Emergence Learning: A Rising Direction from Emergent Abilities and a Monosemanticity-Based Study

Jiachuan Wang [1] **Shimin Di** [1] **Lei Chen** [1 2] **Charles Wang Wai Ng** [1 2]

Abstract

In the past 20 years, artificial neural networks have become dominant in various areas, continually growing in scale. However, the current analysis of large models has mainly focused on functionality, overlooking the influence of scale differences on their properties. To address this, we propose the concept of Emergence Learning, which emphasizes the significance of scale. By studying models of different scales, we have identified a key factor in achieving higher performance in large models: the decrease of monosemantic neurons. Building on this insight, we propose a proactive approach to inhibit monosemanticity for improved performance. Our solution involves a two-phase process that includes monosemantic neuron detection and inhibition, supported by theoretical analysis. Experimental results on various tasks and neural networks demonstrate the effectiveness of our proposed method.

Following the idea of Emergence Learning, though drawing inspiration from scaling phenomena, the applicability of our method is not restricted to large scale alone. Therefore, the experiment is self-contained. However, extending this research to very large-scale datasets is appealing yet impossible for research departments due to limited resources. We are delighted to share the first co-authorship and eagerly await collaboration from any AI company before submission.

1. Introduction

Inspired by biological neurons, the artificial neural network (ANN) was proposed in the last century (S & Walter, 1943). However, its activity diminished for decades before experiencing great success after the 2010s (Krogh, 2008; Krizhevsky et al., 2012). One of the major differences compared to previous models is the relatively large scale.

In recent years, large models have achieved remarkable results in various fields (Floridi & Chiriatti, 2020; Touvron et al., 2023). Compared to previous works, these models have much larger scales in terms of datasets, model sizes, and training quantities. Though the idea of "the larger, the better" is common sense, unexpected improvements could occur solely due to larger scales. Emergence, as an intriguing phenomenon in large-scale models, refers to the gradual improvement of model performance before the scale reaches a certain threshold, followed by a rapid enhancement once the threshold is surpassed (Wei et al., 2022). Increasing evidence suggests that the surprises we experience may not come from the functionality and mechanism, but rather from the underlying nature of scale changes. A critical question arises: people increase model scale, but increase for what?

Therefore, we propose Emergence Learning (EmeL for short), which emphasizes the property differences brought about by scale differences, and seeks to uncover potential principles to improve models.

Related to EmeL, recent research has also analyzed large models and observed changes resulting from differences in scale (Geva et al., 2021b; Gurnee et al., 2023). However, the research paradigm following smaller-scale models hinders a more in-depth investigation. A potential case we found is the study on monosemantic and polysemantic neurons. Through statistical analysis on the relationships between neurons and input features, a neuron is considered monosemantic if it forms a 1-1 correlation with its related input feature (Bau et al., 2020; Elhage et al., 2022) (Figure. 1(b)). In contrast, polysemantic neurons activate for several features that are weakly correlated (Trenton et al., 2023; Chris et al., 2020; Gabriel et al., 2021; Elhage et al., 2022) (Figure. 1(c)).

Recent studies have shown that monosemantic neurons are sparser in larger models and require more complex methods to detect (Gurnee et al., 2023; Trenton et al., 2023). Experimental results discussed in Appendix. A reveal more deficits in monosemantic neurons, e.g., when an input's

[1]The Hong Kong University of Science and Technology, Hong Kong SAR, China [2]The Hong Kong University of Science and Technology (Guangzhou), Guangdong Province, China. Correspondence to: Shimin Di <sdiaa@connect.ust.hk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

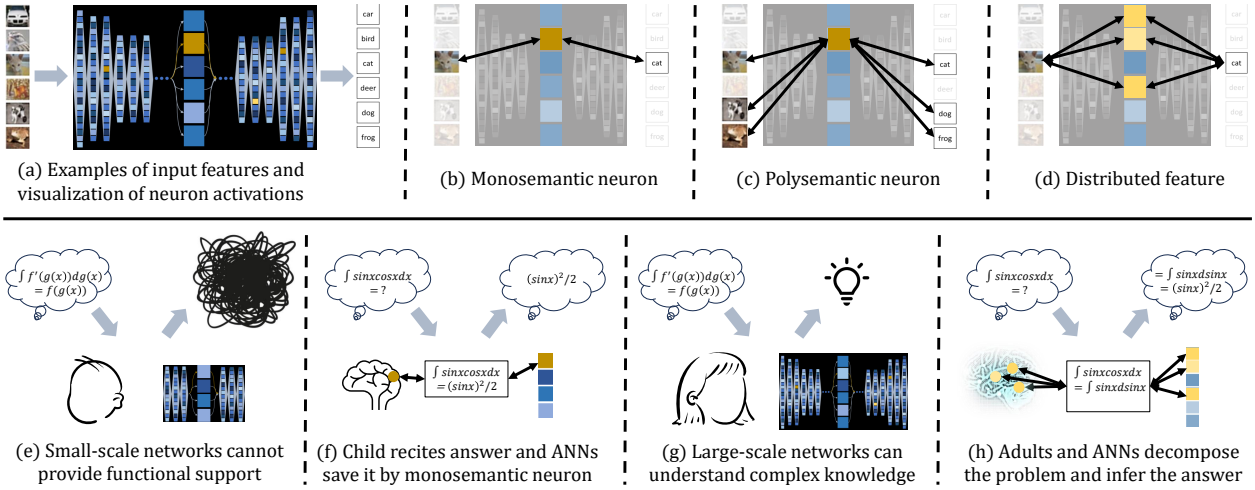


Figure 1. Demonstration of important concepts and motivation: (a) A multilayer neuron network for the image classification task. The middle layer is highlighted. (b) A monosemantic neuron (orange) ideally activates for one specific type of feature. (c) A polysemantic neuron activates for multiple features. (d) A distributed feature is classified based on the activation of multiple neurons. (e) ANNs could be similar to biological brains. Small-scale networks (both biological and artificial) cannot support complex functionality. (f) To solve a difficult problem that requires complex reasoning, pupils rely on rote memorization, while ANNs store QA pairs using monosemantic neurons as key-value pairs. (g) Large-scale networks can learn and master solving skills related to integration. (h) Adults and large-scale ANNs are capable of decomposing features and inferring answers using complex neuron circuits. This ability reduces reliance on rote memory and monosemanticity.

related monosemantic neuron is disabled, smaller models incur much larger errors compared to larger models (Gurnee et al., 2023). Those results indicate a positive relationship between the decrease of monosemanticity and the increase of model scale, which is highly likely to result in good performance.

However, following the favor of monosemantic neurons from studies on small-scale models, existing works focus on enhancing or extracting monosemanticity. In their work, Elhage et al. (2022) introduce the softmax linear unit to create monosemantic models. Additionally, Gurnee et al. (2023) modify the sparse probe by using multiple heads to reduce monosemanticity. To provide interpretability in the feed-forward layer, key-value pairs are constructed, aiming to map each key to identifiable patterns (Geva et al., 2021b). Following the paths of small-scale models, these works overlook the potential for improvement direction behind the harmfulness of monosemanticity.

Therefore, we propose our hypothesis and question:

- **Hypothesis**: As the model scale increases, the decrease of monosemantic neurons is a key factor in achieving higher performance.

- **Question**: Can we proactively induce the decrease of monosemantic neurons to achieve high performance?

An example is given below to illustrate our idea from the angle of EmeL.

- From the perspective of scale differences, when the model is small, the limited number of neurons makes it difficult to abstract and analyze the input, and to provide functional support for integrating a large amount of scattered information (Figure. 1(e)). To achieve good results, neurons tend to specialize in certain features, resulting in stronger monosemanticity (Figure. 1(f)).

  We can analogize a small neural network to a developing brain of a pupil, memorizing question-answer pairs. Constructing monosemantic neurons is easier to achieve good results in complex tasks (such as accurately answering ten complex integration problems), compared to expecting the pupil to understand the solving steps and reason the results by themselves. (Not to mention that providing datasets with solving steps is very rare in neural network training, and small models struggle to automatically acquire reasoning abilities based solely on input-output pairs through simple gradient descent).

  Similarly, we can analogize large models to adults, allowing them to learn and master solving skills related to integration (Figure. 1(g)). This is because adults have accumulated knowledge and have a more ma-

ture brain development, similar to the huge training volume and parameter size of large models. In this process, adults gradually reduce the reliance on rote memorization and improve their reasoning abilities for new problems (Figure. 1(h)). This can explain why large models have fewer monosemantic neurons and stronger generalization abilities in complex tasks.

- To utilize this in model optimization, one issue is that existing large models are **passive** in decreasing monosemanticity, which can be likened to students lacking systematic education. Large models gradually reduce monosemanticity by accumulating training data, similar to students acquiring skills through extensive learning rather than rote memorization, and good guidance can help students improve faster. This leads us to our question: Can we **proactively** guide the reduction of monosemantic neurons to improve model performance?

Ideally, models will improve regardless of their scales. This is because large models also have the potential to decrease monosemanticity, resulting in better results.

To actively guide the reduction of monosemanticity, we propose a 2-phase solution: (i) monosemantic neuron detection and (ii) monosemantic neuron inhibition. Existing works build feature datasets and detect neuron activation when input contains specific features to find monosemantic neurons (Gurnee et al., 2023; Trenton et al., 2023; Steven et al., 2023b). However, these methods have efficiency and flexibility issues. Instead, we provide an effective online metric for detection. To inhibit monosemantic neurons, we identify the drawbacks of simple methods and propose a theoretically supported Reverse Deactivation method to modify monosemantic neurons, guiding the model to reduce monosemanticity during training. Our approach can be integrated as a parameter-free and flexible insertion module called MEmeL, 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 language, image, and physics simulation tasks, applying MEmeL to milestone models such as Bert and ConvRNN. The architectures covered Transformer, CNN, and RNN. The experimental results demonstrated the effectiveness of our method.

We summarize our contribution as follows:

- We raise the concept of "Emergence Learning", though far behind solving it, referring to a more recent research direction aside in middle- to large-scale models, in which patterns of neurons hardly appear in small models. The target is to identify factors related to the difference of scales and design specific solutions to induce the occurrence of emergence or improve performance.
- To validate our hypothesis that the decrease of monosemanticity could be a key factor in high performance, we propose an efficient and effective method to detect monosemanticity.
- We design solution RD to inhibit monosemanticity with theoretical guarantee, which overcomes an inherent drawback of naive solutions.
- We conduct experiments in various tasks with different types of neural networks to validate the effectiveness of our method.

  (We are exploring our computational resources to conduct larger-scale experiments.)

The rest of the paper is organized as follows: Section 2 provides the formal problem definition. Section 3 introduces our method for decreasing monosemanticity. Experimental results are presented in Section 4. We review related works in Section 5. Finally, we conclude our paper in Section 6.

2. Problem Definition

In this section, we will first introduce the basic concepts. Then, we will provide formal definitions for two problems towards our goal: the detection and inhibition of monosemantic neurons.

2.1. Basic Concepts

We define a deep learning model as $\mathcal{F}$, with input and output represented as X and Z, respectively. The relationship between the two is given by the equation:

$$Z = \mathcal{F}(X).$$

The goal of training model $\mathcal{F}$ is to update its parameters so that the output Z becomes closer to a provided or unknown ground truth Y.

Deep learning models are composed of multiple layers, with each layer containing multiple neurons. We denote a specific layer of neurons as $H = \{h_1, h_2, \cdots\}$. In addition to existing methods for model optimization, we considering two new targets during training to improve performance:

- Detection of monosemantic neurons of H.
- Inhibition of monosemantic neurons of H.

2.2. Monosemanticity Detection

To detect monosemantic neurons, existing methods are inflexible as they require a predefined and manually labeled

feature dataset (Gurnee et al., 2023; Trenton et al., 2023). Moreover, detection methods like probes necessitate training and offline inference for statistical analysis, which can be costly for online training. To address these limitations, there is a need for a robust metric ϕ that can accurately detect monosemantic neurons. This metric should fulfill the following criteria:

- Generality. Does not rely on any specific dataset.
- Efficiency. Fast online detection during training.

2.3. Monosemanticity Inhibition

To inhibit a detected monosemantic neuron $h \in H$, we need to design the optimization strategy ω for $\mathcal{F}$. Without loss of generality, when we consider a particular layer of neurons H, we can divide the model F into a frontal model f_1 and a followed model f_2, so that:

$$H = f_1(X), Z = f_2(H, X).$$

The problem can be solved in two phases:

- Optimize the frontal model f_1:

  $$\omega(f_1) \to f_1^*,$$

  so that the neuron h is less activated for input X:

  $$\phi(h^*) < \phi(h),$$

  where $h \in H = f_1(X)$ and $h^* \in H^* = f_1^*(X)$.
- Optimize the followed model f_2:

  $$\omega(f_2) \to f_2^*,$$

  so that the output is still robust when neuron h is deactivated. Formally, replace $h \in H$ with a weakly activated value h' to obtain H', we expect:

  $$\mathcal{L}(f_2^*(H', X)) < \mathcal{L}(f_2(H', X)),$$

  where $\mathcal{L}$ is the loss function, $\phi(h') < \phi(h)$.

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

3.1. Metric for Monosemanticity

As we want to inhibit monosemanticity during training, designing a metric that is general for different tasks and efficient to calculate is important. In this subsection, we first give the definition of our proposed metric. Then we analyze its advantages.

3.1.1. Metric Definition

Formally, we define our metric Monosemantic Scale (MS for short):

Definition 3.1 (Monosemantic Scale)**.** Given a neuron h, we denote its historical samples under m inputs as $\{h_{(1)}, h_{(2)}, \cdots, h_{(m)}\}$ and new value under the $(m+1)^{th}$ input as h_{m+1}. The Monosemantic Scale is defined as

$$\phi(h_{(m+1)}) = \frac{(h_{(m+1)} - \bar{h})^2}{S^2}, \tag{1}$$

where

$$\bar{h} = \frac{\sum_{i=1}^{m} h_{(i)}}{m}, S^2 = \frac{\sum_{i=1}^{m}(h_{(i)} - \bar{h})^2}{m-1},$$

are the sample mean and sample variance, respectively.

3.1.2. Metric Analysis

In this subsection, we state the effectiveness of our metric. Let's begin by discussing two observations of monosemantic neurons.

- High deviation of activation value. Qualitatively, a neuron is considered "activated" when its value for the current input deviates from the mean. For a monosemantic neuron, its value distribution is more skewed and incurs larger deviation. However, strictly defining "monosemantic" and "activated" is still under discussion in quantitative analysis (Elhage et al., 2022).
- Low frequency of activation. 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.

The high deviation of the activation value ensures that the term $(h_{(m+1)} - \bar{h})^2$ in ϕ can effectively filter neurons with high monosemanticity. The low frequency of activation ensures that deviated values, when the neuron activates, hardly influence the value $\bar{h}$. $\bar{h}$ is mainly determined by the values when the neuron is deactivated.

On the other hand, our target only requires finding neurons that are monosemantic, whereas existing works need to first find the neuron-feature relationships (Gurnee et al., 2023; Trenton et al., 2023). Our solution focuses on handling monosemanticity and relaxes the requirement to find their corresponding features, eliminating the need for a predefined feature dataset.

3.1.3. Metric Computing

Computing metric ϕ online is nontrivial due to the involvement of historical samples. Here we show that for each

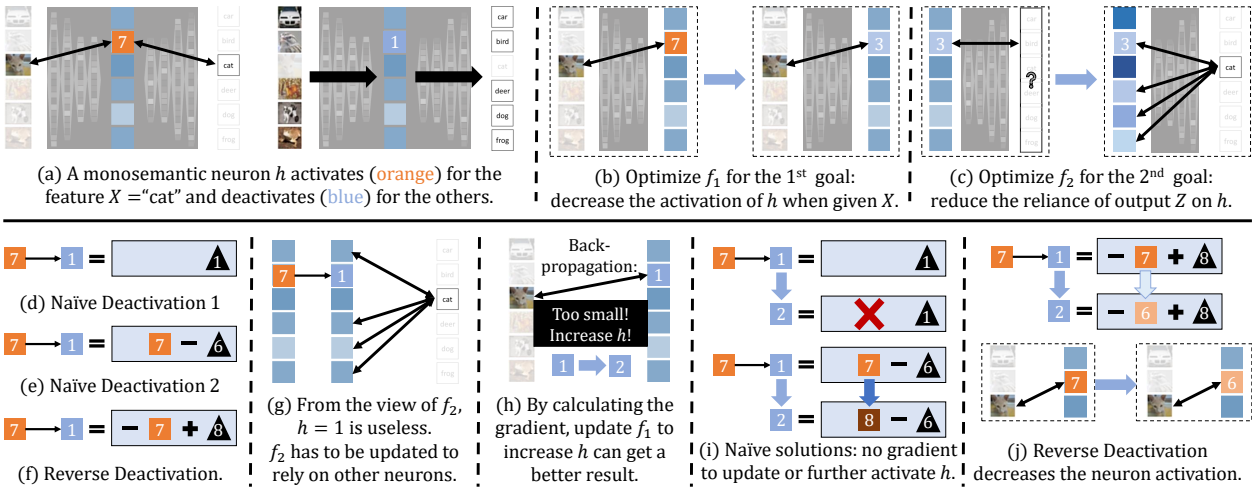


Figure 2. Illustration of problems and solutions to inhibit monosemanticity. (a) A monosemantic neuron h only activates (orange) for the feature "cat" with a high mean value ($= 7$). h is deactivated (blue) for other inputs with a small mean value ($\bar{h} = 1$). (b) The first goal is to optimize the frontal model f_1 so that h 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 h. (d) Naive solution that fixes h to a constant 1 without gradient. (e) Naive solution that decreases h to $\bar{h}$ with a constant 6 without gradient. (f) Reverse Deactivation that first reverses h then pushes the output value to $\bar{h}$ by adding a constant 8 without gradient. (g) All the methods can achieve the second goal by outputting $\bar{h}$ to f_2. As $\bar{h}$ provides little information, f_2 must learn to rely on other neurons. (h) When calculating the gradient, f_1 will find that h is too small and tends to increase it (e.g., from 1 to 2). (i) Naive method 1 cannot be updated without gradient. Naive method 2 further increases the underlying h activation (7 to 8). (j) Reverse Deactivation inherently deactivates h (from 7 to 6). When a new batch arrives, h activates less for "cat".

neuron, our proposed MS ϕ 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 $\bar{h}$ given m samples, while υ_m as the sample variance S^2. When the $(m+1)^{th} \sim (m+b)^{th}$ samples $h_{(m+1)} \sim h_{(m+b)}$ come, one can obtain the updated values via:*

$$\mu_{m+b} = \frac{m\mu_m + b\mu'_b}{m+b}, \tag{2}$$

$$\upsilon_{m+b} = \frac{mb(\mu_m - \mu'_b)^2}{(m+b-1)(m+b)} + \frac{b\upsilon'_b + (m-1)\upsilon_m}{m+b-1}, \tag{3}$$

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

The proof is given in Appendix. B and ϕ can be directly derived using Equation. (1).

In the implementation, we introduce a forget mechanism as the model is updating, where the influence of previous samples should decay. For details, please refer to Algorithm. 1 in Appendix. D.

3.2. Inhibition of Monosemanticity

Recall that given a monosemantic neuron h and an input X that contains its exclusive feature, we aim to optimize the model in two aspects: (1) decrease the activation level of h when given X (Figure. 2(b)); and (2) reduce the reliance of output Z on the activation of h (Figure. 2(c)).

In this subsection, we first demonstrate that directly deactivating monosemantic neurons can instead **enhance** monosemanticity. To address this unexpected phenomenon, we propose a method called Reversed Deactivation and provide the theoretical analysis.

3.2.1. Naive Deactivation

Without loss of generality, let's focus on a single layer of neurons H and a single neuron $h \in H$ that exhibits high monosemanticity. As defined in subsection. 3.1, such a neuron has a large $\phi(h) = (h - \bar{h})^2/S^2$, which is expected to be inhibited during model optimization. We can divide the optimization target into two parts: the frontal model f_1 before the layer, and the followed model f_2 after the layer.

From the perspective of information theory, suppose that the distribution of neuron values follows a normal distribution, a neuron provides the least amount of information when its value equals its statistical mean ($\min_h[I(h)] =$

$\min_h[-\log(P(h))] = -\log(P(\bar{h}))$, which is also its most inactive state. Thus, one can find two naive solutions to deactivate a neuron by replacing the original h with h':

$$(a)h' = \bar{h}_{ng} \tag{4}$$

$$(b)h' = h + (\bar{h} - h)_{ng}, \tag{5}$$

where subscript $\cdot_{ng}$ refers to "no gradient" scalar without trainable parameter. Two examples are given in Figure. 2(d,e)

Both solutions ensure that f_2 receives an updated H', in which the value of $\mathcal{V}(h') = \bar{h}$ provides little information. As h' is valueless, f_2 has to adjust its parameter to utilize information from other neurons in H for good output $Z' = f_2^*(H', X)$. Such a process achieves our second goal of reducing the dependence of output Z on the activation of h (Figure. 2(g)).

However, neither of the two solutions can achieve the first goal (i.e., training f_1 to inhibit the activation of h for a single feature). To be more specific, method (a) wastes all the gradient for f_1 in generating h, preventing the related parameters from being updated. Method (b) outputs $\mathcal{V}(h') = \bar{h}$ by compensating for the gap. As the model obtains a good result with h, by calculating the gradient, f_1 will be updated to push its output value from $\bar{h}$ to h, expressed as $\bar{h} + \delta(h - \bar{h})$, where $\delta > 0$ up to the learning rate (Figure. 2(h)). Ironically, the actual output of h is updated to $h + \delta(h - \bar{h})$, deviating from $\bar{h}$ by a factor of $1 + \delta$:

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

Therefore, these two simple solutions either contribute nothing to the deactivation of h or even enhance its activation (Figure. 2(i)).

(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 h with h' (Figure. 2(f)):

$$h' = -h + (\bar{h} + h)_{ng}. \tag{6}$$

Similar to baselines, RD also ensures the second goal of decreasing the dependence of output Z on the activation of h. To be more specific, f_2 receives a value $\mathcal{V}(h') = \mathcal{V}(-h + (\bar{h} + h)_{ng}) = \bar{h}$, which provides little information and requires f_2 to learn the information from other neurons in H (Figure. 2(g)).

Besides, RD can inhibit the activation level of h when given X. In short, after calculating the gradient, f_1 will update the trainable parameter to push its output value from $\bar{h}$ to h (Figure. 2(h)). Different from method (b), the gradient on h is reversed. To achieve the same shifted value $\bar{h} + \delta(h - \bar{h})$ mentioned above, an insight into the update can be expressed as:

$$\mathcal{V}(-\boldsymbol{h} + (\bar{h} + h)_{ng}) = \bar{h} \tag{7}$$

$$\rightarrow \mathcal{V}(-(\boldsymbol{h - \delta(h - \bar{h})}) + (\bar{h} + h)_{ng}) = \bar{h} + \delta(h - \bar{h}) \tag{8}$$

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

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

which achieves deactivation (Figure. 2(j)). The formal and detailed theory is presented in the following lemma.

Lemma 3.3. *Given a trained model $\mathcal{F}$ with 2 continuous derivatives and a Lipschitz continuous gradient, where $\mathcal{F}$ achieves a desired output Z with minimal loss $\mathcal{L}(Z)$, in which $Z = \mathcal{F}(X) = f_2(f_1(X), X) = f_2(H, X)$ for input X based on its monosemantic neuron h in layer H, suppose that $\mathcal{L}(f_2(\cdot))$ monotonically increases with $|h' - h|$ for any other value h' that replaces h. Then, with a sufficiently small learning rate l, by updating the model $\mathcal{F}$ with gradient descent based on the neuron processed by the RD method, the activation of h on input X can be inhibited.*

The proof is given in Appendix. C

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 **Eme**rgence **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. 3. We present a general framework of a neural network in Figure. 3(a). The output Z is derived from X based on the hidden states H_i. Yellow arrows indicate the reliance of neuron layers on each other. Our method can be applied to any layer of neurons, such as H_3 and H_5 in Figure. 3(b),

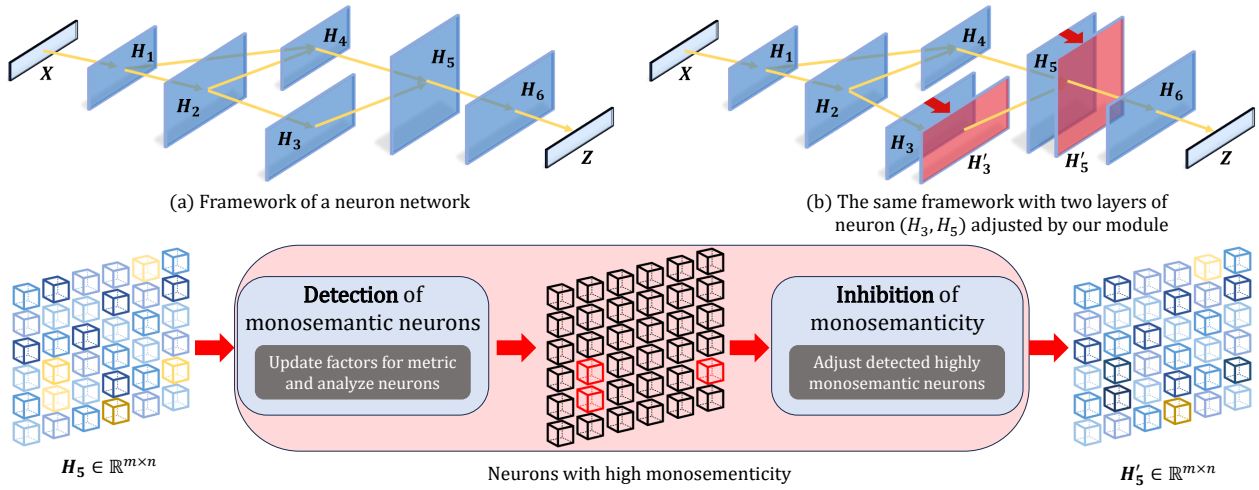


Figure 3. Overview of our method: (a) An arbitrary neural network framework. X represents the input and Z represents the output. H_is represent hidden layers of neurons, and arrows indicate the dependency relationships. (b) Our module is inserted after H_3 and H_5, requiring no changes to the framework. (c) Details of our module applied to H_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.

where the original neurons are adjusted by our modules to inhibit monosemanticity.

Taking H_5 as an example (Figure. 3(c)), we first detect monosemantic neurons using our MS metric, as introduced in subsection. 2.2. After identifying these neurons (colored red), we apply our Reverse Deactivation method, as described in subsection. 2.3, to each of them.

For adaptability, our module (i) does not require a specific input format and (ii) outputs H_5' in the same shape as the input H_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 detailed algorithm is provided in the Appendix. E. 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 emergent learning. Researchers can focus on separable directions: (1) improving the metric to better detect factors related to emergence, and (2) designing factor-oriented solutions for better performance.

4. Experiments

4.1. Setup

To validate our hypothesis on monosemanticity, we conduct experiments on tasks including language, image, and physics simulation to test our MEmeL method:

- Language Task. Our study is based on the BERT (Pre-training of Deep Bidirectional Transformers) model (Devlin et al., 2019) and the GLUE (General Language Understanding Evaluation) benchmark (Wang et al., 2019). 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. We apply EMemL to the BERT-base model on the GLUE dataset. 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 (Liu et al., 2021) and ImageNet dataset (Deng et al., 2009). The Swin-Transformer is a transformer model that uses a hierarchical structure and shift-based windows to capture cross-window connec-

Table 1. Results on GLUE Test datasets, which are evaluated online (https://gluebenchmark.com/leaderboard). 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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	**88.9**	66.4	79.6
$\text{BERT}_{\text{MEMEL}}$	**84.9/83.8**	**71.6**	**91.3**	**93.7**	**53.8**	**85.9**	**88.9**	**67.8**	**80.2**

Table 2. Results on ImageNet-1k dataset, where 3 sizes of Swin-Transformer pretrained on ImageNet-22k are used as backbones. The rows labeled "Original" and "MEmeL" refer to the results of models without and with MEemL, respectively. 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	**85.2**
MEMEL	**81.0**	**83.4**	**85.2**

Table 3. Results on HKO-7 dataset. We initially train a ConvGRU model for 20k steps to create the base model. Then, we perform finetuning for 8k steps with and without MEmeL, labeled as "MEmeL" and "Original" respectively. The metrics used are B-MSE and B-MAE, where a smaller value indicates better performance. The best results are in **bold** fonts. The model is fine-tuned three times on each setting for fairness.

MODEL	B-MAE	B-MSE
ORIGINAL	312.6±15.8	997.5±12.3
MEMEL	**311.9**±14.0	**995.8**±9.4

tions. 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.

- Physics Simulation Task. We apply our EMemL to the ConvGRU model on the HKO-7 dataset, which forecasts precipitation based on images of radar echoes. (Shi et al., 2015; 2017). Precipitation forecasting is a challenging task, for both theory-driven and data-driven approaches, as it exhibits complex chaotic dynamics (Xuebin et al., 2017; Wang et al., 2017). 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.

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.

All the codes are implemented in PyTorch (Adam et al., 2017), which are available in the supplementary materials. Our experiment is conducted on 4 V100 cards, 8 RTX2080 cards, and 4 RTX3090 cards. To avoid a cheating-style parameter search for improvement, we minimize the tunable hyperparameter during training. For example, after calculating the MS for each layer of neurons, one can introduce various hyperparameters to determine the number of neurons to be deactivated by RD. These hyperparameters include top-k, top p percent, and a hard threshold c so that neurons with $MS > c$ are processed, etc. We choose the lightest modification, deactivating the neuron with the top-1 MS in each batch, to prove the effectiveness of our methods. More details are given in Appendix. F. Our setting may somehow compromise the improvement in score, but it preserves fairness. See more discussions in Appendix. G.3.

4.2. Main Experiment Result

We present our results on the GLUE dataset in Table. 1. $\text{BERT}_{\text{MEmeL}}$ is our variant obtained by applying MEmeL to $\text{BERT}_{\text{BASE}}$. Our method improves results on 8 tasks and achieves the same results on MRPC. (The results were obtained from our first submission to GLUE. Since our goal is to validate the effectiveness of our method rather than prove its upper bound, we did not invest time in a second submission, although the score of MRPC might be further improved by simple tuning mentioned above.)

The results on ImageNet-1k are given in Table. 2. We finetune the checkpoints provided by Swin-Transformer on three different scales: Tiny (Swin-T), Small (Swin-S), Base (Swin-B). Our method achieves the same score on the Base and Large models, and improves the results on the two smaller versions. A preliminary hypothesis is that image classification might be decomposed into feature detection and estimation of the likelihood of classes based on the presence of these features, which is relatively linear. [Update: By simply increasing the number of neurons to be deactivated from top-1 to top-10, we can outperform Swin-B. Therefore, another reason is that a large model has an great

number of neurons in each layer, and deactivating only one for each sample is too mild.]

For the precipitation forecasting task, we first pretrain a ConvGRU model on HKO for 20k steps as the base model. Then, we proceed with training for an additional 8k steps, both with and without MEmeL, for comparison purposes. The results are presented in Table. 3. The metrics used are Balanced Mean Absolute Error (B-MAE) and Balanced Mean Square Error (B-MSE), specifically designed to capture heavy rainfall (Shi et al., 2017) (See Appendix. F). Our method demonstrates better performance compared to the baseline, with lower variation.

5. Related Works

5.1. Artificial Neural Networks and the Increasing Scale

Since S & Walter (1943) first modeled a simple neural network, ANN has been investigated under a scale of several layers in the last century (LeCun et al., 1989; Yann et al., 1995). 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 (Krizhevsky et al., 2012). Later on, deeper models such as Inception and VGG increased the scale to tens and hundreds of layers (Szegedy et al., 2015; Simonyan & Zisserman, 2015). The design of critical modules, such as ResNet, also plays a crucial role in ensuring the stability of model training when increasing the depth (He et al., 2016).

After the Transformer was proposed and validated as effective in various areas (Vaswani et al., 2017), both its scale and performance have seen significant growth over the years (Devlin et al., 2019; Liu et al., 2021). In recent years, architectures based on the Transformer have achieved great success at extremely large scales (Floridi & Chiriatti, 2020; Touvron et al., 2023). Investigating the underlying effective properties created by the increase in scale is a highly demanded research direction.

5.2. 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 (Räuker et al., 2022). They study individual components to understand their functionality and usage, such as neurons for identifying dogs and cars (Chris et al., 2020). As Transformer models demonstrate their superiority in various domains, there is an increasing focus on their interpretability. Geva et al. (2021a) propose that the feed-forward layers of Transformers function as key-value pairs. Geva et al. (2021b) 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 (Gurnee et al., 2023; Trenton et al., 2023; Steven et al., 2023a). While the desire to decompose and understand everything is appealing (e.g., achieving monosententicity), it may conflict with the progress of intelligence.

5.3. Debate on the Existence of Emergence

Note that Schaeffer et al. (2023) 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.

6. Conclusion

In this paper, we discuss the concept of Emergence Learning and its importance for large-scale models. We focus on the decrease of monosemanticity as an important factor for Emergence Learning and propose methods to proactively inhibit monosemantic neurons. Our experiments validate this idea. Additionally, we provide further discussion on the monosemanticity, where researchers prefer monosemantic neurons as they have strong one-to-one correlations and are easy for AI explanation (Gurnee et al., 2023; Dar et al., 2023; Geva et al., 2021a). However, amazing intelligence could be a result of chaos, where monosemantic should be reduced instead. This hypothesis also implies bad news in the safety of AI, where more powerful AI tools (e.g., deep learning models) naturally exhibit bad explainability. In other words, serving as an Uncertainty Principle in the AI area, one may hypothesize that an AI model cannot output both the answer and reason of a complex problem, with perfect accuracy.

References

A, S. M., D, H. W., K, B. S., Serena, B., E, H. A., M, A. S., D, S. C., J, F. A., R, H. P., and C, S. C. Neuropil distribution in the cerebral cortex differs between humans and chimpanzees. *Journal of comparative neurology*, 520 (13):2917–2929, 2012.

Adam, P., Sam, G., Soumith, C., Gregory, C., Edward, Y., Zachary, D., Zeming, L., Alban, D., Luca, A., and Adam, L. Automatic differentiation in pytorch. 2017.

Bau, D., Zhu, J., Strobelt, H., Lapedriza, À., Zhou, B., and Torralba, A. Understanding the role of individual units in a deep neural network. *Proc. Natl. Acad. Sci. USA*, 117 (48):30071–30078, 2020.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

Chris, O., Nick, C., Ludwig, S., Gabriel, G., Michael, P., and Shan, C. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

CW, V. R. M., Qiang, B. G., and G, T. G. Stable hebbian learning from spike timing-dependent plasticity. *Journal of neuroscience*, 20(23):8812–8821, 2000.

Dar, G., Geva, M., Gupta, A., and Berant, J. Analyzing transformers in embedding space. In *ACL (1)*, pp. 16124–16170. Association for Computational Linguistics, 2023.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. IEEE Computer Society, 2009.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Elhage, N., Hume, T., Olsson, C., Nanda, N., Henighan, T., Johnston, S., Showk, S. E., Joseph, N., DasSarma, N., Mann, B., Hernandez, D., Askell, A., Ndousse, K., Jones, A., Drain, D., Chen, A., Bai, Y., Ganguli, D., Lovitt, L., Hatfield-Dodds, Z., Kernion, J., Conerly, T., Kravec, S., Fort, S., Kadavath, S., Jacobson, J., Tran-Johnson, E., Kaplan, J., Clark, J., Brown, T., McCandlish, S., Amodei, D., and Olah, C. Softmax linear units. *Transformer Circuits Thread*, 2022.

Floridi, L. and Chiriatti, M. GPT-3: its nature, scope, limits, and consequences. *Minds Mach.*, 30(4):681–694, 2020.

G, T. G. The self-tuning neuron: synaptic scaling of excitatory synapses. *Cell*, 135(3):422–435, 2008.

Gabriel, G., Nick, C., Chelsea, V., Shan, C., Michael, P., Ludwig, S., Alec, R., and Chris, O. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *EMNLP (1)*, pp. 5484–5495. Association for Computational Linguistics, 2021a.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *EMNLP (1)*, pp. 5484–5495. Association for Computational Linguistics, 2021b.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *CoRR*, abs/2305.01610, 2023.

Hasson, U., Nastase, S. A., and Goldstein, A. Direct fit to nature: an evolutionary perspective on biological and artificial neural networks. *Neuron*, 105(3):416–434, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

Krogh, A. What are artificial neural networks? *Nature biotechnology*, 26(2):195–197, 2008.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, 1989.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pp. 9992–10002. IEEE, 2021.

M, G. D. and S, D. A. Neuron-based heredity and human evolution. *Frontiers in neuroscience*, 9:209, 2015.

Räuker, T., Ho, A., Casper, S., and Hadfield-Menell, D. Toward transparent AI: A survey on interpreting the inner structures of deep neural networks. *CoRR*, abs/2207.13243, 2022.

S, M. W. and Walter, P. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

Samuel, S., Jascha, A., Thomas, M., Louis, K., Rojin, Z., S, H., and Jason, E. Brain-inspired learning in artificial neural networks: a review. *arXiv preprint arXiv:2305.11252*, 2023.

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? *CoRR*, abs/2304.15004, 2023.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*, pp. 802–810, 2015.

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D., Wong, W., and Woo, W. Deep learning for precipitation nowcasting: A benchmark and A new model. In *NIPS*, pp. 5617–5627, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Steven, B., Nick, C., Dan, M., Henk, T., Leo, G., Gabriel, G., Ilya, S., Jan, L., Jeff, W., and William, S. 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)*, 2023a.

Steven, B., Nick, C., Dan, M., Henk, T., Leo, G., Gabriel, G., Ilya, S., Jan, L., Jeff, W., and William, S. 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)*, 2023b.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, pp. 1–9. IEEE Computer Society, 2015.

Thomas, B., S, K. P., and A, B. T. Transcellular nanoalignment of synaptic function. *Neuron*, 96(3):680–696, 2017.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

Trenton, B., Adly, T., Joshua, B., Brian, C., Adam, J., Tom, C., Nick, T., Cem, A., Carson, D., Amanda, A., et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

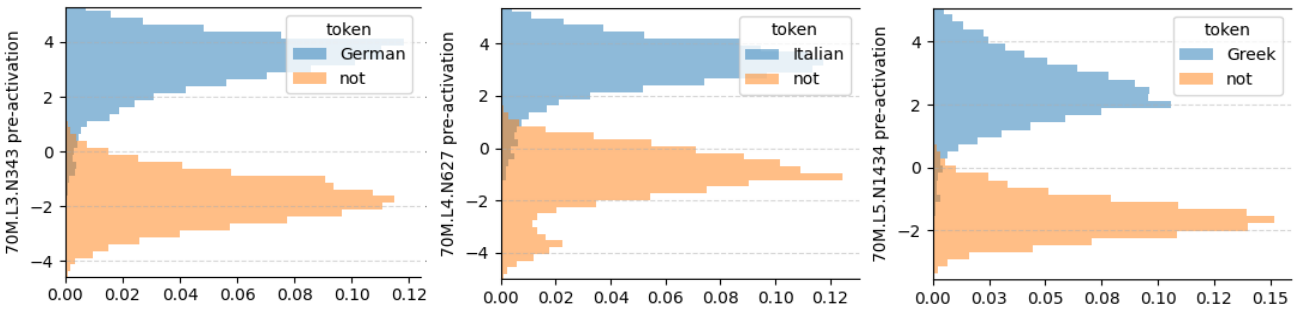
Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR (Poster)*. OpenReview.net, 2019.

Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P. S. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *NIPS*, pp. 879–888, 2017.

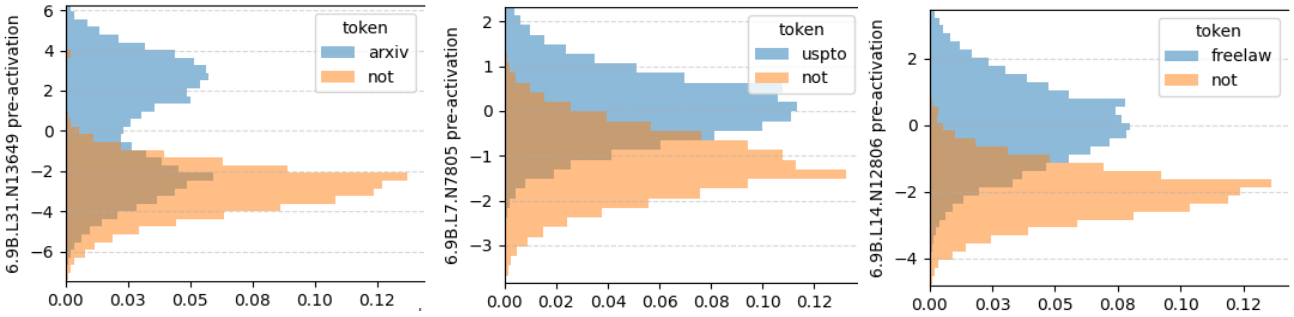
Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

Xuebin, Z., W, Z. F., Guilong, L., Hui, W., and J, C. A. Complexity in estimating past and future extreme short-duration rainfall. *Nature Geoscience*, 10(4):255–259, 2017.

Yann, L., Yoshua, B., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.



(a) Examples of value distribution of monosemantic language context neurons in Pythia-70M.



(a) Examples of value distribution of monosemantic language context neurons in Pythia-6.9B.

Figure 4. Examples of features and their monosemantic neurons from Gurnee et al. (2023). We have taken a screenshot of the distribution of activation values when features present (blue) and not present (orange) in the input context. (a) 70M model: Neurons that activate on tokens in German (left), Italian (middle), and Greek (right), as shown in Figure 15 of Gurnee et al. (2023). (b) 6.9B model: Neurons that activate on tokens in arxiv (left), uspto (middle), and freelaw (right), as shown in Figure 17 of Gurnee et al. (2023).

A. Examples of Decreasing Monosemanticity in Large Models

Even for the neurons that are mostly likely to be monosemantic, larger models show weak reliance on these neurons when given corresponding inputs. In Section 5.2 of the Gurnee et al. (2023), the authors investigate neurons that serve as *context features*, which are expected to have the most monosemanticity. However, deactivating these neurons only results in a small decrease in performance for larger models. For example, ablating a single neuron for "is_french" in a 70M parameter model leads to an average loss increase of 8% per French sequence, while the increase in loss is only 0.2% in the 6.9B model.

Moreover, when examining the activation values, neurons in smaller models show more variation when the corresponding feature is present or absent in the input. Figure. 4 presents figures from Gurnee et al. (2023). The gaps between peaks of value when features are in (blue) and not in (orange) inputs are much larger in 70M model (Figure. 4(a)) than in 6.9B model (Figure. 4(b)), indicating that the sensitivity of neuron on a single feature is much more significant in smaller scales, thus more monosemantic.

When creating interpretable key-value pairs for neurons, it is statistically uncommon to find a one-to-one correlation of monosemanticity in large models. For example, in a 16-layer transformer language model, Geva et al. (2021b) find an average of 3.6 expert identifiable features per neuron.

B. Proof of Lemma. 3.2

Proof. Recall that we define:

$$\mu_m = \frac{\sum_{i=1}^{m} h_{(i)}}{m}, \mu'_b = \frac{\sum_{i=1}^{b} h_{(m+i)}}{b}$$

We have:

$$\mu_{m+b} = \frac{\sum_{i=1}^{m+b} h_{(i)}}{m+b} \tag{9}$$

$$= \frac{\sum_{i=1}^{m} h_{(i)} + \sum_{i=1}^{b} h_{(m+i)}}{m+b} \tag{10}$$

$$= \frac{m\mu_m + b\mu_b'}{m+b} \tag{11}$$

Besides, we define:

$$\upsilon_m = \frac{\sum_{i=1}^{m}(h_{(i)} - \mu_m)^2}{m-1}, \upsilon_b' = \frac{\sum_{i=1}^{b}(h_{(m+i)} - \mu_b')^2}{b}.$$

Then we have:

$$\upsilon_{m+b} = \frac{\sum_{i=1}^{m+b}(h_{(i)} - \mu_{m+b})^2}{m+b-1} = \frac{\sum_{i=1}^{m+b}(h_{(i)} - \frac{m\mu_m + b\mu_b'}{m+b})^2}{m+b-1} \tag{12}$$

$$= \frac{\sum_{i=1}^{b}(h_{(m+i)} - \frac{m\mu_m + b\mu_b'}{m+b})^2 + \sum_{i=1}^{m}(h_{(i)} - \frac{m\mu_m + b\mu_b'}{m+b})^2}{m+b-1} \tag{13}$$

$$= \frac{\sum_{i=1}^{b}\left[h_{(m+i)} - \mu_b' - \frac{m(\mu_m - \mu_b')}{m+b}\right]^2 + \sum_{i=1}^{m}\left[h_{(i)} - \mu_m + \frac{b(\mu_m - \mu_b')}{m+b}\right]^2}{m+b-1} \tag{14}$$

$$= \frac{b\left(\frac{m(\mu_m - \mu_b')}{m+b}\right)^2 + \sum_{i=1}^{b}\left(h_{(i+m)} - \mu_b'\right)^2 - \sum_{i=1}^{b}\left[2(h_{(m+i)} - \mu_b')\frac{m(\mu_m - \mu_b')}{m+b}\right]}{m+b-1} \tag{15}$$

$$+ \frac{m\left(\frac{b(\mu_m - \mu_b')}{m+b}\right)^2 + \sum_{i=1}^{m}\left(h_{(i)} - \mu_m\right)^2 + \sum_{i=1}^{m}\left[2(h_{(i)} - \mu_m)\frac{b(\mu_m - \mu_b')}{m+b}\right]}{m+b-1} \tag{16}$$

$$= \frac{mb^2(\mu_m - \mu_b')^2 + bm^2(\mu_m - \mu_b')^2}{(m+b-1)(m+b)^2} + \frac{\sum_{i=1}^{b}\left(h_{(m+i)} - \mu_b'\right)^2 - 0 + (m-1)\upsilon_m + 0}{m+b-1} \tag{17}$$

$$= \frac{mb(\mu_m - \mu_b')^2}{(m+b-1)(m+b)} + \frac{b\upsilon_b' + (m-1)\upsilon_m}{m+b-1}, \tag{18}$$

where the third terms in Equation. (15) and Equation. (16) equal to zero as $\sum_{i=1}^{b}(h_{(m+i)} - \mu_b') = \sum_{i=1}^{b}(h_{(m+i)}) - b\mu_b' = 0, \sum_{i=1}^{m}(h_{(i)} - \mu_m) = \sum_{i=1}^{m}(h_{(i)}) - m\mu_m = 0$ and other terms are constant factors.

□

C. Proof and Discussion of Lemma. 3.3

Proof. Recall that based on Reversed Deactivation, our gradient flow originates from the following equations

$$Z' = f_2(H', X) = f_2([-h + (h + \bar{h})_{ng}, H^-, X]), \tag{19}$$

$$H = [h, H^-] = f_1(X), \tag{20}$$

where $\cdot_{ng}$ denotes "no gradient" scalar and $H^- \subset H$ are the neurons in H except h.

Though a bit of abusing the symbol, to give a clearer representation for gradient update, we use h to refer to the neuron, h_a refers to its activation value which leads to an ideal Z. $h_o = h_a$ as the value of h before processed by RD, h_u for the value updated after gradient descent. We still use $\bar{h}$ as the mean value of h and represent the difference between h and $\bar{h}$ as an updating variable $t = h - \bar{h}$. Note that except h and t, other definitions in this paragraph are untrainable scalars.

Here we focusing on $h-$related parts of Equation. (19) and (20). We denote parameters of f_1 as θ and rewrite Equation. (20) for gradient calculation and parameter update:

$$h = g(\theta) + \bar{h} \tag{21}$$

$$\Rightarrow t = g(\theta), \tag{22}$$

where g is h-related functions of f_1.

Here we fix other states and focus on the interaction between h, θ, and loss $\mathcal{L}(Z')$. We can calculate the gradient for θ:

$$\nabla\mathcal{L}(\theta) = \frac{\partial\mathcal{L}(Z')}{\partial\theta} \tag{23}$$

$$= \frac{\partial\mathcal{L}(f_2(H'))}{\partial t}\frac{\partial t}{\partial\theta} \tag{24}$$

$$= \frac{\partial\mathcal{L}(f_2)}{\partial(h'-h_a)}\frac{\partial(h'-h_a)}{\partial t}\frac{\partial t}{\partial\theta} \tag{25}$$

$$= -\nabla f(\delta)\nabla g(\theta), \tag{26}$$

where $\frac{\partial t}{\partial\theta} = \nabla g(\theta)$ as the gradient of Equation. (22) and $\frac{\partial(h'-h_a)}{\partial t} = \frac{\partial(-h+h_a+\bar{h}-h_a)}{\partial(h-\bar{h})} = -1$. $\nabla f(\delta)$ denotes $\frac{\partial\mathcal{L}(f_2)}{\partial(h'-h_a)}$. According to our assumption that $f(\delta)$ the monotonically increases with $|h'-h_a|$, $\nabla f(\delta) \geq 0$ when $h' \geq h_a$ so that $h'-h_a = |h'-h_a|$, which means $\bar{h} \geq h_a$ and $t \leq 0$. Also, we can derive that $t \geq 0$ when $\nabla f(\delta) \leq 0$.

With gradient descent of learning rate l, we will update parameter θ as:

$$\theta_u \leftarrow \theta - l\nabla\mathcal{L}(\theta) = \theta + l\nabla f(\delta)\nabla g(\theta) \tag{27}$$

By updating the parameter, we can express the updated h with a variation on the multivariate Taylor expansion:

$$h_u = g(\theta_u) \tag{28}$$

$$= g(\theta) + l\nabla g(\theta)^T\nabla f(\delta)\nabla g(\theta) + l^2\frac{1}{2}(\nabla\mathcal{L}(\theta))^T\nabla^2 g(\theta')(\nabla\mathcal{L}(\theta)), \tag{29}$$

where θ_u is the updated parameter and θ' is between θ and θ_u.

When $\nabla f(\delta) \geq 0$, we have

$$h_u - h_o = l\nabla g(\theta)^T\nabla f(\delta)\nabla g(\theta) + \frac{l^2}{2}(\nabla\mathcal{L}(\theta))^T\nabla^2 g(\theta')(\nabla\mathcal{L}(\theta)) \tag{30}$$

$$= l(d_1 + \frac{l}{2}d_2), \tag{31}$$

which is greater than 0 when $d_2 \geq 0, l > 0$ or $d_2 < 0, 0 < l < \frac{2d_1}{-d_2}$, where $d_1 = \nabla f(\delta)||\nabla g(\theta)||^2$ and $d_2 = (\nabla\mathcal{L}(\theta))^T\nabla^2 g(\theta')(\nabla\mathcal{L}(\theta))$. As $t = h - \bar{h} < 0$ under $\nabla f(\delta) \geq 0$, the activation value is smaller than the mean $\bar{h}$. Note that after the update, $h_u > h_o$. With a small enough learning rate l (i.e., $< \frac{2d_1}{-d_2}$), h_u will be updated towards $\bar{h}$ and deactivated (i.e., $\bar{h} > h_u > h_o$)

When $\nabla f(\delta) \leq 0$, we have

$$h_u - h_o = l\nabla g(\theta)^T\nabla f(\delta)\nabla g(\theta) + \frac{l^2}{2}(\nabla\mathcal{L}(\theta))^T\nabla^2 g(\theta')(\nabla\mathcal{L}(\theta)) \tag{32}$$

$$\leq l\nabla f(\delta)||g(\theta)||^2 + \frac{l^2 L}{2}||\nabla g(\theta)||^2, \tag{33}$$

$$= l(\nabla f(\delta) + l\frac{L}{2})||\nabla g(\theta)||^2 \tag{34}$$

where Equation. (34) is valid as g is the h-related part of model $\mathcal{F}$ and has a Lipschitz continuous gradient. Equation. (34) is smaller than 0 when learning rate $0 < l < (-\nabla f(\delta))\frac{2}{L}$, which ensures that updated output h_u is smaller than original activated value h_o, and thus close to $\bar{h}$ as $t = h - \bar{h} \geq 0$. Together with the case when $\nabla f(\delta) \geq 0$, we show that h is always pushed to mean $\bar{h}$ and deactivated.

□

Algorithm 1 Monosementicity Scale Computing with Needed Variables

Input: new batch of values $\boldsymbol{h} = \left\{h_{(m+1)}, h_{(m+2)}, \cdots, h_{(m+b)}\right\}$ of the neuron.
Local Variables: forget step n_f, current step c_t, current sample mean μ_m and variance υ_m
Calculate the MS for input values $\phi(h_{(m+i)}) = (h_{(m+i)} - \mu_m)^2/\upsilon_m$ for $i \in [1, b]$.
Calculate $\mu_b' = \frac{1}{b}\sum_{i=1}^{b} h_{(m+i)}$ and $\upsilon_b' = \frac{1}{b}\sum_{i=1}^{b}(h_{(m+i)} - \mu_b')^2$.
if $c_t < n_f$ **then**
  $\mu_{m+b} = \frac{m\mu_m + b\mu_b'}{m+b}$, $\upsilon_{m+b} = \frac{mb(\mu_m - \mu_b')^2}{(m+b-1)(m+b)} + \frac{b\upsilon_b' + (m-1)\upsilon_m}{m+b-1}$
else
  $\mu_{m+b} = \frac{n_f\mu_m + \mu_b'}{n_f+1}$, $\upsilon_{m+b} = \frac{n_f(\mu_m - \mu_b')^2}{(n_f+1-1/b)(n_f+1)} + \frac{\upsilon_b' + (n_f - 1/b)\upsilon_m}{n_f+1-1/b}$
end if
RETURN: Monosementicity Scale of inputs $\phi(\boldsymbol{h})$

Algorithm 2 Monosementicity-based Emergence Learning

Input: Values of n neurons with batchsize b: $H = \{\boldsymbol{h_1}, \boldsymbol{h_2}, \cdots, \boldsymbol{h_n}\}$, where $\boldsymbol{h_i} = \left\{h_{i(m+1)}, h_{i(m+2)}, \cdots, h_{i(m+b)}\right\}$.
Local Variables: late start step l_s, current step c_t, current sample mean $\{\mu_{im}\}_{i=1}^n$ and variance $\{\upsilon_{im}\}_{i=1}^n$
if $c_t < l_s$ **then**
  Calculate the MS for each input value

$$\phi(h_{i(m+j)}) = \frac{(h_{i(m+j)} - \mu_{im})^2}{\upsilon_{im} + \epsilon\sum_{k=1}^{n}\upsilon_{km}/n} \text{ for } i \in [1, n], j \in [1, b].$$

  Select values with high ϕ, adjust each of them according to MD. Replacing the original ones in H to form output H'
else
  $H' = H$
end if
Update sample mean $\{\mu_{im}\}_{i=1}^n$ and variance $\{\upsilon_{im}\}_{i=1}^n$ using Algorithm. 1.
RETURN: Adjusted layer H'

We point out that the loss function $\mathcal{L}(Z)$ does not always strictly increase with the deviation from h (i.e., $|h' - h|$) in neural networks. This is due to the presence of local optima, where increasing deviation from h may actually lead to a smaller loss. Fortunately, in the case of a monosemantic neuron, the output is strongly influenced by h and exhibits less nonlinearity, aligning more closely with the assumption.

D. 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 $O(1)$.

E. 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. (1). 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.

F. Experiment Setting

F.1. Benchmarks

For the language task, we insert 4 MEemL layers after the 4 middle Attention layers of BERT-base (12 in total). Since MEemL does not introduce any additional parameters, we can directly use their checkpoint after pretraining to finetune. The finetuning setting is the same as BERT for fairness, where each task is trained with a batch size of 32 and 3 epochs over the data for all GLUE tasks. For each task, we selected the best fine-tuning learning rate ($l \in \{5e-5, 4e-5, 3e-5, 2e-5\}$) on the validation dataset (DEV). Baseline results are from Table.1 in BERT (Devlin et al., 2019).

For the image classification task, we also insert 4 MEemL layers after the 4 middle Attention layers for each model. The training is conducted on ImageNet-1k for 30 epochs using pretrained checkpoints on ImageNet-22k. The settings are directly adopted from their scripts, except for the number of GPU cards. Baseline results are collected by running checkpoints from the github of Swin-Transformer with some updates.

For the precipitation task, we utilize radar images with a granularity of 120×120 from the HKO-7 dataset (Shi et al., 2017), which are centered in Hong Kong. These radar images cover a timespan from 2009 to 2015 and are captured every 6 minutes. The original rainfall intensity is mapped to a scale of [0, 255], forming a regression task. We will release the checkpoints and models soon [1].

F.2. Metrics

Here we list the metrics used in the experiments.

The **Accuracy**:

in a classification task, with the statistic of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), Accuracy can be derived as $\frac{\text{TP+TN}}{\text{TP+TN+FN+FN}}$.

The **F1 score**:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

where Precision is calculated as $\frac{\text{TP}}{\text{TP+FP}}$ and Recall refers to $\frac{\text{TP}}{\text{TP+FN}}$.

The **Spearman correlations**:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}},$$

where $[x_1, x_2, \cdots]$ and $[y_1, y_2, \cdots]$ are two sequences.

The **B-MSE** and **B-MAE** (The balanced mean square error and balanced mean absolute error):

$$\text{B-MSE} = \frac{1}{D} \sum_{m \in M} w(m)(m - \hat{m})^2,$$

$$\text{B-MAE} = \frac{1}{D} \sum_{m \in M} w(m)|m - \hat{m}|,$$

where $M \in \mathrm{R}^{T \times H \times W}$ is the radar intensity map to be predicted with $D = T \times H \times W$ pixels. For each pixel with intensity m, $\hat{m}$ is the prediction output and $w(m)$ is a weight to enhance the performance on heavy rainfall:

$$w(m) = \begin{cases} 1, & m < 2 \\ 2, & 2 \leqslant m < 5 \\ 5, & 5 \leqslant m < 10 \\ 10, & 10 \leqslant m < 30 \\ 30, & m \geqslant 30 \end{cases}$$

[1] Codes are published and updating on https://github.com/dominatorX/MEmeL-code

G. Discussions

G.1. Biological View of Neuron Connection

Monosemanticity and polysemanticity are inherent mappings between neurons. Similar to their functionality as connections, in neuroscience, "synaptic" enables information transmission between neurons (Hasson et al., 2020; Thomas et al., 2017). In view of the evolution of the brain, the increase of neuron activity is positively correlated to the increase of the amount of synaptic (M & S, 2015). Besides, compared with chimpanzees, humans significantly have more synaptics for information transmission (A et al., 2012).

However, the connections of artificial neurons are fixed during design. In contrast, the connection relationships of biological neurons are observable, which can serve as a reference for studying the monosemanticity of AI neurons in large-scale models.

G.2. Brain-inspired ANN design

Researchers have developed various methods for incorporating biological knowledge into ANNs (G, 2008; Samuel et al., 2023). For example, to guide the design of ANNs, Spiking Neural Networks (SNNs) draw inspiration from the way neurons communicate in the brain, where information is encoded in the timing of spikes or action potentials (CW et al., 2000). This spike-based communication enables SNNs to potentially achieve greater efficiency and better biological plausibility. In optimizing ANNs, Eligibility Propagation combines traditional error backpropagation with biologically plausible learning rules (Bellec et al., 2020). This approach updates the model by calculating an eligibility trace for each synapse and measuring its contribution to the neuron.

These methods tend to help ANN by focusing on the functionality and mechanism of biological neurons, whereas EmeL emphasizes the importance of scale as an angle of improvement.

G.3. On the Selection of Benchmark and Experimental Settings

Currently, middle- to large-scale models typically follow a pretrain-finetune training procedure, such as Bert. Pretraining is usually conducted on multiple datasets to learn general abilities like feature extraction. After pretraining, the model is finetuned for a specific task, which may sacrifice its general ability in favor of better performance.

Intuitively, to achieve the best result, a model should be trained in the following way: (i) use MEmeL during pretraining to decrease monosemanticity and increase the general ability, and (ii) exclude MEmeL during finetuning on a specific task to allow for an increase in monosemanticity and achieve better results. To illustrate, pretraining is similar to a student accumulating knowledge from multiple courses, starting from elementary school to university, while finetuning is akin to preparing for the GRE exam. Monosemanticity is similar to rote memorization, where memorizing questions and answers during courses does not help improve the GRE score, but it is useful when studying for the exam.

However, validating the idea and performing pretraining is much more computationally expensive compared to finetuning, especially for large models. For instance, using the naive method to finetune LLAMA2-7B based on our A800 is estimated to cost over $450k$ and over **5800 GPU-days** (Touvron et al., 2023) with our preliminary test. Emergence occurs primarily near 10^{22} FLOPs and costs more than **10500 GPU-days** (Wei et al., 2022). Conducting corresponding validation experiments is impossible for most researchers without the assistance of top companies.

Under such an arms race on GPUs, our experiments explore our resources to validate the generality and effectiveness of our methods. For the three task domains, we perform fine-tuning on their milestone models using MEmeL, although this approach does not fully exploit the capabilities of MEmeL compared to pretraining.

G.4. Difference from Traditional Normalizations

Previous normalizations have focused on the interaction between neurons in a layer, whereas our metric is based on the behavior of each individual neuron. For example, a neuron could be activated at 0 and deactivated at 2, which would be considered outstanding for normalization, as it consistently outputs a large value. One can better understand the importance of its 0-value through monosemantic analysis.