled. Following the favor of monosemantic neurons from studies on small-scale models, existing works focus on enhancing or extracting monosemanticity [9, 13, 35]. However, some evidences support that monosemantic neurons are sparser in larger models [13, 35]. Those observations imply a positive relationship between the decrease in monosemanticity and the increase in model scale. In other words, monosemanticity could be negatively related to large-scale and good performance. In Figure 2, we conduct experiments to show that, when a monosemantic neuron for "French" is deactivated, a smaller model has a greater increase in loss than a larger model. In this paper, inspired by those observations, we propose an important hypothesis that **the decrease of monosemantic neurons is a key factor in achieving higher performance as the model scale increases**. Figure 5 in Appendix A of technical report [1] shows an example analogy to demonstrate our idea.

Since the research community does not realize the above hypothesis, we rather conclude the current paradigm of training neural networks as a **passive** process in decreasing monosemantic neurons. This raises an interesting question: *can we* ***proactively*** *induce the decrease of monosemantic neurons in artificial neural networks to achieve high performance?*

In this paper, we propose to learn from emergence and present a study on proactively inhibiting the monosemantic neurons, which is achieved in two phases: (i) monosemantic neuron detection and (ii) monosemantic neuron inhibition. Unfortunately, it is challenging to design detection and inhibition methods. First, strictly defining monosemantic neurons is still under discussion in quantitative analysis, i.e., detection is impossible without a consensual definition. Besides, existing works [13, 30, 35] for detecting the activation degree of neurons introduce extra computation and may suffer from the efficiency issue. Second, we prove that simply prohibiting the activation of monosemantic neurons will intensify the monosemanticity of artificial neural networks. To solve these challenges, we first propose a new metric to measure the monosemanticity of neurons with the guarantee of efficiency for online computation. Then, we identify the drawbacks of simple methods and propose a theoretically supported Reverse Deactivation method to suppress monosemantic neurons and promote the ratios of polysemantic neurons in training neural networks. Our contributions are summarized as follows:

- Inspired by emergence, we propose a novel idea to study the impact of monosemantic and polysemantic neurons on the performance of artificial neural networks. Different from the literature, we propose a hypothesis that monosemanticity could be negatively related to good performance on large-scale models.
- There is no quantitative definition of monosemanticity with high computational efficiency. To effectively detect monosemantic neurons, we propose a new evaluation metric of monosemanticity with a theoretical guarantee on online computation.
- To overcome the inherent drawback of the naive inhibition methods, we propose a theoretically supported reverse deactivation method to suppress monosemantic neurons. It can be integrated as a parameter-free and flexible insertion module, which can adapt to various neural network structures and introduce no computational overhead during testing.
- To validate the effectiveness and generalizability of our method, we conducted tests on various tasks including language, image, and physics simulation. The proposed method MEmeL is applied to milestone models such as Bert, ConvRNN, and different architectures like Transformer, CNN, and RNN. The experimental results demonstrate that MEmeL performs well on different neural networks and corresponding tasks.

2 Preliminary

Given a training set $D = \{(\mathbf{x}, \mathbf{y})\}$ of labeled examples, artificial neural networks aim to train a neural network model $f(\mathbf{W}; \mathbf{x})$ parameterized by $\mathbf{W}$ so that the predicted output of $\mathbf{x}$ is close to the ground truth $\mathbf{y}$. Under the supervised learning task, we take the cross entropy function $\mathcal{L}(\mathbf{W}; D)$ as the example:

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}; D) = \min_{\mathbf{W}} - \sum_{(\mathbf{x},\mathbf{y}) \in D} \mathbf{y} \log f(\mathbf{W}; \mathbf{x}), \tag{1}$$

where $\mathbf{x} = [x_1, \cdots, x_d] \in \mathbb{R}^d$ is a d-dimensional vector representing the n-th input data sample, such as a weather record series covering d days or an image with height × width = d pixels. Without loss of generality, let $f(\mathbf{W}; \mathbf{x})$ be a L-th layer fully connected neural networks [14]. We denote h_i^ℓ, $\sigma_i^\ell(\cdot)$, z_i^ℓ as the input after linear transformation, activation function, and output of the i-th hidden unit at ℓ-th layer, respectively. Then, we may formally define the computation within neurons as follows:

$$h_j^\ell = \sum_i w_{ij}^\ell z_i^{\ell-1}, \tag{2}$$

$$z_i^\ell = \sigma_i^\ell(h_i^\ell), \tag{3}$$

where $z_i^0 \in \mathbf{z}^0 = \mathbf{x}$, w_{ij}^ℓ denotes the weight of i-th hidden neuron at $(\ell - 1)$-th layer to j-th hidden neuron at ℓ-th layer. We use $\mathbf{o} = \mathbf{z}^L$ to denote the model output. Existing works on monosemanticity mainly study the layers of values outputted by activation functions (i.e., $\mathbf{z}^\ell$). In contrast to other linear layers, these activated values undergo element-wise nonlinearity and are more likely to have independent meanings [6, 13]. Thus, without loss of generality, we zoom in and study monosemanticity based on one single layer of activated values, e.g., $\mathbf{z}^\ell$ in the ℓ-th layer. Then, $f(\mathbf{W}; \mathbf{x})$ can be divided into a frontal model $f_1(\cdot)$ and a followed model $f_2(\cdot)$ at ℓ-th layer of output as follows:

$$\mathbf{z} = f_1(\mathbf{x}) = \sigma^\ell(\mathbf{W}^\ell(\cdots \sigma^2(\mathbf{W}^2 \sigma^1(\mathbf{W}^1 \mathbf{x})) \cdots)), \tag{4}$$

$$\mathbf{o} = f_2(\mathbf{z}, \mathbf{x}) = \text{Softmax}(\mathbf{W}^L \cdots \sigma^{\ell+2}(\mathbf{W}^{\ell+2} \sigma^{\ell+1}(\mathbf{W}^{\ell+1} \mathbf{z})) \cdots), \tag{5}$$

where the frontal model $f_1(\cdot)$ takes the original data $\mathbf{x}$ as inputs and delivers output values in ℓ-th layer $\mathbf{z}$ (with the superscript ℓ omitted for simplicity in notation.) As the rest components of the network, the following model $f_2(\cdot)$ takes $\mathbf{x}$ and $\mathbf{z}$ and outputs $\mathbf{o}$ (the entire model's output), and $\mathbf{W}^\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ denotes weights of linear transformation from the ℓ-th layer to the $(\ell + 1)$-th layer.

2.1 Activation and Monosemantic

Despite the progress of neural networks and emergence, there is no consensus definition for an "activated" neuron and a "monosemantic" neuron [9]. Here we try to summarize an intuitive introduction to describe them.

The Concept of Activation: If an input $\mathbf{x}^{[n]}$ triggers a neuron z_i to output a value $f_1(\mathbf{x}^{[n]})_i$ that deviates "significantly" from the statistical mean (i.e., $\bar{z}_i$) , we say that neuron z_i is activated by input $\mathbf{x}^{[n]}$. The challenge in defining activation lies in reaching a consensus on the meaning of "significantly". Instead, we can provide a relative definition as follows. Generally, i-th neuron at ℓ-th layer is considered to be more activated on $\mathbf{x}^{[2]}$ if:

$$\left\| \bar{z}_i - (f_1(\mathbf{x}^{[1]}))_i \right\| < \left\| \bar{z}_i - (f_1(\mathbf{x}^{[2]}))_i \right\|, \tag{6}$$

where $\mathbf{x}^{[n]} \in D$ is the n-th sample in the training set D, $\bar{z}_i$ is the mean value of i-th neuron given all training samples D at ℓ-th layer, $(f_1(\mathbf{x}^{[1]}))_i$ denotes the i-th output of the frontal model (i.e., z_i^ℓ for $\mathbf{x}^{[1]}$), and $\|\cdot\|$ is a distance metric. However, while this relative definition is accurate for illustration purposes, it is not concise and is difficult to use for further analysis.

Thus, to give a one-input-wise definition, one may rely on a threshold to define whether a neuron is activated, such as a hard threshold τ. If the deviation of activation value from the mean $\|\bar{z}_i - (f_1(\mathbf{x}^{[n]}))_i\|$ exceeds τ, it is generally considered that the neuron is activated. Otherwise, it is in an inactive state. The reason for setting τ is that neurons will generally have certain fluctuating outputs even for those unrelated different inputs. Unfortunately, setting a universal or adaptive τ remains to be explored. Therefore, it is also impractical to quantify activation by the threshold.

The Concept of Monosemanticity: However, further illustration for monosemanticity is closely related to a definition of activation that considers only one input at a time. We provide an abstract definition for it: $act(z_i, \mathbf{x}^{[n]})$, which equals 1 when z_i is activated by $\mathbf{x}^{[n]}$, and 0 otherwise. One can use a task-oriented definition as implementation.

To understand neural networks, an important research direction is to correlate neuron activations with human-interpretable features, such as Python and German for language processing; fur and grass for image processing, et al.. Existing works construct feature datasets $\{C_1, \cdots, C_m\}$ for m features, each containing a set of inputs. These feature datasets are specifically designed to partition the inputs $X = \{\mathbf{x}\}$, mathematically:

$$\forall_{i \neq j} C_i \cap C_j = \emptyset; \bigcup_{i=1}^{m} C_i = X.$$

A neuron z_i is "monosemantic" if it is only activated on inputs that share a specific feature C_j, that is:

$$\forall_{\mathbf{x}} act(z_i, \mathbf{x}) = 1, \mathbf{x} \in C_j; \forall_{\mathbf{x}} act(z_i, \mathbf{x}) = 0, \mathbf{x} \notin C_j$$

However, features are human-defined and vary a lot. In the previous study, Gurnee et al. [13] considered about 100 features, while Trenton et al. [35] studied up to 10^5 features to fully capture monosemanticity. Without unified feature datasets, it is also difficult to explicitly formalize the definition of "monosemantic". More related works are discussed in Appendix B.

Thus, to detect monosemantic neurons, previous studies require manually labeled feature datasets and time-consuming offline computations after model training. To detect and inhibit monosemantic neurons during training, it is necessary to define a lightweight online metric $\phi(\cdot)$ that does not rely on feature datasets, which indicates the level of monosemanticity of a neuron.

2.2 Monosemanticity Inhibition

To achieve the desired output z, deep learning models use optimization strategy ω to update parameters ($\mathbf{W}$), such as minimizing the explicit loss function through gradient descent.

However, to inhibit monosemanticity, there is no related loss function to minimize. In this paper, we achieve this objective in two phases.

Recall that the model f is parameterized by $\mathbf{W}$ and split into a frontal model f_1 and a followed model f_2 with respect to the studied layer of neurons $\mathbf{z}$.

Assume that we feed input $\mathbf{x}$ to f and find $z_i \in \mathbf{z}$ is monosemantic. By using a well-designed optimization strategy ω, parameters are updated to $\mathbf{W}^*$. By feeding the same input $\mathbf{x}$ into the neural network, the frontal model, the followed model, and the layer of neurons are updated to f_1^*, f_2^*, and $\mathbf{z}^*$, respectively. We hope that:

- With updated f_1, neuron z_i is less activated for input $\mathbf{x}$. Formally, given old $z_i \in \mathbf{z} = f_1(\mathbf{x})$ and updated $z_i^* \in \mathbf{z}^* = f_1^*(\mathbf{x})$, we expect:
$$\phi(z_i^*) < \phi(z_i).$$
- With updated f_2, the output is still robust when neuron z_i is deactivated. Formally, replace z_i with a weakly activated value z_i' (i.e., $\phi(z_i') < \phi(z_i)$) to obtain $\mathbf{z}'$, we expect the loss $\mathcal{L}$ satisfies:
$$\mathcal{L}(f_2^*(\mathbf{z}', \mathbf{x})) < \mathcal{L}(f_2(\mathbf{z}', \mathbf{x})).$$

The two phases (1) prevent the **neuron** from exclusively serving only **one feature** type, and (2) prevent this **feature** modeling from relying solely on **one neuron**.

3 Methods

As discussed in Sec. 1, the existing neural network training paradigm is a passive process of decreasing monosemantic neurons. To proactively reduce monosemanticity, it is an intuitive solution to detect monosemantic neurons and inhibit them. Unfortunately, achieving these goals is challenging because the monosemanticity measurement does not exist and the simple inhibition method is counterproductive. Therefore, we first propose a new metric to measure the monosemanticity of artificial neurons with the guarantee of computation efficiency, then introduce a theoretically supported inhibition method to suppress monosemantic neurons to proactively reduce the ratios of monosemantic neurons in training artificial neural networks.

3.1 Metric for Monosemanticity

To proactively detect monosemantic neurons, it is important to design a metric that is general for different tasks and efficient to calculate. However, as discussed in Sec. 2, strictly defining "activated" is still under discussion in quantitative analysis [9], which in turn leads to the difficulty of defining "monosemantic". Although some pioneer works explore the measurement of monosemanticity, these metrics are inflexible for different tasks since they require a predefined and manually labeled feature dataset [13, 35]. To address these limitations, we propose a robust metric $\phi(\cdot)$ that can accurately detect monosemantic neurons in this subsection. This metric fulfills two important criteria: (1) generality to make the metric do not rely on any specific dataset, (2) efficiency to enable fast online detection during training.

Intuitively, defining monosemantic neurons mainly requires starting from two principles: high deviation of activation value and low frequency of activation. First, a neuron is considered "activated" when its output value for the current input deviates from the mean value of outputs for all possible inputs. For a monosemantic neuron, its value distribution is more skewed and incurs large deviation [9]. Second, each monosemantic neuron only activates when its corresponding feature is inputted, which rarely happens considering the current datasets with steadily growing scales of samples and feature types. Following two principles, we formally define our metric Monosemantic Scale (MS for short) as follows.

Definition 3.1 (Monosemantic Scale). Given a neuron $z_i \in \mathbf{z}$, we denote its historical samples under m inputs $\{\mathbf{x}^{[1]}, \mathbf{x}^{[2]}, \cdots, \mathbf{x}^{[m]}\}$ as $\{z_i^{[1]}, z_i^{[2]}, \cdots, z_i^{[m]}\}$ and new value under the $(m+1)$-th input $\mathbf{x}^{[m+1]}$ as $z_i^{[m+1]}$. The Monosemantic Scale is defined as:

$$\phi(z_i^{[m+1]}) = \frac{(z_i^{[m+1]} - \bar{z}_i)^2}{S^2}, \tag{7}$$

where

$$\bar{z}_i = \frac{\sum_{j=1}^{m} z_i^{[j]}}{m}, S^2 = \frac{\sum_{j=1}^{m} (z_i^{[j]} - \bar{z}_i)^2}{m-1},$$

are the sample mean and sample variance, respectively.

In the following contents, we will focus on this single neuron and use z for simplicity.

As shown in Eq. (7), the measurement $\phi(\cdot)$ is proportional to the degree of deviation from the mean. The high deviation of the activation value ensures that the term $(z^{[m+1]} - \bar{z})^2$ in ϕ can effectively identify neurons with high monosemanticity. Besides, the metric is inversely proportional to the degree of fluctuation because the size of the deviation is also highly correlated with the fluctuation of the activation value on the current data set. Since $\bar{z}_i$ is mainly determined by the values when the neuron is deactivated, using the mean as the benchmark for evaluating deviations ensures that the defined neurons with high activation values are rare, i.e., low frequency.

As discussed in Sec. 1 and 2, prior works need to first find the neuron-feature relationships [13, 35] under the manually defined feature data set. Instead, we relax the requirement of discovering corresponding features and eliminate the need for a predefined feature set since we focus on finding monosemantic neurons.

Metric Online Computing Guarantee. It is mandatory to compute the metric $\phi(\cdot)$ to avoid excessive computational burden caused by detecting monosemanticity. As discussed in Sec. 2, pioneer works (e.g., pairwise comparison in Eq. (6)) and other detection methods like probes necessitate training and offline inference for statistical analysis may be costly for online training. Here we show that for each neuron, our proposed MS $\phi(\cdot)$ can be obtained by keeping track of 2 variables in constant time ($O(1)$). Since inputs are typically received in batches (with the number of new samples being greater than 1) during the training of deep learning models, we present the following lemma for training with a batch size of b.

LEMMA 3.2. *Denote μ_m as the value of the sample mean $\bar{z}$ given m samples, while v_m as the sample variance S^2. When the $(m+1)^{th} \sim (m+b)^{th}$ samples $z^{[m+1]}, \cdots, z^{[m+b]}$ come, one can obtain the updated values via:*

$$\mu_{m+b} = \frac{m\mu_m + b\mu'_b}{m+b}, \tag{8}$$

$$v_{m+b} = \frac{mb(\mu_m - \mu'_b)^2}{(m+b-1)(m+b)} + \frac{bv'_b + (m-1)v_m}{m+b-1}, \tag{9}$$

where $\mu'_b = \frac{\sum_{i=1}^{b} z_{[m+i]}}{b}$ and $v'_b = \frac{\sum_{i=1}^{b}(z_{[m+i]} - \mu'_b)^2}{b}$, which is of $O(1)$ time and memory complexity as b is a constant.

The proof is given in Appendix D.1 in technical report [1]. In the implementation, we introduce a forgetting mechanism during model updates, where the influence of previous samples should decay. For details, please refer to Algorithm 1 in Appendix A.1.

3.2 Inhibition of Monosemanticity

After defining a quantitative metric in Sec. 3.1, it is intuitive to inhibit those monosemantic neurons directly. Unfortunately, simply deactivating monosemantic neurons will intensify the monosemanticity of neural networks. In this subsection, we will first prove the weakness of the native solution in Sec. 3.2.1. To address this unexpected phenomenon, we propose a theoretically supported reverse deactivation method in Sec. 3.2.2.

3.2.1 Naive Deactivation. Recall that given a monosemantic neuron z and an input $\mathbf{x}$ that contains its exclusive feature, we aim to optimize the model in two aspects: (1) decrease the activation level of z when given $\mathbf{x}$ (Figure 3(b)); and (2) reduce the reliance of output $\mathbf{o}$ on the activation of z (Figure 3(c)).

Without loss of generality, we focus on the most monosemantic single neuron z in a layer of neurons $\mathbf{z}$. As defined in Sec. 3.1, such a neuron has a large $\phi(z) = (z - \bar{z})^2/S^2$, which is expected to be inhibited during model optimization. From the perspective of information theory, suppose that the distribution of activation values follows a normal distribution, a neuron provides the least amount of information $I(z)$ when its value equals its statistical mean:

$$\min_{z} \mathbb{E}[I(z)] = \min_{z} \mathbb{E}[-\log(P(z))] = -\mathbb{E}\log(P(\bar{z})),$$

which is also its most inactive state. $P(\cdot)$ represents the probability density function of values of z.

Thus, a straightforward idea to deactivate z is to modify its value to $\bar{z}$. We denote the modified neuron as z'. One can find two naive solutions to deactivate a neuron:

$$\text{way}(a): \quad z' = \bar{z}_{ng}, \tag{10}$$

$$\text{way}(b): \quad z' = z + (\bar{z} - z)_{ng}, \tag{11}$$

where subscript $\cdot_{ng}$ refers to a "no gradient" fixed-value scalar without trainable parameters, compared with $z \in \mathbf{z} = f_1(\mathbf{x})$ which is updatable. Two examples are given in Figure 3(e,f). Both solutions ensure that f_2 receives an updated z', in which the value of $\mathcal{V}(z') = \bar{z}$ provides little information. As z' is valueless, f_2 has to adjust its parameter to utilize information from other neurons in $\mathbf{z}$ for good output $\mathbf{o}^* = f_2^*(\mathbf{z}', \mathbf{x})$. Such a process achieves our second goal of reducing the dependence of output $\mathbf{o}$ on the activation of z (Figure 3(l)).

However, neither of the two solutions can achieve the first goal (i.e., training f_1 to inhibit the activation of z for a single feature). To be more specific, method (a) wastes all the gradient for f_1 in generating z, preventing the related parameters from being updated. Method (b) outputs $\mathcal{V}(z') = \bar{z}$ by compensating for the gap. As the model obtains a good result with z, by calculating the gradient, f_1 will be updated to push its output value from $\bar{z}$ to z, expressed as $\bar{z} + \delta(z - \bar{z})$, where $\delta > 0$ up to the learning rate (Figure 3(i)). Ironically, the actual output of z is updated to $z + \delta(z - \bar{z})$, deviating from $\bar{z}$ by a factor of $1 + \delta$:

$$\frac{z + \delta(z - \bar{z}) - \bar{z}}{z - \bar{z}} = 1 + \delta.$$

Therefore, these two simple solutions either contribute nothing to the deactivation of z or even enhance its activation (Figure 3(g,k)).

After delving deeper into the literature, it is possible that previous researchers also recognized the negative effects of monosemanticity, but they may have discontinued their efforts after obtaining bad results from implementing these naive solutions.

3.2.2 Reversed Deactivation. To address the aforementioned problems, we propose our method called Reversed Deactivation (RD for short). Following the above-mentioned definitions, we replace the original z with z' (Figure 3(f)):

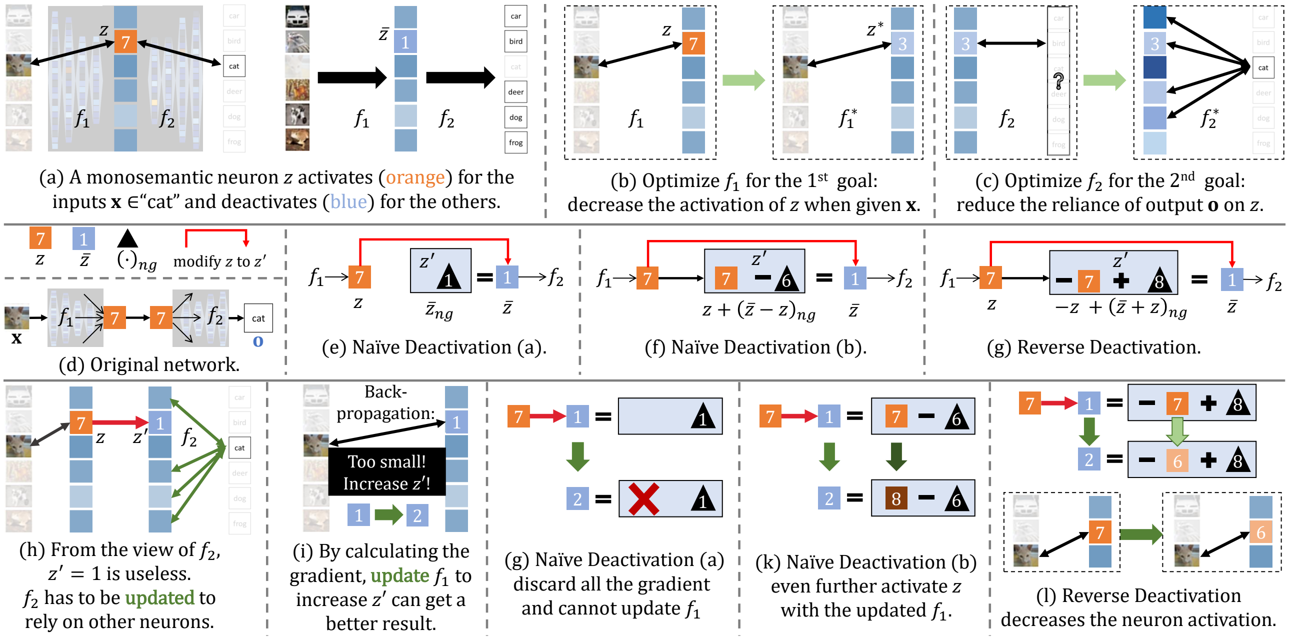
$$z' = -z + (\bar{z} + z)_{ng}. \tag{12}$$

Similar to baselines, RD also ensures the second goal of decreasing the dependence of output $\mathbf{o}$ on the activation of z. To be more specific, f_2 receives a value $\mathcal{V}(z') = \mathcal{V}(-z + (\bar{z} + z)_{ng}) = \bar{z}$, which provides little information and requires f_2 to learn the information from other neurons in $\mathbf{z}$ (Figure 3(h)).

Besides, RD can inhibit the activation level of z when given $\mathbf{x}$. In short, after calculating the gradient, f_1 will update the trainable parameter to push its output value from $\bar{z}$ to z (Figure 3(i)). Different from method (b), the gradient on z is reversed. To achieve the same shifted value $\bar{z} + \delta(z - \bar{z})$ mentioned above, an insight into the update can be expressed as:

$$\mathcal{V}(-z + (\bar{z} + z)_{ng}) = \bar{z} \tag{13}$$

$$\rightarrow \mathcal{V}(-(z - \delta(z - \bar{z})) + (\bar{z} + z)_{ng}) = \bar{z} + \delta(z - \bar{z}) \tag{14}$$



(a) A monosemantic neuron z activates (orange) for the inputs $\mathbf{x}$ ∈“cat” and deactivates (blue) for the others.

(b) Optimize f_1 for the 1st goal: decrease the activation of z when given $\mathbf{x}$.

(c) Optimize f_2 for the 2nd goal: reduce the reliance of output $\mathbf{o}$ on z.

(d) Original network.

(e) Naïve Deactivation (a).

(f) Naïve Deactivation (b).

(g) Reverse Deactivation.

(h) From the view of f_2, $z' = 1$ is useless. f_2 has to be updated to rely on other neurons.

(i) By calculating the gradient, update f_1 to increase z' can get a better result.

(g) Naïve Deactivation (a) discard all the gradient and cannot update f_1

(k) Naïve Deactivation (b) even further activate z with the updated f_1.

(l) Reverse Deactivation decreases the neuron activation.

Figure 3: Illustration of problems and solutions to inhibit monosemanticity. (a) A monosemantic neuron z only activates (orange) for the feature “cat" with a high mean value (= 7). z is deactivated (blue) for other inputs with a small mean value ($\bar{z}$ = 1). (b) The first goal is to optimize the frontal model f_1 so that z is less activated given the input “cat". (c) The second goal is to optimize the followed model f_2 so that a correct output for “cat" does not solely rely on z. (d) Zoom in on the original model at neuron z. (e) Naive solution that sets z' to a constant 1 without gradient. (f) Naive solution that decreases the value of z to $\bar{z}$ with a constant 6 without gradient. (g) Reverse Deactivation that first reverses z then pushes the output value to $\bar{z}$ by adding a constant 8 without gradient. (h) All the methods can achieve the second goal by outputting a value $\mathcal{V}(z') = \bar{z}$ to f_2. As $\bar{z}$ provides little information, f_2 must learn to rely on other neurons. (i) When calculating the gradient, f_1 will find that z is too small and tends to increase it (e.g., from 1 to 2). (j) Naive method (a) cannot update related parameters without gradient. (k) Naive method (b) further increases the underlying z activation (7 to 8). (l) Reverse Deactivation inherently deactivates z (from 7 to 6). When a new batch arrives, the updated z^* activates less (=6) for “cat" compared with z.

The output of f_1 without post deactivation (i.e., $z - \delta(z - \bar{z})$) is closer to $\bar{z}$ with a factor $1 - \delta$:

$$\frac{z - \delta(z - \bar{z}) - \bar{z}}{z - \bar{z}} = 1 - \delta,$$

which achieves deactivation (Figure 3(j)). The formal and detailed theory is presented in the following lemma.

Lemma 3.3. *Given a trained model f with a continuous second derivative and a Lipschitz continuous gradient, where f achieves a desired output $\mathbf{o}$ with minimal loss $\mathcal{L}(\mathbf{o})$, in which $\mathbf{o} = f(\mathbf{x}) = f_2(f_1(\mathbf{x}), \mathbf{x}) = f_2(\mathbf{z}, \mathbf{x})$ for input $\mathbf{x}$ based on its monosemantic neuron z in layer $\mathbf{z}$, suppose that $\mathcal{L}(f_2(\cdot))$ monotonically increases with $|z' - z|$ for any other value z' that replaces z. Then, with a sufficiently small learning rate l, by updating the model f with gradient descent based on the neuron processed by the RD method, the activation of z on input $\mathbf{x}$ can be inhibited.*

The proof is given in Appendix D.2 in technical report [1].

Additionally, we conduct validation experiments to compare the optimization outputs based on different methods in subsection 4.3. The results are consistent with the theory and the analysis of naive methods and RD, showing that RD is effective in inhibiting monosemanticity.

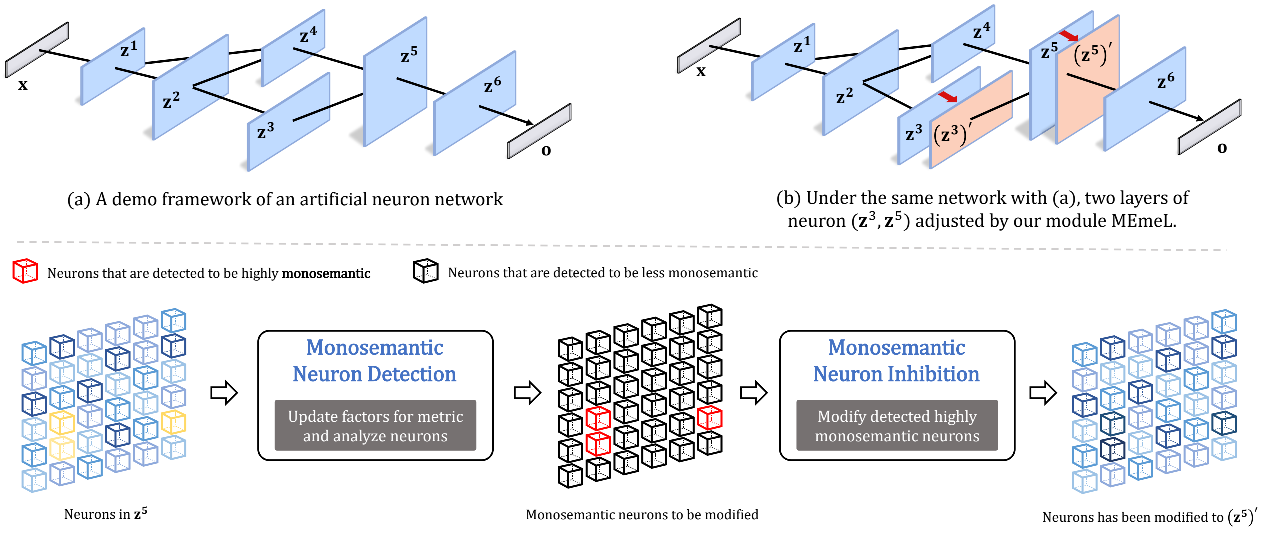
3.3 Flexible Plug-in Module

In this subsection, we demonstrate the implementation of our method, which can be inserted after any neuron layer to inhibit its monosemanticity for emergence induction. We name our module **M**onosemanticity-based **Em**ergence **L**earning (MEmeL for short) and outline the advantages of MEmeL:

- Adaptivity: Our module is compatible with any design of framework, and no structural changes are needed after inserting it.
- Light weight: No additional trainable parameters are introduced.

The details of MEmeL are displayed in Figure 4. We present a general framework of a neural network in Figure 4(a). The output $\mathbf{o}$ is derived from $\mathbf{x}$ based on the layers of neurons $\{\mathbf{z}\}$. Yellow arrows indicate the reliance of neuron layers on each other. Our method can be applied to any layer of neurons, such as $\mathbf{z}^3$ and $\mathbf{z}^5$ in Figure 4(b), where the original neurons are adjusted by our modules to inhibit monosemanticity.

Taking $\mathbf{z}^5$ as an example (Figure 4(c)), we first detect monosemantic neurons using our MS metric, as introduced in subsection 3.1. After identifying these neurons (colored red), we apply our Reverse Deactivation method, as described in subsection 3.2, to each of them.


x
z1
z2
z3
z4
z5
z6
o
x
z1
z2
z3
(z3)′
z4
z5
(z5)′
z6
o
Neurons that are detected to be highly monosemantic
Neurons that are detected to be less monosemantic
Monosemantic Neuron Detection
Update factors for metric and analyze neurons
Monosemantic Neuron Inhibition
Modify detected highly monosemantic neurons
Neurons in z5
Monosemantic neurons to be modified
Neurons has been modified to (z5)′

(a) A demo framework of an artificial neuron network

(b) Under the same network with (a), two layers of neuron ($\mathbf{z}^3, \mathbf{z}^5$) adjusted by our module MEmeL.

(c) The functionality insight of MEmeL under $\mathbf{z}^5$ as an example

Figure 4: Overview of our method: (a) An arbitrary neural network framework. x represents the input and o represents the output. zs represent hidden layers of neurons, and arrows indicate the dependency relationships. (b) Our module is inserted after z^3 and z^5, requiring no changes to the framework. (c) Details of our module applied to z^5. The input neurons are first analyzed using our metric. Once monosemantic neurons are identified, they are inhibited using RD. The resulting processed layer has the same shape as the input.

For adaptability, our module (i) does not require a specific input format and (ii) outputs $(\mathbf{z}^5)'$ in the same shape as the input $\mathbf{z}^5$, ensuring format consistency during data propagation. Thus, no adjustments to the framework are needed to apply our module.

For lightweightness, neither of our two methods introduces any trainable parameters. Additionally, MEmeL focuses on presenting the idea of functionality **induction**. The methods we propose do not directly decrease monosemanticity during training, but instead induce the model to do so, supported by theoretical analysis. Once we have a well-trained model, the performance is expected to be robust even without induction. Therefore, during testing, the module can be directly removed without incurring any inference overhead. The results of the validation experiments also support our analysis of removing MEmeL during testing. (see Appendix C.1).

The detailed algorithm is provided in the Appendix A.2. We also apply two simple tricks for implementation: late start and variance compensation, to avoid unstable value fluctuations during the cold start.

Last but not least, we emphasize the proposition of the pipeline for emergence-based learning. Researchers can focus on separable directions: (1) improving the metric to better detect factors related to scale change, and (2) designing factor-oriented solutions for better performance.

4 Experiments

In this section, we first introduce the basic experimental setup in subsection 4.1, then present the main empirical studies in subsection 4.2. We show a case study that the proposed Reverse Deactivation is more powerful in inhibition monosematicity than the naive methods in subsection 4.3. Furthermore, we discuss the potentials and limitations of MEmeL in subsection 4.4. Note the experiment about the impact of removing MEmeL during test is provided in Appendix C.1 due to the limited space.

4.1 Experimental Setup

4.1.1 Data Sets, Base Models, and Tasks. To validate the conjecture of monosemantic neurons, our inhibition method, and corresponding theories, we apply MEmel to milestone models such as Bert, Transformer, and ConvRNN on various tasks (e.g., language, image, and physics simulation), respectively.

- Language Task. We apply MEmeL to the BERT (Pre-training of Deep Bidirectional Transformers) model [8] on the GLUE (General Language Understanding Evaluation) benchmark [37]. BERT utilizes a transformer architecture and is pretrained on a large corpus of text data. It excels at capturing context and generating high-quality representations of words and sentences. GLUE serves as a benchmark for natural language understanding tasks, encompassing various tasks such as natural language inference, sentiment analysis, and similarity analysis.
- Image Task. We conduct experiments on the Swin-Transformer model [20] and ImageNet dataset [7]. The Swin-Transformer is a transformer model that uses a hierarchical structure and shift-based windows to capture cross-window connections. Our experiment follows their approach, which involves fine-tuning the Swin-Transformer on ImageNet-1K using checkpoints pretrained on ImageNet-22K. ImageNet-1K is a classification benchmark with 1,000 classes, consisting of 1.28 million training images and 50,000 validation images.

Table 1: Results on GLUE Test datasets. We follow the setting of BERT to demonstrate results on 8 datasets and calculate the average score. The scores are F1 scores for QQP and MRPC, Spearman correlations for STS-B, and accuracy scores for the other tasks. All metrics are the larger the better with best results in bold font.

Model	MNLI-(M/MM)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
Original	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
Naive (a)	84.3/83.6	**71.7**	90.6	**93.8**	52.1	85.8	88.2	66.4	79.6
Naive (b)	84.7/**84.1**	71.6	90.6	93.6	51.8	86.5	87.2	68.0	79.8
MEmeL	**84.8**/83.9	**71.7**	90.9	93.6	54.5	**86.6**	87.6	**68.2**	80.2
MEmeL-Tune	**84.8**/83.9	**71.7**	**91.2**	93.7	**55.7**	**86.6**	**89.0**	**68.2**	**80.5**

Table 2: Results on ImageNet-1k dataset, where 3 sizes of Swin-Transformer pretrained on ImageNet-22k are used as backbones. The metric used is top-1 accuracy, where a higher value indicates better performance. The best results are indicated in bold font.

Model Size	Swin-T 28M	Swin-S 50M	Swin-B 88M
Original	80.9	83.2	85.1
Naive (a)	81.0	83.4	84.6
Naive (b)	81.0	83.4	85.1
MEmeL	**81.1**	83.4	85.1
MEmeL-Tune	**81.1**	**83.5**	**85.2**

Table 3: Results on HKO-7 dataset. We initially trained a ConvGRU model for 20k steps to create the base model. The metrics used are B-MSE and B-MAE, where a smaller value indicates better performance. The best results are in bold fonts. We repeated each experiment three times and reported the average scores.

Model	B-MAE	B-MSE
Original	1003.41	309.96
Naive (a)	1003.56	309.83
Naive (b)	1003.40	310.13
MEmeL	1003.25	309.94
MEmeL-Tune	**998.81**	**298.16**

- Physics Simulation Task. We apply our MEmeL to the ConvGRU model on the HKO-7 dataset [28], which forecasts precipitation based on images of radar echoes [27]. ConvGRU is a lightweight module that belongs to the ConvRNN class, a milestone structure that combines CNN and RNN to capture spatiotemporal correlations simultaneously. HKO is a large-scale benchmark dataset from the Hong Kong Observatory (HKO), providing high-resolution radar images spanning multiple years. Generally, precipitation forecasting is a challenging task, for both theory-driven and data-driven approaches, as it exhibits complex chaotic dynamics [38, 40].

The configuration of above base models generally follow their original settings (see more details in Appendix E.1 in technical report [1]). The metrics can be found in Appendix A.1. We emphasize that our method is a general module applicable to models of any scale. After applying our module, the effectiveness of MEmeL can be evaluated by comparing it with the original model.

4.1.2 Hyper-parameter Setting. At the beginning of the experiment design, we tend to focus on the performance improvement brought by introducing reverse deactivation into milestone models. To fairly validate the influence of the proposed MEmeL with other deactivation approaches (Naive (a) and (b)), we deactivate the neuron with **only top-**1 MS in each batch (recorded as MEmeL vs. Naive(a) and (b)). During further discussion, people raised interest about the potential of tuning and we also display the results with tuned hyper-parameters (recorded as MEmeL − Tune). See more details in Appendix E.3 in the technical report [1].

4.1.3 Implementation. Our experiment is conducted on 4 V100 GPU cards. All the codes are implemented in PyTorch [2], which are given in the link https://github.com/dominatorX/MEmeL-code.

4.2 Main Experiment Result

Table 1, Table 2, and Table 3 demonstrate the performance of MEmeL incorporated with BERT, Swin-Transformer, and ConvGRU, respectively. The "Original" method denotes the raw model, "Naive (a)" and "Naive (b)" are corresponding to the original model incorporated with naive inhibition methods (see details in Equation (10)). By comparing the original method and the method enhanced with MEmeL, we can see that the MEmeL (especially MEmeL-Tune) achieves better or comparable results on different tasks, different data sets, and different base models. We also conduct the paired t-test on all three datasets to verify that the improvements are statistically significant (see Appendix C.2). This validates that neural networks can achieve better results by deactivating monosemantic neurons. By comparing Naive (a), (b) with MEmeL (and MEmeL-Tune), we can easily observe that naive deactivation methods may intensify the monosemanticity of neural networks, which may lead to inferior performance improvement as discussed in subsection 3.2.1.

4.3 The Effectiveness of Inhibition

To validate that the monosemanticity is indeed inhibited by our reverse deactivation, we conducted experiments on the ImageNet-1k dataset using the Swin-Transformer model as shown in Table 4. We

Table 4: Validation experiments conducted on the Swin-B model. We record the Decrease Ratios and Update Scales of 10k neurons. The model that utilizes our Reverse Deactivation method is compared with those using two Naive methods and the original Swin-B.

Methods	Original	Naive (a)	Naive (b)	Reverse Deactivation
Average Decrease Ratio	0.003%	-0.017%	-0.044%	**0.013%**
Average Total Update Ratio	0.052%	0.118%	0.161%	0.189%

collected 10k samples for 4 different settings, where the modification of z was done using Original (no modification), and two naive methods (a) and (b), and Reverse Deactivation in subsection 3.2.2.

We feed inputs $\mathbf{x}$ to the model again to check how z is optimized after updating the model with $\mathbf{x}$. The new value is denoted as z'. Then, the Decrease Ratio represents how the monosemanticity is decreased upon z: *Decrease Ratio* $= (1 - \phi(z')/\phi(z)) \times 100\%$. Without any modification, monosemanticity showed a slight decrease with a small positive average decrease ratio (0.003%) in the original model. Naive methods (a) and (b) intensify monosemanticity since their decrease ratios are negative, which is consistent with our analysis in Sec. 3.2.1. On the contrary, reverse deactivation has the most significant impact on decreasing monosemanticity (0.013%). But the value is relatively small. That is mainly because the small learning rate (2×10^{-6}) usually leads to a small update step in training. To provide a clear illustration, we further define the Total Update Ratio as *Total Update Ratio* $= |z'/z - 1| \times 100\%$. The scale of modification on z and on $\phi(\cdot)$ is compatible (e.g., 0.013% versus 0.189% for RD). Thus, the small scale of the Decrease Ratio is due to the small learning rate, indicating a stable impact on the inhibition of monosemanticity using our method.

4.4 Potential and Limitation of MEmeL

According to our hypothesis, MEmeL induces the model to accumulate general and abstract functionality instead of monosemanticity for a specific task, which is consistent with the goal of per-taining. Although MEmeL achieves good results during fine-tuning (demonstrated at Main Experiments in subsection 4.2), the improvement is expected to be even greater when it is applied to the pre-training phase.

The main obstacle is the high computational resource cost required for pre-taining. Currently, we have only completed validation on relatively small precipitation forecasting datasets. In Table 5, the model pre-trained with MEmeL (P-MEmeL) outperforms the one without it (P-Original). Based on these two models, the same finetuning process is conducted without MEmeL. Using MEmeL during pretraining improves B-MAE by 0.18% and B-MSE by 1.29% (T-MEmeL). In contrast, using MEmeL during finetuning improves B-MAE by 0.02% B-MSE by 0.01% (MEmeL in Table 2). The results validate our hypothesis.

5 Conclusion

Different from the literature, we hypothesize a key factor that highly promote the performance of large neural networks: the reduction of monosemantic neurons. There is no unified quantitative evaluation metric and simply banning monosemantic neurons does not promote polysemanticity in neural networks. Therefore, we propose to learn from emergence and present a study on proactively inhibiting the monosemantic neurons in this paper. More specifically, we first propose a new metric to measure the monosemanticity of neurons with the guarantee of efficiency for online computation, then introduce a theoretically supported method to suppress monosemantic neurons and proactively promote the ratios of polysemantic neurons in training neural networks. We validate our conjecture that monosemanticity brings about performance change at different model scales on a variety of neural networks and benchmark datasets in different areas, including language, image, and physics simulation tasks. Further experiments validate our analysis and theory regarding the inhibition of monosemanticity.

Table 5: Validation experiments conducted on the HKO-7 dataset. In addition, we pretrain a ConvGRU model for 20k steps using MEmeL, labeled as "P-MEmeL". The "P-Original" model is pretrained based on the original model and is used in the main experiment. Based on these two models, we perform finetuning for 2k steps using the original model, labeled as "T-MEmeL" and "T-Original", respectively. The metrics used for evaluation were B-MSE and B-MAE, where a smaller value indicates better performance. The best results are shown in bold fonts.

Model	B-MAE	B-MSE
P-Original	1004.25	311.54
P-MEmeL	**1000.98**	**306.67**
T-Original	1003.41	309.96
T-MEmeL	**1001.56**	**305.96**

Unfortunately, extending this research to very large data sets or models (e.g., large language models) is appealing yet impossible for research departments due to limited resources. Therefore, we are trying to find ways to extend this paper to extremely large models trained on large data sets as future work.

Acknowledgments

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Guangdong Province Science and Technology Plan Project 2023A0505030011, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant and HKUST-Webank joint research lab grants.

References

[1] 2024. *Technical report of our paper.* https://github.com/dominatorX/MEmeL-code/blob/main/EmeL_tech_report.pdf

[2] Paszke Adam, Gross Sam, Chintala Soumith, Chanan Gregory, Yang Edward, DeVito Zachary, Lin Zeming, Desmaison Alban, Antiga Luca, and Lerer Adam. 2017. Automatic differentiation in PyTorch. (2017).

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Àgata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proc. Natl. Acad. Sci. USA* 117, 48 (2020), 30071–30078.

[4] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*. PMLR, 2397–2430.

[5] Olah Chris, Cammarata Nick, Schubert Ludwig, Goh Gabriel, Petrov Michael, and Carter Shan. 2020. Zoom in: An introduction to circuits. *Distill* 5, 3 (2020), e00024–001.

[6] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing Transformers in Embedding Space. In *ACL (1)*. Association for Computational Linguistics, 16124–16170.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*. IEEE Computer Society, 248–255.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[9] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer El Showk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. 2022. Softmax Linear Units. *Transformer Circuits Thread* (2022).

[10] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds Mach.* 30, 4 (2020), 681–694.

[11] Goh Gabriel, Cammarata Nick, Voss Chelsea, Carter Shan, Petrov Michael, Schubert Ludwig, Radford Alec, and Olah Chris. 2021. Multimodal neurons in artificial neural networks. *Distill* 6, 3 (2021), e30.

[12] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *EMNLP (1)*. Association for Computational Linguistics, 5484–5495.

[13] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding Neurons in a Haystack: Case Studies with Sparse Probing. *CoRR* abs/2305.01610 (2023).

[14] Simon Haykin. 1994. *Neural networks: a comprehensive foundation.* Prentice Hall PTR.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *CoRR* abs/2001.08361 (2020).

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*. 1106–1114.

[18] Anders Krogh. 2008. What are artificial neural networks? *Nature biotechnology* 26, 2 (2008), 195–197.

[19] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* 1, 4 (1989), 541–551.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*. IEEE, 9992–10002.

[21] Wolchover Natalie. 2018. New theory cracks open the black box of deep learning. (2018).

[22] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).

[23] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. 2022. Toward Transparent AI: A Survey on Interpreting the Inner Structures of Deep Neural Networks. *CoRR* abs/2207.13243 (2022).

[24] McCulloch Warren S and Pitts Walter. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5 (1943), 115–133.

[25] Andrew M. Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D. Tracey, and David D. Cox. 2018. On the Information Bottleneck Theory of Deep Learning. In *ICLR (Poster)*. OpenReview.net.

[26] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are Emergent Abilities of Large Language Models a Mirage? *CoRR* abs/2304.15004 (2023).

[27] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS*. 802–810.

[28] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. In *NIPS*. 5617–5627.

[29] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

[30] Bills Steven, Cammarata Nick, Mossing Dan, Tillman Henk, Gao Leo, Goh Gabriel, Sutskever Ilya, Leike Jan, Wu Jeff, and Saunders William. 2023. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)* (2023).

[31] Bills Steven, Cammarata Nick, Mossing Dan, Tillman Henk, Gao Leo, Goh Gabriel, Sutskever Ilya, Leike Jan, Wu Jeff, and Saunders William. 2023. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)* (2023).

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*. IEEE Computer Society, 1–9.

[33] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *ITW*. IEEE, 1–5.

[34] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023).

[35] Bricken Trenton, Templeton Adly, Batson Joshua, Chen Brian, Jermyn Adam, Conerly Tom, Turner Nick, Anil Cem, Denison Carson, Askell Amanda, et al. 2023. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread* (2023).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[37] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR (Poster)*. OpenReview.net.

[38] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu. 2017. PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In *NIPS*. 879–888.

[39] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *Trans. Mach. Learn. Res.* 2022 (2022).

[40] Zhang Xuebin, Zwiers Francis W, Li Guilong, Wan Hui, and Cannon Alex J. 2017. Complexity in estimating past and future extreme short-duration rainfall. *Nature Geoscience* 10, 4 (2017), 255–259.

[41] LeCun Yann, Bengio Yoshua, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.

[42] Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2023. The Mystery and Fascination of LLMs: A Comprehensive Survey on the Interpretation and Analysis of Emergent Abilities. *CoRR* abs/2311.00237 (2023).

Algorithm 1 Monosementicity Scale Computing with Needed Variables

```
Input: new batch of values $\left\{\mathbf{z}^{[m+1]}, \mathbf{z}^{[m+2]}, \cdots, \mathbf{z}^{[m+b]}\right\}$ of the neuron.
Local Variables: forget step $n_f$, current step $c_t$, current sample mean $\mu_m$ and variance $v_m$
Calculate the MS for each input value $z \in \mathbf{z}$: $\phi(z^{[m+i]}) = (z^{[m+i]} - \mu_m)^2 / v_m$ for $i \in [1, b]$.
Calculate $\mu'_b = \frac{1}{b} \sum_{i=1}^{b} z^{[m+i]}$ and $v'_b = \frac{1}{b} \sum_{i=1}^{b} (z^{[m+i]} - \mu'_b)^2$.
if $c_t < n_f$ then
    $\mu_{m+b} = \frac{m\mu_m + b\mu'_b}{m+b}$, $v_{m+b} = \frac{mb(\mu_m - \mu'_b)^2}{(m+b-1)(m+b)} + \frac{bv'_b + (m-1)v_m}{m+b-1}$
else
    $\mu_{m+b} = \frac{n_f\mu_m + \mu'_b}{n_f+1}$, $v_{m+b} = \frac{n_f(\mu_m - \mu'_b)^2}{(n_f+1-1/b)(n_f+1)} + \frac{v'_b + (n_f-1/b)v_m}{n_f+1-1/b}$
end if
RETURN: Monosementicity Scale of inputs $\phi(\mathbf{z})$
```

Algorithm 2 Monosementicity-based Emergence Learning

```
Input: Values of $n$ neurons with batchsize $b$: $\mathbf{z} = \{\{z_i^{[j]}\}_{j=1}^{b}\}_{i=1}^{n}$.

Local Variables: late start step $l_s$, current step $c_t$, current sample mean $\{\mu_{i_m}\}_{i=1}^{n}$ and variance $\{v_{i_m}\}_{i=1}^{n}$
if $c_t < l_s$ then
    Calculate the MS for each input value
    $$\phi(z_i^{[m+j]}) = \frac{(z_i^{[m+j]} - \mu_{i_m})^2}{v_{i_m} + \epsilon \sum_{k=1}^{n} v_{k_m}/n} \text{ for } i \in [1, n], j \in [1, b].$$
    Select values with high $\phi$, adjust each of them according to MD. Replacing the original ones in $\mathbf{z}$ to form output $\mathbf{z}'$.
else
    $\mathbf{z}' = \mathbf{z}$
end if
Update sample mean $\{\mu_{i_m}\}_{i=1}^{n}$ and variance $\{v_{i_m}\}_{i=1}^{n}$ using Algorithm 1.
RETURN: Adjusted layer of neurons $\mathbf{z}'$
```

A Implementation Details

A.1 Implementation Details of MS ϕ

During training, the model is continuously updated. However, samples from the early stages become outdated, and more importance should be given to the new samples. To address this, we introduce a forget step (n_f) and keep track of the current training steps (c_t). Once $c_t \geq n_f$, we update the sample mean and variance by replacing the number of samples from m to $b \cdot n_f$. This reduces the influence of previous samples. Both variable updating and mean and variance computation have a complexity of $O(1)$.

A.2 Implementation Details of MEmeL

As described in Algorithm 1, the statistical variables are calculated online. However, due to the limited number of samples at the beginning of training, the estimation can be unstable. Additionally, if the estimation of S^2 is extremely small, it may cause overflow during calculation. To improve the robustness of training, we introduce a late start step l_s and a variance compensation in the denominator of the metric calculation in Equation (7). As outlined in Algorithm 2, when a batch of data arrives, we only update neurons with RD if the current step is greater than l_s. The calculation of MS is also adjusted by incorporating the mean of variances of other neurons, weighted by a small value ϵ to prevent overflow.

B Related Works

B.1 Artificial Neural Networks and the Increasing Scale

Since S and Walter [24] first modeled a simple neural network, ANN has been investigated under a scale of several layers in the last century [19, 41]. With improvements in hardware, deeper models can be supported. AlexNet, which uses 8 layers for image classification, achieved excellent performance in the 2012 ImageNet challenge [17]. Later on, deeper models such as Inception and VGG increased the scale to tens and hundreds of layers [29, 32]. The design of critical modules, such as ResNet, also plays a crucial role in ensuring the stability of model training when increasing the depth [15].

After the Transformer was proposed and validated as effective in various areas [36], both its scale and performance have seen significant growth over the years [8, 20]. In recent years, architectures based on the Transformer have achieved great success at extremely large scales [10, 34]. Investigating the underlying effective properties created by the increase in scale is a highly demanded research direction.

B.2 Mechanistic Interpretability

With the improved performance of neural networks, their black-box nature raises more questions than it answers. To gain a better understanding of and diagnose neural networks, researchers seek mechanistic interpretability [23]. They study individual components to understand their functionality and usage, such as neurons for identifying dogs and cars [5]. As Transformer models demonstrate their superiority in various domains, there is an increasing focus on their interpretability. Geva et al. [12] propose that the feed-forward layers of Transformers function as key-value pairs. Dar et al. [6] extend this mapping to the embedding space. Although the complexity greatly increases with larger models, the success of these models attracts researchers who strive to find interpretability in the vast sea of neurons [13, 31, 35]. In their work, Elhage et al. [9] introduce the softmax linear unit to create monosemantic models. Additionally, Gurnee et al. [13] modify the sparse probe by using multiple heads to classify neurons with decreasing monosemanticity. Trenton et al. [35] construct more powerful and complex probes to construct mappings between neurons and features. While the desire to decompose and understand everything is appealing (e.g., achieving monosententicity), it may conflict with the progress of intelligence.

B.3 Information Bottleneck

As an important method to explain the deep learning mechanism, the Information bottleneck (IB) provides a compression view for deep learning and emphasizes forgetting together with information retaining [33]. In contrast, we emphasize the way neurons **manipulate** the retained information (distributed or one-to-one), while IB

Table 6: Validation results on GLUE Test datasets. The settings are the same with Table 1. All metrics are the larger the better with best results in bold font.

Model	MNLI-(M/MM)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
MEmeL	84.8/**83.9**	**71.7**	**90.9**	93.6	**54.5**	**86.6**	87.6	**68.2**	**80.2**
MEmeL-T	**84.9**/83.7	71.5	90.5	**93.9**	52.1	85.9	**88.7**	68.1	79.9

Table 9: The paired t-test on all 3 datasets. "Mean-Original" and 'Mean-Ours" refers to the mean scores of the two methods on each dataset. t-statistic>0 for GLUE and ImageNet as their metrics are the larger the better. t-statistic<0 for HKO-7 as its metric is the smaller the better.

Dataset	GLUE	ImageNet	HKO-7
p-value	**0.02**	**0.07**	**0.06**
t-statistic	2.77	3.46	-2.38
Mean-Original	79.6	83.1	656.69
Mean-Ours	80.5	83.3	648.49

Table 7: Validation results on ImageNet-1k dataset. See Table 2 for detailed settings. The metric is the higher the better. The best results are indicated in bold font.

Model Size	Swin-T 28M	Swin-S 50M	Swin-B 88M
MEmeL	**81.1**	83.4	**85.1**
MEmeL-T	**81.1**	**83.5**	**85.1**

Table 8: Results on HKO-7 dataset. The settings are the same with Table 3. The metrics are the smaller the better with the best results in bold fonts.

Model	B-MAE	B-MSE
MEmeL	1003.25	309.94
MEmeL-T	**1002.46**	**309.52**

emphasizes the total **amount** of information that different layers of neurons record at different training stages [25].

As the scale of neural networks continues to grow, effective information processing and reasoning have become as important a need as information compression. In addition to what information should be squeezed through the bottleneck, how to process the squeezed information is also important and is what our paper concerns [21]. Our method is a promising direction to enrich deep learning understanding together with IB, especially when large model application requires more and more powerful reasoning ability.

B.4 Debate on the Existence of Emergence

Note that Schaeffer et al. [26] states that "emergent abilities appear due to the researcher's choice of metric rather than due to fundamental changes in models with scale". Here we point out that their finding does not diminish the value of our finding, but instead partially coincides with our idea:

- Though evaluation metrics can be smooth and well-designed, models are improved based on training data. However, solving hard and realistic problems via advanced AI involves more and more data with poorly labeled or even without labels. Emergence learning is demanded to find and boost the underlying ability accumulation of models, which diminishes the correctness of individual answers and focuses on the potential knowledge learning brought by scale changes.
- To observe the accumulated ability of the model on these hard problems, we need smooth and mild metrics. Such metrics may not be available for challenging problems in the future, where the research would be like roaming at deep night. Factors discovered through Emergence Learning can help validate the potential improvement of models and enlighten the darkness.

C More Experiments

C.1 The Impact of Removing MEmeL During Test

Recall that in subsection 3.3, we emphasize the lightness of our MEmeL. It aims to induce models to reduce monosemanticity during training and can be removed during testing. Here, we conduct experiments to enable MEmeL during testing, which are expected to have similar results compared to the results displayed in the main experiments (see Section 4.2). These models are labeled with the postfix "-T" (e.g., MEmeL-T for MEmeL as the base model) while keeping other experimental settings unchanged.

The results are displayed in Table 6, Table 7, and Table 8 for GLUE, ImageNet, and HKO-7 datasets, respectively. The best results are shown in bold fonts. The performance differences for the two settings are very close on all three tasks, indicating a stable performance when tested without the induction of MEmeL. Interestingly, on GLUE datasets, the performance when using MEmeL during testing is much more fluctuated. It would be valuable to study the dynamics of MEmeL in more depth as future work.

C.2 Statistical Significance

To statistically verify the improvement from our method, we conduct the paired t-test on all three datasets. The results are given in Table. 9. Each score is obtained from the results of the **Original** and **MEmeL-Tune**. All the datasets show a 90% confidence level (p-value <0.1) supporting the hypothesis that our performance significantly differs from the baseline.