In-context Learning with Transformer Is Really Equivalent to a Contrastive Learning Pattern

Ruifeng Ren
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
renruifeng920@ruc.edu.cn

Yong Liu*
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
liuyonggsai@ruc.edu.cn

ABSTRACT

Pre-trained large language models based on Transformers have demonstrated amazing in-context learning (ICL) abilities. Given several demonstration examples, the models can implement new tasks without any parameter updates. However, it is still an open question to understand the mechanism of ICL. In this paper, we interpret the inference process of ICL as a gradient descent process in a contrastive learning pattern. Firstly, leveraging kernel methods, we establish the relationship between gradient descent and self-attention mechanism under generally used softmax attention setting instead of linear attention setting. Then, we analyze the corresponding gradient descent process of ICL from the perspective of contrastive learning without negative samples and discuss possible improvements of this contrastive learning pattern, based on which the self-attention layer can be further modified. Finally, we design experiments to support our opinions. To the best of our knowledge, our work is the first to provide the understanding of ICL from the perspective of contrastive learning and has the potential to facilitate future model design by referring to related works on contrastive learning.

KEYWORDS

In-context learning, Contrastive learning, Gradient descent

1 INTRODUCTION

Recently, large language models (LLMs) based on the Transformer architectures [33] has shown surprising in-context learning (ICL) capabilities [5, 13, 23, 35]. By prepending several training examples before query inputs without labels, the models can make predictions for the queries and achieve excellent performance without any parameter updates. This excellent capability enables pre-trained LLMs such as GPT models to be used in general downstream tasks conveniently. Despite the good performance of the ICL capabilities, the mechanism of ICL still remains an open question.

In order to better understand the ICL capabilities, many works began to give explanations from different aspects. Xie et al. [37] propose a Bayesian inference framework to explain how ICL occurs between pretraining and test time, where the LLMs infers a shared latent concept among the demonstration examples. Zhang et al. [40] adopt a Bayesian perspective and show that ICL implicitly performs the Bayesian model averaging algorithm, which is approximated by the attention mechanism. Garg et al. [14] demonstrate through experiments that pre-trained Transformer-based models can learn new functions from in-context examples, including (sparse) linear functions, two-layer neural networks, and decision trees. Li et al. [22] define ICL as an algorithm learning problem where a transformer model implicitly builds a hypothesis function at inference-time and derive generalization bounds for ICL. Han et al. [17] suggest that LLMs can emulate kernel regression algorithms and exhibit similar behaviors during ICL. These works have provided significant insights into the interpretation of ICL capabilities from various perspectives.

In addition, there are some works trying to better understand ICL capabilities from the perspective of gradient descent. Inspired by Aiserman et al. [1], Irie et al. [20] and Dai et al. [12] figure out a dual form of gradient descent for linear Transformer attention and then explain ICL as implicit fine-tuning. However, there is a certain gap between the linear attention and the softmax attention, which is more commonly used in the LLMs. In addition, the analogy between ICL and gradient descent is only limited to the formal resemblance where the specific details of gradient descent, including the choice of loss function and training data, have not been clearly defined or provided. Von Oswald et al. [34] show the equivalence of linear self-attention mechanism and gradient descent on a linear regression task by giving weight constructions. Similarly, Akyürek et al. [2] also prove by construction that transformers can learn linear models as learning algorithms based on gradient descent and closed-form ridge regression. However, the method of linking ICL with gradient descent using weight construction does not necessarily align with real-world scenarios. In practice, pre-trained model weights may not always meet the requirements of such construction. Thus, the question arise: *Without using weight construction and linear attention setting, can we combine ICL with gradient descent in the setting of softmax attention?*

Following these works linking ICL to gradient descent, we try to further give a novel interpretation of ICL. Firstly, we give the gradient descent of the dual form of softmax attention instead of linear attention, which is more in line with the application scenario of the actual LLMs. We then try to find the loss function that this gradient descent procedure aims to optimize, which does not require the weighting constructions as previous works. Finally, the results show that the loss function and the process of gradient descent have a structure similar to that of contrastive learning, especially without negative samples. Since there are lots of mature works in the field of contrastive learning, this can inspire us to improve the model structure in the future to make LLMs achieve better ICL capabilities. In this paper, we attempt to modify the structure of the self-attention layer from the perspective of applying regularization to the loss function, enhancing data augmentation, and introducing

*Corresponding author

negative samples. And we also design experiments to validate the effects of these modifications.

The rest of the paper is organized as follows. In Section 2, we describe the necessary notations and basic preliminaries. In Section 3, with the help of kernel methods, we establish the relationship between gradient descent and self-attention mechanism under generally used softmax attention setting instead of linear setting, and further we analyze the corresponding gradient descent process from the perspective of contrastive learning without negative samples. In Section 4, we discuss possible improvements of this contrastive learning pattern, based on which self-attention layer can be modified. In Section 5, we design experiments to support our opinions. Finally, we conclude this paper in Section 6.

2 PRELIMINARIES

2.1 In-context Learning with Transformers

We consider a model consisting of Transformer layers, each of which is represented by a self-attention sublayer and a feed-forward network sublayer. During ICL, the model accepts a sequence of input vectors $\boldsymbol{H} = [\boldsymbol{H}_D, \boldsymbol{h}_{N+1}] \in \mathbb{R}^{d_{in} \times (N+1)}$ which is composed of N demonstration example tokens $\boldsymbol{H}_D = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N] \in \mathbb{R}^{d_{in} \times N}$ and one query token $\boldsymbol{h}_{N+1} \in \mathbb{R}^{d_{in}}$. For classification tasks, each of token in $\boldsymbol{H}_D$ has the form $\boldsymbol{h}_i = [\boldsymbol{x}_i, \boldsymbol{y}_i]$ where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are encoded input text and corresponding labels respectively. And the query token has the form $\boldsymbol{h}_{N+1} = [\boldsymbol{x}_{N+1}, \boldsymbol{y}_{N+1}]$ where $\boldsymbol{y}_{N+1}$ is its label we need to predict and we can set $\boldsymbol{y}_{N+1} = \boldsymbol{0}$ as initialization.

The self-attention output can be formulated as

$$\widehat{\boldsymbol{H}} = Atten(\boldsymbol{H}) = \boldsymbol{W}_V \boldsymbol{H} softmax\left(\frac{(\boldsymbol{W}_K \boldsymbol{H})^T \boldsymbol{W}_Q \boldsymbol{H}}{\sqrt{d_{out}}}\right), \tag{1}$$

where $\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V \in \mathbb{R}^{d_{out} \times d_{in}}$ are projection matrix. We obtain the final vector $\hat{\boldsymbol{h}}_{N+1}$ in $\widehat{\boldsymbol{H}}$ to read out the prediction label for the query input. Specifically, $\hat{\boldsymbol{h}}_{N+1}$ can be formulated as

$$\hat{\boldsymbol{h}}_{N+1} = \boldsymbol{W}_V \boldsymbol{H} softmax\left(\frac{(\boldsymbol{W}_K \boldsymbol{H})^T \boldsymbol{W}_Q \boldsymbol{h}_{N+1}}{\sqrt{d_{out}}}\right). \tag{2}$$

2.2 Contrastive Learning without Negative Samples

Contrastive learning is a significant approach to self-supervised learning (SSL) which aims at learning representations by minimizing the distance between the augmentations of the same data point (positive samples) while maximizing the distance from different data points (negative samples) [7, 18, 27, 28, 31]. However, traditional contrastive learning methods need a certain sufficient number of negative pairs to avoid representational collapse and ensure the quality of the representation, which brings great burden to the calculation [4, 8, 9, 36, 38]. Thus, to avoid this, some works propose architectures for contrastive learning without negative samples, which mainly use weight-sharing network known as Siamese networks [6, 10, 15, 26, 32]. The architecture takes two augmentations $\boldsymbol{x}_1, \boldsymbol{x}_2$ from the same data $\boldsymbol{x}$ as inputs. Then, $\boldsymbol{x}_1, \boldsymbol{x}_2$ will be processed by online network and target network as encoders respectively to obtain the corresponding representations, that is, $\hat{\boldsymbol{x}}_1 = f_{online}(\boldsymbol{x}_1), \hat{\boldsymbol{x}}_2 = f_{target}(\boldsymbol{x}_2)$. The two encoder networks share weights directly or using Exponential Moving Average (EMA). Then, a prediction head will take the online representation $\hat{\boldsymbol{x}}_1$ as input to obtain the prediction representation, that is, $z_1 = g(\hat{\boldsymbol{x}}_1)$. Finally, we minimize the distance between the prediction representation and target representation,

$$\mathcal{L}\left(z_1, StopGrad(\hat{\boldsymbol{x}}_2)\right),$$

where $StopGrad(\cdot)$ means $\hat{\boldsymbol{x}}_2$ is treated as a constant during backpropagation. For $\mathcal{L}(\cdot)$, we often choose the cosine similarity or the l_2-norm as a measure of distance, although they are equivalent when the vector is normalized. We will show that the gradient descent process induced by ICL using attention mechanism can be seen as a simplified form of this architecture.

2.3 Gradient Descent on Linear Layer is the Dual Form of Linear Attention

It has been found that the linear attention can be connected to the linear layer optimized by gradient descent [1, 20, 34]. According to Aiserman et al. [1] and Irie et al. [20], during test time, a linear layer produces outputs using dot attention over all training datasets. A simple linear layer can be defined as

$$f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}, \tag{3}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_{out} \times d_{in}}$ is the projection matrix. Given training inputs $[\boldsymbol{x}_i]_{i=1}^N \in \mathbb{R}^{d_{in}}$ with their labels $[\boldsymbol{y}_i]_{i=1}^N \in \mathbb{R}^{d_{out}}$ and some loss function $\mathcal{L}$ with learning rate η, gradient descent process produces the corresponding backpropagation signals $[\boldsymbol{e}_i]_{i=1}^N \in \mathbb{R}^{d_{out}}$ where $\boldsymbol{e}_i = -\eta\left(\nabla_{\hat{\boldsymbol{y}}_i}\mathcal{L}\right)$ and $\hat{\boldsymbol{y}}_i = \boldsymbol{W}\boldsymbol{x}_i$. During test time, the trained weight matrix $\widehat{\boldsymbol{W}}$ can be represented by its initialization $\boldsymbol{W}_0$ and the updated part $\Delta\boldsymbol{W}$, that is,

$$\widehat{\boldsymbol{W}} = \boldsymbol{W}_0 + \Delta\boldsymbol{W} = \boldsymbol{W}_0 + \sum_{i=1}^{N} \boldsymbol{e}_i \otimes \boldsymbol{x}_i, \tag{4}$$

where $\otimes$ denotes the outer product according to the chain rule of differentiation.

On the other hand, this process can be associated with linear attention. Let $[\boldsymbol{k}_i]_{i=1}^N, [\boldsymbol{v}_i]_{i=1}^N, \in \mathbb{R}^{d_{in}}$ denote the N key and value vectors which form the key and value matrix $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d_{in} \times N}$ respectively. For a given query input $\boldsymbol{q} \in \mathbb{R}^{d_{in}}$, linear attention is typically defined as the weighted sum of these value vectors

$$LinearAtten(\boldsymbol{V}, \boldsymbol{K}, \boldsymbol{q}) = \boldsymbol{V}\boldsymbol{K}^T\boldsymbol{q} = \sum_{i=1}^{N} \boldsymbol{v}_i \left(\boldsymbol{k}_i^T \boldsymbol{q}\right) = \left(\sum_{i=1}^{N} \boldsymbol{v}_i \otimes \boldsymbol{k}_i\right)\boldsymbol{q}.$$

Then, we can rewrite the output of a linear layer during test time as

$$\begin{aligned} f(\boldsymbol{x}_{test}) = \widehat{\boldsymbol{W}}\boldsymbol{x}_{test} &= \boldsymbol{W}_0\boldsymbol{x}_{test} + \left(\sum_{i=1}^{N} \boldsymbol{e}_i \otimes \boldsymbol{x}_i\right)\boldsymbol{x}_{test} \\ &= \boldsymbol{W}_0\boldsymbol{x}_{test} + LinearAtten(\boldsymbol{E}, \boldsymbol{X}, \boldsymbol{x}_{test}), \end{aligned} \tag{5}$$

where $\boldsymbol{E} \in \mathbb{R}^{d_{out} \times N}$ and $\boldsymbol{X} \in \mathbb{R}^{d_{in} \times N}$ are stacked by backpropagation signals $[\boldsymbol{e}_i]_{i=1}^N$ and training inputs $[\boldsymbol{x}_i]_{i=1}^N$ respectively. We can find from Eq (5) that the weight matrix records all training datapoints and the output of a linear layer during test time indicates which training datapoints are chosen to activate using linear attention, where backpropagation signals can be considered as value

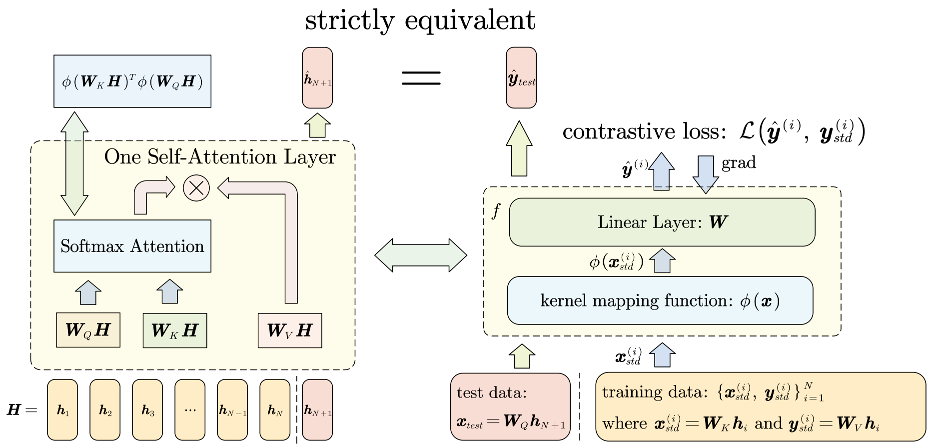


Figure 1: The inference process of ICL is equivalent to performing gradient descent on a reference model. Left part: One self-attention layer take N demonstration tokens and one query token as input. The final prediction $\hat{h}_{N+1}$ will be read out; Right part: The reference model $f(x) = W\phi(x)$ will be trained given the contrastive loss $\mathcal{L}$ and training datas $\{x^{(i)}_{std}, y^{(i)}_{std}\}_{i=1}^{N}$ transformed by the demonstration tokens. Then, the prediction $\hat{y}_{test}$ for test input x_{test} will be exactly equivalent to $\hat{h}_{N+1}$.

vectors while training inputs as keys. This interpretation uses gradient descent as a bridge to connect predictions of linear layers with linear attention, which can be seen as a simplified softmax attention used in Transformers, based on which Dai et al. [12] understand ICL as implicit fine-tuning. However, this interpretation based on linear attention deviates from the softmax attention used in practical Transformers. Furthermore, this alignment is "formly looks like" as the specific details of the gradient descent process, including the form of loss function and dataset, have not been explicitly addressed. In addition, Von Oswald et al. [34] also use weight construction methods to link the ICL ability of Transformers with gradient descent under linear regression tasks, in which case the weights W_K, W_Q and W_V of the self-attention layer need to roughly adhere to a specific constructed form. While in practice, these weights in pre-trained Transformers are fixed and may not conform to the intended construction, rendering this explanation ineffective. Thus, we will address these issues in Section 3.

3 ICL IN A CONTRASTIVE LEARNING PATTERN

In this section, we will address two questions discussed above: (i) *Without using weight construction methods, how to relate ICL to gradient descent in the setting of softmax attention instead of linear attention?* (2) *What specific datas and loss function should be used for the gradient descent process corresponding to ICL?* In addressing these two questions, we will discover that the gradient descent process corresponding to ICL has a form that is very similar to contrastive learning without negative samples.

3.1 Connecting Softmax Attention with Kernels

Before we begin establishing the connection between the ICL reasoning process and contrastive learning, we need to firstly rethinking softmax attention with kernel methods. Dai et al. [12] connect ICL with gradient descent under the linear attention setting. In fact, it is completely feasible to interpret ICL under more general softmax attention with the help of kernel methods. We define

$$A = softmax\left(\frac{(W_K H)^T W_Q H}{\sqrt{d_{out}}}\right),$$

as the attention block in Eq (1), which can be viewed as the product of the unnormalized part and normalizing multiplier, that is,

$$\begin{aligned} A &= D^{-1} A_u,\ D = diag(A_u \mathbf{1}_N), \\ A_u &= exp\left((W_K H)^T W_Q H / \sqrt{d_{out}}\right), \end{aligned} \tag{6}$$

where $exp(\cdot)$ is element-wise. Similar in [11], we define softmax kernel $K_{sm} : \mathbb{R}^{d_{out}} \times \mathbb{R}^{d_{out}} \to \mathbb{R}_+$ as:

$$K_{sm}(x, y) = exp(x^T y) = exp(\frac{\|x\|^2}{2}) K_{guass}(x, y) exp(\frac{\|y\|^2}{2}),$$

where $K_{guass} = exp(-\|x - y\|^2/2)$ is the guassian kernel when the variance $\sigma^2 = 1$. According to Mercer's theorem [25], there exists some mapping function $\phi : \mathbb{R}^{d_{out}} \to \mathbb{R}^{d_r}$ satisfying that,

$$K_{sm}(x, y) = \phi(x)^T \phi(y).$$

Thus, noting that when omitting the $\sqrt{d_{out}}$-renormalization and equivalently normalize key and value vectors in Eq (6), every entry in the unnormalized part A_u can be seen as the output of softmax kernel K_{sm} defined for the mapping ϕ, which can be formulated as:

$$\begin{aligned} A_u(i, j) &= exp\left((W_K h_i)^T W_Q h_j\right) \\ &= K_{sm}(W_K h_i, W_Q h_j) \\ &= \phi(W_K h_i)^T \phi(W_Q h_j). \end{aligned} \tag{7}$$

There have been many forms of mapping function $\phi(\cdot)$ used in efficient Transformers research to approximate this non-negative kernel [11, 21, 24, 29]. For example, we can choose $\phi(\cdot)$ as positive random features which has the form $\phi(x) = exp(w^T x - \|x\|^2/2)$

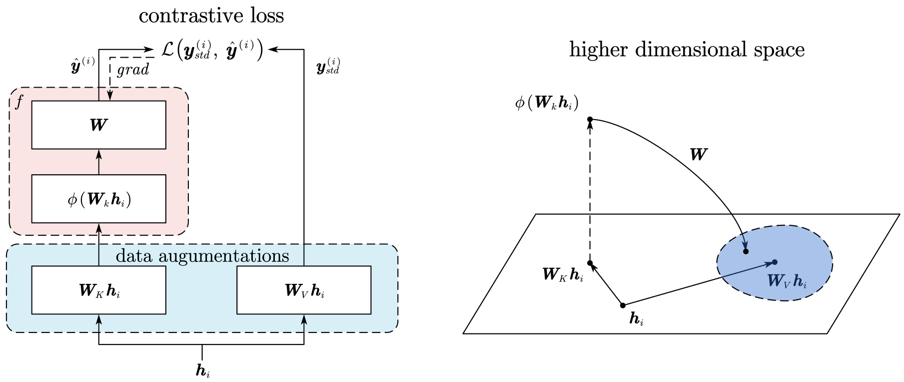


Figure 2: Illustration of the gradient descent process of ICL in a contrastive learning pattern. Left part: We can interpret the corresponding gradient descent process of ICL from the perspective of contrastive learning. Right part: For a encoded demonstration representation $\boldsymbol{h}_i$, the key and value projections act as two data augmentations $W_K\boldsymbol{h}_i$ and $W_V\boldsymbol{h}_i$. These two types of augmentation aim to create a certain distance between data representations in space. Then, the function $\phi(x)$ projects $W_K\boldsymbol{h}_i$ into a higher-dimensional space to capture deeper-level features. Finally, the weight matrix W, which maps $\phi(W_K\boldsymbol{h}_i)$ back to the original space, will be trained to make the mapped vector as close as possible to $W_V\boldsymbol{h}_i$.

to achieve unbiased approximation for attention matrix [11]. Alternatively, $\phi(\boldsymbol{x}) = elu(\boldsymbol{x}) + 1$ can also be used to approximate one order Taylor expansion of the exponential function [21].

3.2 The Gradient Descent Process of ICL

Now, given a specific softmax kernel mapping function $\phi(\boldsymbol{x})$ that satisfies Eq (7), we can define a reference model

$$f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x}), \tag{8}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_{out}\times d_r}$ is the parameters to learn. We assume that this reference model obtain its updated weights $\widehat{\boldsymbol{W}}$ after undergoing one step of gradient descent with some loss function $\mathcal{L}$. Subsequently, when we use $\boldsymbol{W}_Q\boldsymbol{h}_{N+1}$ as the test input, we can obtain its corresponding test prediction as

$$\hat{\boldsymbol{y}}_{test} = f\left(\boldsymbol{W}_Q\boldsymbol{h}_{N+1}\right) = \widehat{\boldsymbol{W}}\phi\left(\boldsymbol{W}_Q\boldsymbol{h}_{N+1}\right).$$

We will show that the $\hat{\boldsymbol{h}}_{N+1}$ in Eq (2), is strictly equivalent to the above test prediction $\hat{\boldsymbol{y}}_{test}$, which implies that the inference process of ICL involves a gradient descent step on the reference model. This can be illustrated by the following theorem:

Theorem 1. *The last token $\hat{\boldsymbol{h}}_{N+1}$ obtained through ICL inference process is strictly equivalent to the test prediction $\hat{\boldsymbol{y}}_{test}$ obtained by performing one step of gradient descent on the weight $\boldsymbol{W}$ in the reference model $f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$. The form of the loss function $\mathcal{L}$ is:*

$$\mathcal{L} = -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\boldsymbol{W}\phi(\boldsymbol{W}_K\boldsymbol{h}_i), \tag{9}$$

where η is the learning rate and D is a constant.

Proof. The derivative of $\mathcal{L}$ with respect to $\boldsymbol{W}$ should be:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{W}} = -\left[\sum_{i=1}^{N}\frac{1}{\eta D}\boldsymbol{W}_V\boldsymbol{h}_i\otimes\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right].$$

Thus, after one step of gradient descent , the learned $\widehat{\boldsymbol{W}}$ will be

$$\widehat{\boldsymbol{W}} = \boldsymbol{W}_0 - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{W}} = \boldsymbol{W}_0 + \left[\sum_{i=1}^{N}\frac{1}{D}\boldsymbol{W}_V\boldsymbol{h}_i\otimes\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right], \tag{10}$$

where $\boldsymbol{W}_0$ is its initialization and η is the learning rate. So the test prediction will be

$$\hat{\boldsymbol{y}}_{test} = \boldsymbol{W}_0\phi\left(\boldsymbol{W}_Q\boldsymbol{h}_{N+1}\right) + \left[\sum_{i=1}^{N}\frac{1}{D}\boldsymbol{W}_V\boldsymbol{h}_i\otimes\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right]\phi\left(\boldsymbol{W}_Q\boldsymbol{h}_{N+1}\right). \tag{11}$$

On the other hand, from the perspective of ICL process with one self-attention layer, with Eq (7) in our mind, we can rewrite Eq (2) as

$$\hat{\boldsymbol{h}}_{N+1} = \frac{1}{D'}\left[\boldsymbol{V}_D, \boldsymbol{v}\right]\left[\phi(\boldsymbol{K}_D), \phi(\boldsymbol{k})\right]^T\phi(\boldsymbol{q}),$$

where $[\boldsymbol{V}_D, \boldsymbol{v}] = \boldsymbol{W}_V[\boldsymbol{H}_D, \boldsymbol{h}_{N+1}]$, $[\boldsymbol{K}_D, \boldsymbol{k}] = \boldsymbol{W}_K[\boldsymbol{H}_D, \boldsymbol{h}_{N+1}]$, $\boldsymbol{q} = \boldsymbol{W}_Q\boldsymbol{h}_{N+1}$ for simplify and $D' = \boldsymbol{1}_N\phi(\boldsymbol{K}_D)^T\phi(\boldsymbol{q}) + \phi(\boldsymbol{k})^T\phi(\boldsymbol{q})$ is a constant to normalize the equivalent attention block. Further, we expand the above equation to connect the inference process of ICL using softmax attention with the gradient descent as follows

$$\begin{aligned}\hat{\boldsymbol{h}}_{N+1} &= \frac{1}{D'}\boldsymbol{v}\phi(\boldsymbol{k})^T\phi(\boldsymbol{q}) + \frac{1}{D'}\boldsymbol{V}_D\phi(\boldsymbol{K}_D)^T\phi(\boldsymbol{q}) \\ &= \boldsymbol{W}_0'\phi(\boldsymbol{q}) + \frac{1}{D'}\left[\sum_{i=1}^{N}\boldsymbol{V}_D^{(i)}\otimes\phi(\boldsymbol{K}_D^{(i)})\right]\phi(\boldsymbol{q})\end{aligned}$$

where $\boldsymbol{W}_0' = D'^{-1}\boldsymbol{v}\phi(\boldsymbol{k})^T$ and $\boldsymbol{V}_D^{(i)}, \boldsymbol{K}_D^{(i)}$ are the i-th column vetors respectively. Then, in Eq (11), when setting the initialization $\boldsymbol{W}_0 = \boldsymbol{W}_0'$ and the constant $D = D'$, we will find that

$$\hat{\boldsymbol{y}}_{test} = \boldsymbol{W}_0\phi(\boldsymbol{q}) + \frac{1}{D}\left[\sum_{i=1}^{N}\boldsymbol{V}_D^{(i)}\otimes\phi(\boldsymbol{K}_D^{(i)})\right]\phi(\boldsymbol{q}) = \hat{\boldsymbol{h}}_{N+1}, \tag{12}$$

which means $\hat{\boldsymbol{y}}_{test}$ is strictly equivalent to $\hat{\boldsymbol{h}}_{N+1}$. Thus, we have completed our proof. □

More discussion about Theorem 1 can be seen in the Appendix A.1. Theorem 1 demonstrates the equivalence between the ICL inference process and gradient descent. However, in terms of this gradient descent process, *what is the form of the training set?* In fact, once a self-attention layer has already been pre-trained, that is, $\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V$ has been determined, it seems that Transformer is using the demonstration tokens $[\boldsymbol{h}_i]_{i=1}^N$ to construct a training dataset for the reference model. Specifically, the training data points have the form $\{\boldsymbol{x}_{std}^{(i)}, \boldsymbol{y}_{std}^{(i)}\}_{i=1}^N$ where $\boldsymbol{x}_{std}^{(i)} = \boldsymbol{W}_K\boldsymbol{h}_i$ as inputs and $\boldsymbol{y}_{std}^{(i)} = \boldsymbol{W}_V\boldsymbol{h}_i$ as their labels. Then, for each data points $\boldsymbol{x}_{std}^{(i)}$, the model will output its corresponding prediction,

$$\hat{\boldsymbol{y}}^{(i)} = f\left(\boldsymbol{x}_{std}^{(i)}\right) = \boldsymbol{W}\phi\left(\boldsymbol{x}_{std}^{(i)}\right) = \boldsymbol{W}\phi\left(\boldsymbol{W}_K\boldsymbol{h}_i\right).$$

Simultaneously, the loss function can be rewritten as

$$\mathcal{L} = -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\boldsymbol{W}\phi(\boldsymbol{W}_K\boldsymbol{h}_i) = -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{y}_{std}^{(i)})^T\hat{\boldsymbol{y}}^{(i)},$$

which is related to the cosine similarity. Then, Eq (10) can be seen as one step of Stochastic Gradient Descent (SGD) on this training set to obtain the updated $\widehat{\boldsymbol{W}}$. Finally, during the testing time, we will take $\boldsymbol{x}_{test} = \boldsymbol{W}_Q\boldsymbol{h}_{N+1}$ as a test input to get the test prediction,

$$\hat{\boldsymbol{y}}_{test} = f(\boldsymbol{x}_{test}) = \widehat{\boldsymbol{W}}\phi\left(\boldsymbol{x}_{test}\right) = \widehat{\boldsymbol{W}}\phi\left(\boldsymbol{W}_Q\boldsymbol{h}_{N+1}\right) = \hat{\boldsymbol{h}}_{N+1}.$$

This process can be illustrated in Figure 1.

From the perspective of gradient descent, it seems that demonstration examples provide information about the training data points $\{\boldsymbol{x}_{std}^{(i)}, \boldsymbol{y}_{std}^{(i)}\}$ and after training process, the weight matrix $\widehat{\boldsymbol{W}}$ is optimized to learn sufficient knowledge about demonstrations. Then, during test time, we take $\boldsymbol{W}_Q\boldsymbol{h}_{N+1}$ as the input of $f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$ to produce the final representation $\hat{\boldsymbol{h}}_{N+1}$, which is consistent with the result produced by the softmax attention. In this process, we should notice that $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$ are predetermined parameters optimized after pre-training, not by construction as in Von Oswald et al. [34]. Given a reference model $f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$, by comparing Eq (12) and Eq (5), we can easily observe that the gradient descent on the loss function $\mathcal{L}$ applied to $f(\boldsymbol{x})$ is the dual form of the inference process of ICL, where $\boldsymbol{V}_D^{(i)}$, $\phi(\boldsymbol{K}_D^{(i)})$ and $\phi(\boldsymbol{q})$ play the roles of backpropagation signals, training inputs and test inputs respectively.

Recalling the form of Eq (5), we can interpret the $\boldsymbol{W}_0$ as the initialization of the weight matrix which provide the information under the zero-shot case while the second part in Eq (12) shows that the demonstration examples in ICL acts as the training samples in gradient descent. Theorem 1 also illustrates that more demonstration examples during ICL is equivalent to have sufficient training datasets in gradient descent, which contributes to giving better performance in the inference process of ICL, or equivalently the test output after training by gradient descent. Compared to Dai et al. [12] building the connection under the linear attention setting, the theorem gives more general explanation considering softmax attention with the help of kernel methods. We also note that Von Oswald et al. [34] obtain similar results by specific weight construction on a linear regression task, where $\phi(\boldsymbol{x})$ is explained as a neural network to learn the mapping representation. However, we give the derivation not by construction and our interpretation will still be applicable regardless of the form that $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$ take after pre-training.

3.3 Rethinking ICL with Contrastive Learning

Even though we have now clarified the details of the gradient descent process of ICL, as we can see from the loss function, this gradient descent process is similar to contrastive learning without negative samples: The $\boldsymbol{x}_{std}^{(i)}$ and $\boldsymbol{y}_{std}^{(i)}$ of training data, are actually transformed from the same representation $\boldsymbol{h}_i$, and moreover, the loss function $\mathcal{L}$, which has the form of cosine similarity, is often used to minimize the distance between positive pairs in contrastive learning. Thus, from the perspective of contrastive learning, we can interpret the above gradient descent process as follows: For a encoded demonstration representation $\boldsymbol{h}_i$, the key and value projections act as two data augmentations $\boldsymbol{W}_K\boldsymbol{h}_i$ and $\boldsymbol{W}_V\boldsymbol{h}_i$. These two types of augmentation aim to create a certain distance between data representations in space. And then, $\phi(\boldsymbol{x})$ projects $\boldsymbol{W}_K\boldsymbol{h}_i$ into a higher-dimensional space to capture deeper-level features. Finally, we need to train the weight matrix $\boldsymbol{W}$, which maps $\phi(\boldsymbol{W}_K\boldsymbol{h}_i)$ back to the original space, aiming to make the mapped vector as close as possible to $\boldsymbol{W}_V\boldsymbol{h}_i$. This process is illustrated in Figure 2. Thus, we give our theorem to connect ICL with contrastive learning:

Theorem 2. *The inference process of ICL can be seen as performing gradient descent on a reference model $f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$ using a contrastive learning pattern: Given $[\boldsymbol{h}_i]_{i=1}^N$ as encoded demonstration examples representations, we can obtain two augmentations $\boldsymbol{W}_K\boldsymbol{h}_i, \boldsymbol{W}_V\boldsymbol{h}_i$ by applying linear projections. One of the augmentations $\boldsymbol{y}_{std}^{(i)} = \boldsymbol{W}_V\boldsymbol{h}_i$ is directly treated as one of the positive sample while the other augmentation $\boldsymbol{W}_K\boldsymbol{h}_i$ will be taken into the reference model to learn the other positive sample's representation $\hat{\boldsymbol{y}}^{(i)} = f(\boldsymbol{W}_K\boldsymbol{h}_i)$. The gradient descent of ICL aims at optimizing the weight matrix $\boldsymbol{W}$ to narrow the cosine similarity of $\hat{\boldsymbol{y}}^{(i)}$ and $\boldsymbol{y}_{std}^{(i)}$, which can be seen as positive pairs in contrastive learning without negative samples.*

Since there are lots of mature works in contrastive learning, it is possible for us to draw on these works to improve the model design [8–10, 18, 36]. We will provide some simple perspectives from the loss function, data augmentations and negative samples to try to adjust the self-attention mechanism.

4 IMPROVING THE TRANSFORMER STRUCTURE FROM CONTRASTIVE LEARNING

4.1 Discussion on the Contrastive Loss

Although we have figured out the contrastive learning loss of the implicit gradient updates, it can be observed that this loss function is not "perfect", where the calculation of cosine similarity does not normalize $\boldsymbol{y}_{std}^{(i)}$ and $\hat{\boldsymbol{y}}^{(i)}$, allowing $\|\boldsymbol{W}\|_F$ to be optimized to infinity and preventing the loss from having a lower bound. To address this issue, we can introduce regularization to constrain the norm of $\boldsymbol{W}$, specifically by

$$\mathcal{L} = -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\boldsymbol{W}\phi(\boldsymbol{W}_K\boldsymbol{h}_i) + \frac{\alpha}{2\eta}\|\boldsymbol{W}\|_F^2, \quad (13)$$

where α is a hyperparameter to balance the two parts. As a result, we can see that the gradient update for $\boldsymbol{W}$ will be in an exponentially smoothed manner meaning that a portion of the initial part will be discarded at every step, that is,

$$\boldsymbol{W}^{(t)} = \boldsymbol{W}^{(t-1)} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = (1-\alpha)\boldsymbol{W}^{(t-1)} + \sum_{i=1}^{N} D^{-1} \boldsymbol{W}_V \boldsymbol{h}_i \otimes \phi(\boldsymbol{W}_K \boldsymbol{h}_i).$$

Equivalently, the inference process of ICL can be seen as the first step of the aforementioned update, and the self-attention mechanism will be correspondingly adjusted as,

$$\hat{\boldsymbol{h}}_{N+1} = (1-\alpha)\boldsymbol{W}_0 \phi(\boldsymbol{q}) + D^{-1} \left[\sum_{i=1}^{N} \boldsymbol{V}_D^{(i)} \otimes \phi(\boldsymbol{K}_D^{(i)}) \right] \phi(\boldsymbol{q}), \quad (14)$$

which means more demonstration information will be attended to. And for a self-attention layer, if all other tokens adopt the same modification, the self-attention layer will become

$$\widehat{\boldsymbol{H}} = Atten(\boldsymbol{H}) = \boldsymbol{W}_V \boldsymbol{H} softmax\left(\frac{(\boldsymbol{W}_K \boldsymbol{H})^T \boldsymbol{W}_Q \boldsymbol{H}}{\sqrt{d_{out}}} \right) - \alpha \boldsymbol{W}_V \boldsymbol{H}. \quad (15)$$

4.2 Discussion on the Data Augmentation

In addition to discussing the loss function, the contrastive learning paradigm also offers our some insights. In the corresponding contrastive learning of ICL, we can easily notice that data augmentation is performed using a simple linear mapping, which may be not sufficient for learning deeper-level features. To address this, we can employ more complicated nonlinear functions for more complex augmentations. Denoting these two augmentations as g_1 and g_2, consequently, the process of contrastive learning will be modified as follows

$$\mathcal{L} = -\frac{1}{\eta D} \sum_{i=1}^{N} [g_1(\boldsymbol{W}_V \boldsymbol{h}_i)]^T \boldsymbol{W} \phi(g_2 [\boldsymbol{W}_K \boldsymbol{h}_i]),$$

And from the perspective of ICL, correspondingly, the last token will be updated as

$$\hat{\boldsymbol{h}}_{N+1} = \boldsymbol{W}_0 \phi(\boldsymbol{q}) + D^{-1} \left[\sum_{i=1}^{N} g_1(\boldsymbol{V}_D^{(i)}) \otimes \phi(g_2(\boldsymbol{K}_D^{(i)})) \right] \phi(\boldsymbol{q}).$$

Equivalently, the self-attention layer will be adjusted as,

$$\widehat{\boldsymbol{H}} = Atten(\boldsymbol{H}) = g_1(\boldsymbol{W}_V \boldsymbol{H}) softmax\left(\frac{g_2(\boldsymbol{W}_K \boldsymbol{H})^T \boldsymbol{W}_Q \boldsymbol{H}}{\sqrt{d_{out}}} \right), \quad (16)$$

where $g_1(\cdot)$ and $g_2(\cdot)$ will be column-wise here. It is worth noting that here we have only presented the framework of using nonlinear functions as data augmentations to modify the self-attention layer and in the simplest case, we can set $g_1(x)$ and $g_2(x)$ as MLPs (Multi-Layer Perceptrons). However, in practice, it is encouraged to use data augmentation functions that are tailored to specific data structures. For example, in the case of CMT [16], the used Convolutional Neural Networks (CNNs) can be considered as a form of "strong data augmentations" suitable for image datas within our framework. We consider the exploration of various augmentation methods tailored to different types of data as an open question for future research.

4.3 Discussion on the Negative Samples

Although the gradient descent process corresponding to ICL exhibits some similarities with traditional contrastive learning approaches without negative samples, there are also significant differences: In traditional Siamese networks, the augmented representations as positive pairs are further learned through target and online network that share weights (or at least influence each other). The output of the target network is then passed through a predictor to compute the contrastive loss. In contrast, the contrastive learning pattern corresponding to ICL indeed performs more simply, which may potentially limit the ability of the reference model to learn representations fully without negative samples. To address this, similar to most contrastive learning approaches, we can introduce negative samples forcing the model to separate the distances between positive and negative samples at the same time, that is,

$$\mathcal{L} = -\frac{1}{\eta D} \sum_{i=1}^{N} (\boldsymbol{W}_V \boldsymbol{h}_i)^T \boldsymbol{W} \phi(\boldsymbol{W}_K \boldsymbol{h}_i) + \frac{\beta}{\eta D} \sum_{i=1}^{N} \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (\boldsymbol{W}_V \boldsymbol{h}_j)^T \boldsymbol{W} \phi(\boldsymbol{W}_K \boldsymbol{h}_i),$$

where $\mathcal{N}(i)$ is the set of the negative samples for h_i and β is a hyperparameter. Similarly, the self-attention layer is modified as

$$\begin{aligned} \widehat{\boldsymbol{H}} &= Atten(\boldsymbol{H}) = \tilde{\boldsymbol{V}} softmax\left(\frac{(\boldsymbol{W}_K \boldsymbol{H})^T \boldsymbol{W}_Q \boldsymbol{H}}{\sqrt{d_{out}}} \right), \\ \tilde{\boldsymbol{V}}^{(i)} &= \boldsymbol{W}_V \boldsymbol{h}_i - \frac{\beta}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \boldsymbol{W}_V \boldsymbol{h}_j. \end{aligned} \quad (17)$$

More details can be seen in Appendix A.2. Noting that here we simply use other token representations as negative samples for h_i. However, there are more ways to construct negative samples that are worth exploring (for instance, using noise vectors or tokens with low semantic similarity as negative samples). For specific data structures and application scenarios, customizing the selection or construction of negative samples may be more effective.

5 EXPERIMENTS

5.1 Experiment Setting

In this part, we design numerical experiments to validate the equivalence between the ICL inference process and gradient descent process with the contrastive loss.

Firstly, similar to Von Oswald et al. [34] and Garg et al. [14], we pre-train single self-attention layer using linear regression tasks. We generate the task by $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ where every element of $\boldsymbol{W} \in \mathbb{R}^{d_y \times d_x}$ is sampled from a normal distribution $\boldsymbol{W}_{ij} \sim \mathcal{N}(0,1)$ and $\boldsymbol{x}$ from uniform distribution $\boldsymbol{x} \sim U(-1,1)^{d_x}$. We set $d_x = 11$ and $d_y = 1$. Then, at each step, we use $N+1$ tokens $\{\boldsymbol{h}_i = [\boldsymbol{x}_i; y_i]\}_{i=1}^{N+1}$ as the input while the y part of the last token is masked to be zero, that is, $\boldsymbol{h}_{N+1} = [\boldsymbol{x}_i, 0]$. The self-attention layer is expected to predict $\hat{y}_{N+1}$ to approximate the ground truth value y_{N+1}. We use mean square error (MSE) as the loss function and stochastic gradient descent (SGD) [3] as the optimizer.

It is worth noting that, to align with the reference model using gradient descent, we do not use the traditional softmax function for

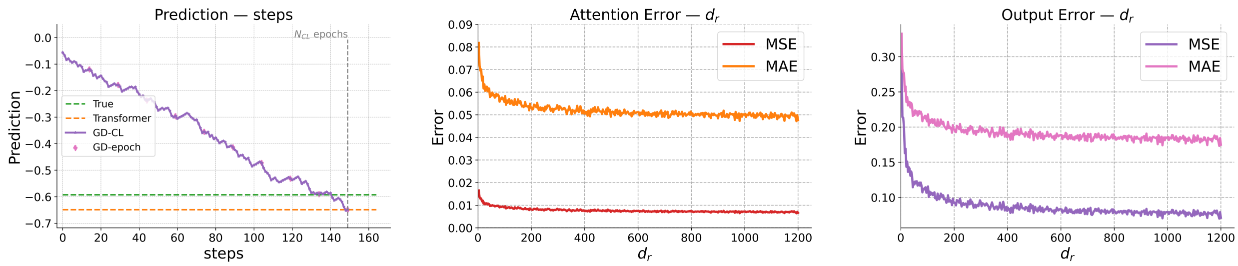
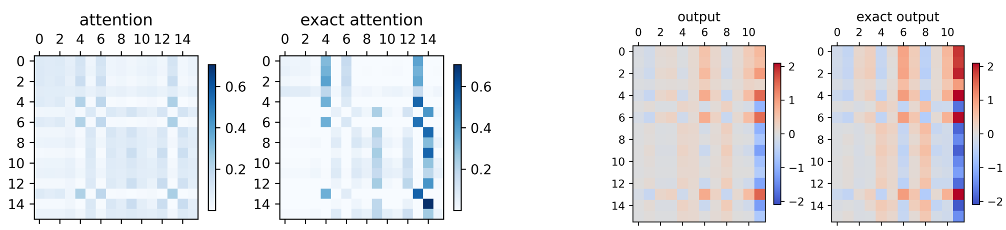


Figure 3: Analysis on the equivalence between inference process of Transformer and gradient descent under contrastive loss: Left: the changes of the prediction of the reference model as the gradient descent proceeds under setting $N = 15$ and $N_{CL} = 10$; Center: estimation error of attention matrix as d_r increases; Right: estimation error of output matrix as d_r increases.



(a) the exact attention matrix and its approximation

(b) the exact output and its approximation

Figure 4: The comparison between the exact attention matrix, output and their estimated approximations using random features under setting $N = 16$.

computing the attention matrix in the self-attention layer. Instead, we approximate the attention matrix calculation using positive random features as kernel mapping functions (Performer architecture [11]), that is $\phi(\boldsymbol{x}) = exp(\boldsymbol{w}^T\boldsymbol{x} - \|\boldsymbol{x}\|^2/2)$ where $\boldsymbol{w} \sim \mathcal{N}(0, I)$. And orthogonal random features [11, 39] or simplex random features [30] can be chosen to achieve better performance theoretically. To obtain more accurate estimation of the attention matrix, we set the output dimension of the mapping function to be 100 times the token dimension, that is, $d_r = 100(d_x + d_y) = 1200$. After pre-training, the weights of the single self-attention layer have been determined. Then, we generate test $N + 1$ tokens in the same way and y part of the $N + 1$ token is also set to be zero. We then input the test tokens into the self-attention layer to obtain the final corresponding prediction $y_{TF} = \hat{y}_{N+1}$.

On the other hand, we construct a reference model $f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$ where $\phi(\cdot)$ strictly matches the kernel mapping function used in the self-attention layer. We transform the first N tokens as the training set according to Theorem 1 and train the reference model using contrastive loss formed by Eq (9). We denote the learning rate η in the loss function Eq (9) as η_{loss} and in the gradient descent process Eq (10) as η_{gd}. We set $\eta_{gd} = \eta_{loss}/N_{CL}$ allowing the transformed N demonstration tokens to undergo N_{CL} epochs training. The transformed last token is used as the test input to obtain the test prediction y_{CL} of the reference model. We are interested in whether y_{TF} and y_{CL} are strictly equivalent after N_{CL} epochs training. More details of experiment setting can be found in the Appendix A.3.

5.2 Equivalence Between Inference Process of Transformer and Gradient Descent under Contrastive Loss

The results under setting $N = 15$ are shown in the left part of Figure 3. At each step, we choose one demonstration token as training input to update the weight $\boldsymbol{W}$ in $\boldsymbol{y} = \boldsymbol{W}\phi(\boldsymbol{x})$. After pre-training, the predictions of the single self-attention layer are very close to the ground truth values (marked as Transformer and True in the figure). Under setting $N = 15$, the label part of the last test token is $y_{N+1} = -0.5927$ and the prediction of the self-attention layer is $y_{TF} = -0.6490$. In addition, after N_{CL} epochs of training with the contrastive loss, the output of the reference model (marked as GD-CL) matches exactly with the output of the self-attention layer, which aligns with our theoretical analysis. More results can be seen in the Appendix A.4.

We investigate the impact of changing the dimension of random features d_r on the approximation of attention matrices and output, using Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics, where we conduct 50 repeated experiments and calculated the average values for each value of d_r, as shown in the center and right part of Figure 3. It can be observed that as the dimension of random features increases, the approximation performance gradually improves, with both errors reaching a low level in the end. Furthermore, we visualize the exact attention matrix and the output with the approximation results, which are shown in the Figure 4. As we can see, although some larger values are not estimated accurately due to the limited dimension of the random features we select, the majority of the information is still

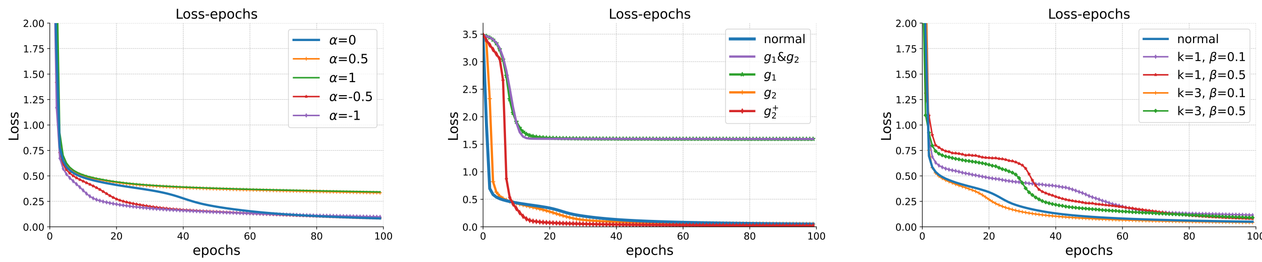


Figure 5: Analysis on different model modifications. Left: the performance for regularized-model when varying different values of α; Center: the performance for augmented-model when choosing different data augmentations; Right: the performance for negative–model when varying the number of negative samples and β.

estimated comprehensively well, which illustrates the validity of the selected kernel mapping function.

5.3 Analysis on the Regularized Contrastive Loss

In Section 4.1, we discussed the impact of applying regularization to the contrastive loss and the changes it brings to the structure of the self-attention layer (we call it regularized-model). We vary different values of α to investigate the impact on pre-training performance for the same setting, as shown in the left part of Figure 5. It can be observed that when $\alpha > 0$, under the pre-training setting, the regularized-model converges to a poorer result while when $\alpha < 0$, the model converges faster and achieves results comparable to the model without regularization ($\alpha = 0$). At least for this setting, this is contrary to our initial intention of applying regularization to the contrastive loss. This may be due to the fact that when $\alpha < 0$, some information is enhanced by the corresponding reference model which implicitly performs gradient descent. In fact, we can find from Eq (15) that when $\alpha < 0$, a residual connection multiplied by a coefficient α with respect to $\boldsymbol{W}_V\boldsymbol{H}$ is introduced to the normal self-attention layer, which may contribute to accelerating convergence.

5.4 Analysis on the Data Augmentation

Here, we further analyze the expressive ability of this modified self-attention layer inspired by using data augmentation (we called augmented-model) through experiments. During the pre-training process, we simply use $g_1(\boldsymbol{x}) = g_2(\boldsymbol{x}) = \sigma(\boldsymbol{W}\boldsymbol{x})$ as data augmentations to enhance the value and key embeddings respectively in Eq (16) and we choose GELU [19] as the activation function $\sigma(\cdot)$. We consider simultaneous and separate use of g_1 and g_2 and the experiment results are shown in the center part of Figure 5. It can be observed that when g_1 and g_2 are used simultaneously, as well as when only using g_1, the model converges to a poorer solution. However, when we only use g_2, that is, only provide data augmentation for the key vectors, the model actually shows slightly faster convergence. Furthermore, we use $g_2^+(\boldsymbol{x}) = \sigma(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x}))$ as a more complicated augmentation function, the result indicates that although the model initially converges slightly slower due to the increased number of parameters, it eventually accelerates convergence and achieves a better solution. This indicates that appropriate data augmentation indeed have the potential to enhance the capabilities of the self-attention layer. However, as we discussed in Section 4.2, data augmentations specifically designed for the specific data may be more beneficial in improving model performance. We leave the exploration of more specific augmentation techniques for future research.

5.5 Analysis on the Negative Samples

We also discuss the potential modifications that could arise from introducing negative samples in Section 4.3 (we call negative-model here). Similarly, we conduct experiments to explore the effects on the model performance by selecting the k tokens with the lowest attention scores as negative samples for each token. From Eq (18), we can see that when we select the other tokens as negative samples, it is equivalent to subtracting a certain value from the attention scores corresponding to those tokens and the sum of these subtracted attention scores is β. To ensure numerical stability, we appropriately increase the attention scores of tokens that are not treated as negative samples so that the sum of attention scores still remains 1. We vary the number of negative samples k and the β in the Eq (18) to investigate the impact on the model's performance, as shown in the right part of Figure 5. It can be found that, with appropriate settings of k and β (for example, $k = 3$ and $\beta = 0.1$), the model has the potential to achieve faster convergence. In fact, we can note that in the original attention mechanism, attention scores are non-negative, which means that the information from tokens with lower similarity can still retain some level of information and the harmful information that has already been retained cannot be removed. However, in the improved structure, attention scores can potentially become negative, which implies that after learning, the updated tokens can remove some harmful information. Certainly, as we discussed in Section 4.3, for different data structures, more refined methods of selecting and constructing negative samples may be more effective and we leave these aspects for future exploration.

6 CONCLUSION

In this paper, we present an interpretation of the inference process of ICL as a gradient descent process within a contrastive learning pattern. Firstly, we establish the relationship between gradient descent and the self-attention mechanism by leveraging kernel methods under the commonly used softmax attention setting, rather than the linear attention setting. Then, we analyze the corresponding gradient descent process of ICL from the perspective of contrastive learning without negative samples and discuss potential enhancements to this contrastive learning pattern, which can lead to further modifications of the self-attention layer. Furthermore, we conduct experiments to validate and support our findings. To the best of our knowledge, our work is the first to provide an understanding of ICL from the viewpoint of contrastive learning and has the potential to inspire future model design by drawing insights from related works on contrastive learning.

REFERENCES

[1] MA Aiserman, Emmanuil M Braverman, and Lev I Rozonoer. 1964. Theoretical foundations of the potential function method in pattern recognition. *Avtomat. i Telemeh* 25, 6 (1964), 917–936.
[2] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661* (2022).
[3] Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing* 5, 4-5 (1993), 185–196.
[4] Philip Bachman, R Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems* 32 (2019).
[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems* 33 (2020), 9912–9924.
[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.
[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.
[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297* (2020).
[10] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 15750–15758.
[11] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794* (2020).
[12] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559* (2022).
[13] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234* (2022).
[14] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems* 35 (2022), 30583–30598.
[15] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33 (2020), 21271–21284.
[16] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. 2022. Cmt: Convolutional neural networks meet vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12175–12185.
[17] Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. 2023. In-Context Learning of Large Language Models Explained as Kernel Regression. *arXiv preprint arXiv:2305.12766* (2023).
[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.
[19] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).
[20] Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. 2022. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *International Conference on Machine Learning*. PMLR, 9639–9659.
[21] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. (2020), 5156–5165.
[22] Yingcong Li, M. Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. 2023. Transformers as Algorithms: Generalization and Stability in In-context Learning. *arXiv preprint arXiv:2301.07067* (2023).
[23] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.
[24] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. 2021. Soft: Softmax-free transformer with linear complexity. *Advances in Neural Information Processing Systems* 34 (2021), 21297–21309.
[25] J. Mercer. 1909. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 209 (1909), 415–446. http://www.jstor.org/stable/91043
[26] Ishan Misra and Laurens van der Maaten. 2020. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6707–6717.
[27] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4004–4012.
[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).
[29] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. 2021. Random feature attention. *arXiv preprint arXiv:2103.02143* (2021).
[30] Isaac Reid, Krzysztof Marcin Choromanski, Valerii Likhosherstov, and Adrian Weller. 2023. Simplex random features. In *International Conference on Machine Learning*. PMLR, 28864–28888.
[31] Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems* 29 (2016).
[32] Yuandong Tian, Xinlei Chen, and Surya Ganguli. 2021. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*. PMLR, 10268–10278.
[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[34] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. (2023), 35151–35174.
[35] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).
[36] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3733–3742.
[37] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An Explanation of In-context Learning as Implicit Bayesian Inference. *arXiv preprint arXiv:2111.02080* (2021).
[38] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. 2019. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6210–6219.
[39] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. 2016. Orthogonal random features. *Advances in neural information processing systems* 29 (2016).
[40] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. 2023. What and How does In-Context Learning Learn? Bayesian Model Averaging, Parameterization, and Generalization. *arXiv preprint arXiv:2305.19420* (2023).

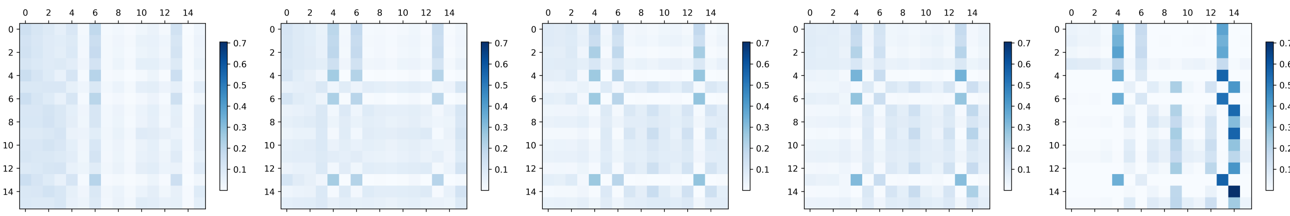
A APPENDIX

A.1 More Discussion on Theorem 1

In this part, we would like to explain how the form of the loss function is derived. In fact, we can obtain the form of the loss function illustrated in the theorem 1 through a simple backward derivation. Given a reference nonlinear model

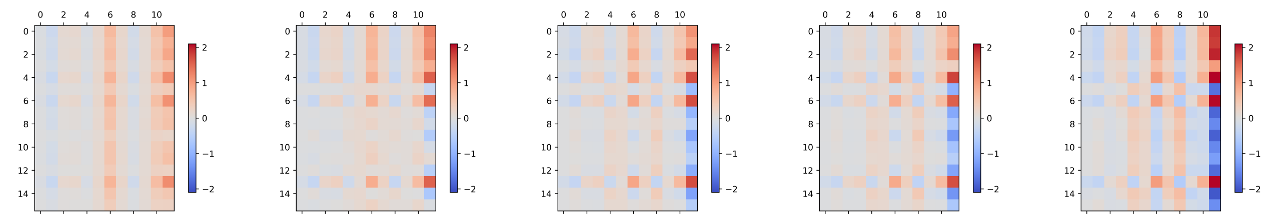
$$y = f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$$

where $\phi(\boldsymbol{x})$ is a kernel mapping function for softmax attention, satisfying unbiased approximation for softmax kernel, that is, $exp(\boldsymbol{q}^T\boldsymbol{k}) =$



(a) $d_r = 3$ (b) $d_r = 12$ (c) $d_r = 120$ (d) $d_r = 12000$ (e) exact attention

Figure 6: The changes of the prediction of the reference model as the gradient descent proceeds with the contrastive loss.



(a) $d_r = 3$ (b) $d_r = 12$ (c) $d_r = 120$ (d) $d_r = 12000$ (e) exact output

Figure 7: The changes of the prediction of the reference model as the gradient descent proceeds with the contrastive loss.

$\phi(\boldsymbol{q})^T\phi(\boldsymbol{k})$. Then, after one step of gradient descent, $\boldsymbol{W}$ will be updated as:

$$\boldsymbol{W} = \boldsymbol{W}_0 + \Delta\boldsymbol{W} = \boldsymbol{W}_0 + \sum_{i=1}^{N} \boldsymbol{e}_i \otimes \phi(\boldsymbol{x}_i),$$

where $\otimes$ denotes the outer product and $\boldsymbol{e}_i = -\eta\nabla_{\phi(\boldsymbol{x})}\mathcal{L}$ is the backpropagation signal. The prediction for a test input $\boldsymbol{x}_{test}$ will be

$$\hat{\boldsymbol{y}}_{test} = f(\boldsymbol{x}_{test}) = \boldsymbol{W}_0\boldsymbol{x}_{test} + \sum_{i=1}^{N} \boldsymbol{e}_i \otimes \phi(\boldsymbol{x}_i)\boldsymbol{x}_{test}$$

On the other hand, the forward process of ICL can be expressed as

$$\begin{aligned}
\hat{\boldsymbol{h}}_{N+1} &= \boldsymbol{W}_V\boldsymbol{H}softmax\left(\frac{(\boldsymbol{W}_K\boldsymbol{H})^T\boldsymbol{W}_Q\boldsymbol{h}_{N+1}}{\sqrt{d_{out}}}\right)\\
&= D^{-1}\boldsymbol{W}_V[\boldsymbol{H}_D, \boldsymbol{h}_{N+1}]\phi\left((\boldsymbol{W}_K[\boldsymbol{H}_D, \boldsymbol{h}_{N+1}])^T\,\phi(\boldsymbol{W}_Q\boldsymbol{h}_{N+1}\right)\\
&= D^{-1}[\boldsymbol{V}_D, \boldsymbol{v}]\,[\phi(\boldsymbol{K}_D), \phi(\boldsymbol{k})]^T\,\phi(\boldsymbol{q})\\
&= \boldsymbol{W}_0\phi(\boldsymbol{q}) + \left[\sum_{i=1}^{N} D^{-1}\boldsymbol{V}_D^{(i)} \otimes \phi(\boldsymbol{K}_D^{(i)})\right]\phi(\boldsymbol{q})
\end{aligned}$$

where $\boldsymbol{W}_0 = D^{-1}\boldsymbol{v}\phi(\boldsymbol{k})^T$ and $\boldsymbol{V}_D^{(i)}, \boldsymbol{K}_D^{(i)}$ are the i-th column vetors respectively. Comparing the form of $\hat{y}_{test}$ and $\hat{h}_{N+1}$, we can interpret the $\boldsymbol{W}_0$ as the initialization of the weight matrix which provide the information under the zero-shot case while the second part in Eq (12) shows that the demonstration examples in ICL acts as the training samples in gradient descent.

Noting that $\mathcal{L}$ is a scalar function, the differential for $\mathcal{L}$ should be

$$\begin{aligned}
dL &= tr\left(-\frac{1}{\eta D}\sum_{i=1}^{N}\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\boldsymbol{h}_i^T W_V^T dW\right)\\
&= tr\left(-\frac{1}{\eta D}\sum_{i=1}^{N}\boldsymbol{h}_i^T W_V^T dW\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right)\\
&= d\left(tr\left(-\frac{1}{\eta D}\sum_{i=1}^{N}\boldsymbol{h}_i^T W_V^T W\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right)\right)
\end{aligned}$$

Remove the differential sign, we get

$$\begin{aligned}
\mathcal{L} &= tr\left(-\frac{1}{\eta D}\sum_{i=1}^{N}\boldsymbol{h}_i^T W_V^T W\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right)\\
&= -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\,\boldsymbol{W}\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\\
&= -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\,f(\boldsymbol{W}_K\boldsymbol{h}_i)\\
&= -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{y}_{std}^{(i)})^T\hat{\boldsymbol{y}}^{(i)},
\end{aligned}$$

After calculation and simplification, we get the final loss function as

$$\mathcal{L} = -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\,f(\boldsymbol{W}_K\boldsymbol{h}_i) = -\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{y}_{std}^{(i)})^T\hat{\boldsymbol{y}}^{(i)},$$

where $\boldsymbol{y}_{std}^{(i)} = \boldsymbol{W}_V\boldsymbol{h}_i$ can be defined as the standard labels for the corresponding training inputs $\boldsymbol{W}_K\boldsymbol{h}_i$. This way, we obtain the loss function mentioned in Theorem 1.

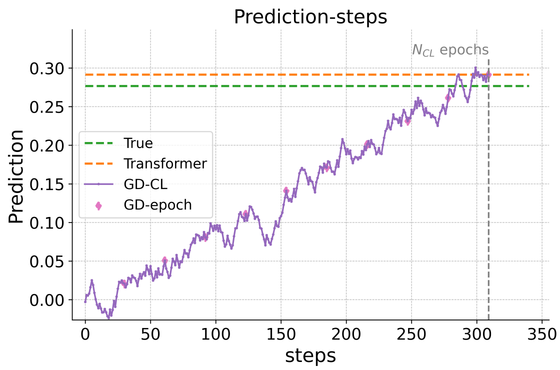


Figure 8: The changes of the prediction of the reference model as the gradient descent proceeds with the contrastive loss under setting $N = 31$ and $N_{CL} = 10$.

A.2 More Details of the Model Modification Based on Negative Samples

We can introduce negative samples forcing the model to separate the distances between positive and negative samples at the same time, that is,

$$\begin{aligned}\mathcal{L} = &-\frac{1}{\eta D}\sum_{i=1}^{N}(\boldsymbol{W}_V\boldsymbol{h}_i)^T\boldsymbol{W}\phi(\boldsymbol{W}_K\boldsymbol{h}_i)\\ &+\frac{\beta}{\eta D}\sum_{i=1}^{N}\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}(\boldsymbol{W}_V\boldsymbol{h}_j)^T\boldsymbol{W}\phi(\boldsymbol{W}_K\boldsymbol{h}_i),\end{aligned}$$

where $\mathcal{N}(i)$ is the set of the negative samples for h_i and β is a hyperparameter. Similarly, the self-attention layer is modified as

$$\begin{aligned}\widehat{\boldsymbol{H}} &= Atten(\boldsymbol{H}) = \tilde{\boldsymbol{V}}softmax\left(\frac{(\boldsymbol{W}_K\boldsymbol{H})^T\boldsymbol{W}_Q\boldsymbol{H}}{\sqrt{d_{out}}}\right),\\ \tilde{\boldsymbol{V}}^{(i)} &= \boldsymbol{W}_V\boldsymbol{h}_i - \frac{\beta}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\boldsymbol{W}_V\boldsymbol{h}_j.\end{aligned} \tag{18}$$

As a result, the gradient descent and inference process of ICL will be modified as

$$\boldsymbol{W} = \boldsymbol{W}_0 + \sum_{i=1}^{N}D^{-1}\left(\boldsymbol{W}_V\boldsymbol{h}_i - \frac{\beta}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\boldsymbol{W}_V\boldsymbol{h}_j\right)\otimes\phi(\boldsymbol{W}_K\boldsymbol{h}_i),$$

$$\begin{aligned}\hat{\boldsymbol{h}}_{N+1} &= \boldsymbol{W}_0\phi(\boldsymbol{q}) + D^{-1}\left[\sum_{i=1}^{N}\tilde{\boldsymbol{V}}_D^{(i)}\otimes\phi(\boldsymbol{K}_D^{(i)})\right]\phi(\boldsymbol{q}),\\ \tilde{\boldsymbol{V}}_D^{(i)} &= \boldsymbol{V}_D^{(i)} - \frac{\beta}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\boldsymbol{V}_D^{(j)},\end{aligned}$$

respectively. And finally, we can get modified self-attention layer as Eq (18). In corresponding experiments, for each token, we simply choose other the k least relevant tokens as its negative samples, i.e., the k tokens with the lowest attention scores.

A.3 More Details of Experiment Setting

In this part, we will discuss our experimental setup in more details.

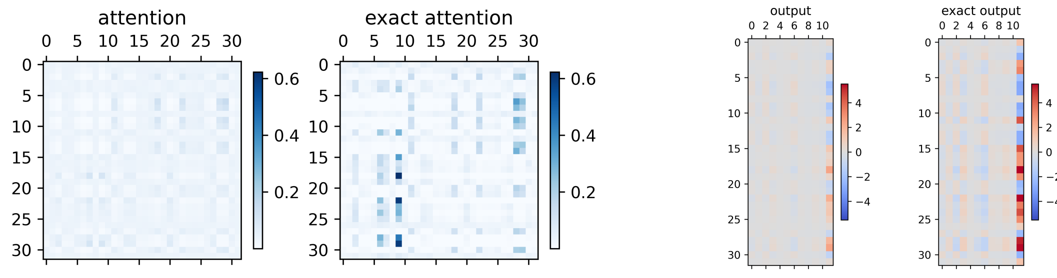
Although we can explore the equivalence between the inference process of a single-layer self-attention with any given specific weights and gradient descent under contrastive loss as the proposed equivalence does not depend on the specific construction of the weights, in order to align with real-world scenarios and facilitate the investigation of the effects of subsequent model structure modifications, we still choose to train the model to incorporate task-specific knowledge before inference. Specifically, we generate the task by $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ where every element of $\boldsymbol{W} \in \mathbb{R}^{d_y\times d_x}$ is sampled from a normal distribution $\boldsymbol{W}_{ij} \sim \mathcal{N}(0,1)$ and $\boldsymbol{x}$ is sampled from a Gaussian distribution $\boldsymbol{x} \sim U(-1,1)^{d_x}$. To facilitate more accurate estimation of attention matrices using random features, we only set a small value for $d_x = 11$ and $d_y = 1$. Then, at each step, we use $N+1$ tokens $\{\boldsymbol{h}_i = [\boldsymbol{x}_i;\boldsymbol{y}_i]\}_{i=1}^{N+1}$ as the input while the $\boldsymbol{y}$ part of the last token is masked to be zero, that is, $\boldsymbol{h}_{N+1} = [\boldsymbol{x}_i, \boldsymbol{0}]$. The single-layer self-attention layer is expected to predict $\hat{\boldsymbol{y}}_{N+1}$ to approximate the ground truth value $\boldsymbol{y}_{N+1}$. We use mean square error (MSE) as the loss function, that is, for each epoch,

$$\mathcal{L} = \frac{1}{N_{step}}\sum_{j=1}^{N_{step}}\|\hat{\boldsymbol{y}}_{N+1}^{(j)} - \boldsymbol{y}_{N+1}^{(j)}\|^2,$$

where $\hat{\boldsymbol{y}}_{N+1}^{(j)}$ and $\boldsymbol{y}_{N+1}^{(j)}$ are the prediction and ground truth value at j-th step and N_{step} is the number of steps. We set $N_{step} = 1024$ for $N+1 = 16$ while $N_{step} = 512$ for $N+1 = 32$, which means the total number of tokens remains unchanged at $16,384$. We choose stochastic gradient descent (SGD) [3] as the optimizer and we set the learning rate to 0.003 for experiments in Section 5.2 and 5.3, while the remaining experiments to 0.005. We also attempt the multi-task scenario, where the input token at each step is generated from a different task. However, we find it challenging for a single-layer self-attention to effectively learn in this setting, resulting in disordered predictions. Therefore, our experiments are currently limited to single-task settings, and the multi-task scenario is worth further investigation in the future.

It is worth noting that we approximate the attention matrix calculation using random features as kernel mapping function instead of using the traditional softmax function in the self-attention layer[11]. The mapping function $\phi : \mathbb{R}^{d_{out}} \to \mathbb{R}^{d_r}$ has the form of $\phi(\boldsymbol{x}) = exp(\boldsymbol{w}^T\boldsymbol{x} - \|\boldsymbol{x}\|^2/2)$ where $\boldsymbol{w} \sim \mathcal{N}(0, I)$. We visualize the exact attention matrix and compare it with the estimated attention matrices obtained using different values of d_r, as shown in Figure 6. Again, it can be seen that as d_r increases, the approximation of the true attention matrix improves gradually and similar results can be observed for the analysis of output matrices in Figure 7. These findings indicate that our choice of using random features as mapping functions to estimate the true softmax attention and conduct experiments is entirely appropriate.

To obtain a more accurate estimation of the attention matrix, we set the output dimension of the mapping function to be 100 times the input dimension, that is, $d_r = 100(d_x + d_y) = 1200$. After the weights $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$ of the single-layer self-attention layer have been determined, we generate test $N+1$ tokens in the same way where the y part of the $N+1$ token is also set to be zero and finally input the test tokens into the single-layer attention layer to obtain the corresponding predicted $\hat{\boldsymbol{y}}_{N+1}$.



(a) the exact attention matrix and its approximation **(b) the exact output and its approximation**

Figure 9: The comparison between the exact attention matrix, output and their estimated approximations using random features under setting $N = 32$.

On the other hand, we choose a reference model $f(\boldsymbol{x}) = \boldsymbol{W}\phi(\boldsymbol{x})$ where $\phi(\cdot)$ is strictly equivalent to the kernel mapping function used in the above self-attention layer. We transform the first N tokens as the training set according to Theorem 1 and train the reference model using the contrastive loss formed by Eq (9). In fact, according to Theorem 1, when we perform one step of gradient descent on this training set, the output results for the test inputs will strictly equal $\hat{\boldsymbol{y}}_{N+1}$. We can set different values for η in the Eq (9) and Eq (10), named as η_{loss} and η_{gd} respectively. If we want to align more closely with the actual scenario, that is, to perform multiple descent steps, we set $\eta_{gd} = \eta_{loss}/N_{CL}$ allowing the transformed N demonstration tokens to undergo N_{CL} epochs of training and we choose to use only one training data per gradient descent step (batchsize = 1), resulting in N steps for each epoch. Thus, the update process for $\boldsymbol{W}$ will be,

$$\begin{aligned}\widehat{\boldsymbol{W}} &= \boldsymbol{W}_0 - \eta_{gd}\frac{\partial\mathcal{L}}{\partial\boldsymbol{W}} = \boldsymbol{W}_0 + \frac{\eta_{gd}}{\eta_{loss}}\left[\sum_{i=1}^{N}\frac{1}{D}\boldsymbol{W}_V\boldsymbol{h}_i \otimes \phi(\boldsymbol{W}_K\boldsymbol{h}_i)\right] \\ &= \boldsymbol{W}_0 + N_{CL}\left[\sum_{i=1}^{N}\frac{1}{D}\boldsymbol{W}_V\boldsymbol{h}_i \otimes \phi(\boldsymbol{W}_K)\right].\end{aligned}$$

It can be observed that due to the unchanged update signal obtained from each epoch of $\boldsymbol{W}$, after N_{CL} training epochs, our results will strictly equal the result of one update step when $\eta_{gd} = \eta_{loss}$ and using full training datas as one batch. It should be noted that this result is based on the fact that the gradient signal remains constant during each update step. Conversely, if the gradient signal varies during the updates, there will be some deviation in the final result. For example, in the regularized-model, each gradient update undergoes exponential decay and the gradient signal is no longer constant, leading to a slight deviation between $\hat{\boldsymbol{h}}_{N+1}$ and $\hat{\boldsymbol{y}}_{test}$, which will also be illustrated in Figure 10.

A.4 More Experiment Results

The results under setting $N = 31$ are shown in Figure 8. Under setting $N = 31$, the label part of the last test token is $y_{N+1} = 0.2767$ and the prediction of the self-attention layer is $y_{TF} = 0.2915$. Furthermore, to demonstrate the effectiveness of using random features as kernel mapping functions to approximate the softmax attention mechanism, we compare the exact attention matrix and the output with the approximation results, which are shown in the Figure 9.

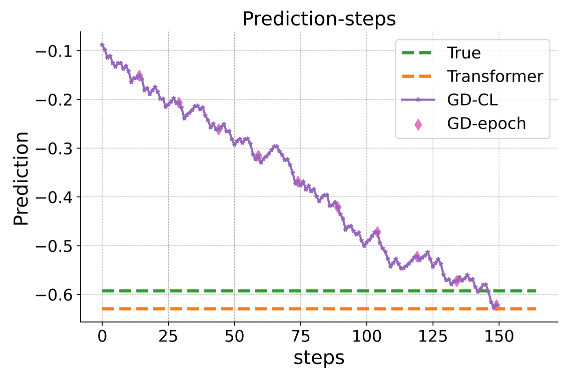


Figure 10: The changes of the prediction of the reference model as the gradient descent proceeds with the regularized contrastive loss under setting $N_{CL} = 10$ and $\alpha = 0.01$.

Furthermore, for regularized model, we explore the equivalence between the model's inference process and gradient descent with the regularized contrastive loss, as shown in Figure 10. We find that under setting $N_{CL} = 10$ and $alpha = 0.01$, as the reference model discards a portion of the original weight matrix information at each step, there is a slight deviation between the final output and the model's inference result. In fact, under the setting of $N_{CL} = 1$ and we take all demonstration tokens as one batch, the two outputs will be strictly equivalent.