LoRA Meets Dropout under a Unified Framework

Anonymous ACL submission

Abstract

With the remarkable capabilities, large language models (LLMs) have emerged as essential elements in numerous NLP applications, while parameter-efficient finetuning, especially LoRA, has gained popularity as a lightweight approach for model customization. Meanwhile, various dropout methods, initially designed for full finetuning with all the parameters updated, alleviates overfitting associated with excessive parameter redundancy. Hence, a possible contradiction arises from negligible trainable parameters of LoRA and the effectiveness of previous dropout methods, which has been largely overlooked. To fill this gap, we first confirm that parameter-efficient LoRA is also overfitting-prone. We then revisit transformer-specific dropout methods, and establish their equivalence and distinctions mathematically and empirically. Building upon this comparative analysis, we introduce a unified framework for a comprehensive investigation, which instantiates these methods based on dropping position, structural pattern and compensation measure. Through this framework, we reveal the new preferences and performance comparisons of them when involved with limited trainable parameters. This framework also allows us to amalgamate the most favorable aspects into a novel dropout method named HiddenKey. Extensive experiments verify the remarkable superiority and sufficiency of HiddenKey across multiple models and tasks, which highlights it as the preferred approach for high-performance and parameter-efficient finetuning of LLMs.

1 Introduction

Recently, transformers (Vaswani et al., 2017), such as GPT-4 (OpenAI, 2023), PaLM 2 (Anil et al., 2023) and LLaMA 2 (Touvron et al., 2023b), have been rapidly expanded to billions of parameters, leading to remarkable performance boost. When customizing these models for downstream tasks, parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019; Hu et al., 2021; Liu et al., 2022) has been widely adopted as a lightweight method, which generally freezes the majority of parameters while only updating or adding negligible trainable parameters. Among these methods, LoRA (Hu et al., 2021) gains the most popularity due to its high effectiveness, robustness and generality.

In parallel with this, dropout (Hinton et al., 2012) has been widely adopted to mitigate overfitting, which is generally caused by excessive parameter redundancy. Its variants, including DropKey (Li et al., 2023), DropAttention (Zehui et al., 2019) and HiddenCut (Chen et al., 2021), have also demonstrated superiority for transformers. With a specified probability, they randomly deactivate attention logits, weights and hidden representations, respectively. However, the effectiveness of these methods is only verified in full finetuning scenarios, where all the parameters are updated and easily lead to excessive redundancy. When it comes to LoRA-based PEFT scenarios, a potential contradiction arises. Specifically, *since overfitting primarily stems from excessive parameter redundancy, dropout may prove ineffective in LoRA-based finetuning because of the extremely limited trainable parameters.* Besides, all the above methods are proposed independently, lacking a clear guideline to unify them systematically, which hinders comprehensive comparative analysis and the development of more effective dropout methods.

In this study, we first conduct extensive experiments and confirm that LoRA also suffers from overfitting easily, which serves as a prerequisite for our following analysis. As shown in Figure 5, as the rank and trainable parameters increase, the model's performance initially improves but gradually deteriorates due to the intensifying overfitting. Much more experiments in Sec. 4 provide further evidence and affirm that this overfitting susceptibility can be improved with dropout methods. Besides, we compare the above transformer-specific dropout

methods mathematically and empirically. For the first time, we find that DropKey and DropAttention share the equivalent forwarding process, while the gradient stopping operator introduces gradient noise into the backpropagation of DropAttention, impairing the training stability and performance.

Based on the comparative analysis, we identify three key dimensions for a dropout method and derive a unified framework along dropping position, structural pattern and compensation measure. With this framework, empirical experiments firstly reveal the new preferences of these methods in LoRA scenarios. For example, span-wise HiddenCut is no longer superior to the element-wise one due to the limited tunable parameters. Secondly, this framework enables the comprehensive comparisons among different methods. Empirically, we find that DropKey performs the best followed by HiddenCut, while DropAttention exhibits the worst performance due to the gradient noise. As an alternative compensation, Bidirectional Kullback-Leibler (KL) divergence loss consistently achieves performance gains, while Jensen-Shannon (JS) consistency regularization loss becomes ineffective.

Guided by this framework, we also derive a new dropout method named HiddenKey, which drops attention logits column-wisely and hidden representations element-wisely, respectively, and augment the vanilla loss with KL loss. It consistently exhibits superiority across multiple models in both natural language understanding (NLU) and generation (NLG) tasks, which also fills the largely overlooked gap on the effect of dropout methods on NLG tasks. Integrating with input and output dropout does not provide further complementarity, demonstrating the sufficiency of our method. Hence, HiddenKey excels as the better method for high-performance and parameter-efficient finetuning of LLMs on both NLU and NLG tasks.

In summary, our contributions are mainly as follows:

- We present the first comprehensive investigation to explore the potential contradiction between various dropout methods and LoRA.
- We compare three typical transformer-specific dropout methods theoretically and empirically, and derive the core dimensions for designing a dropout method.
- We further introduce a unified framework to instantiate existing dropout methods, within which we discover the new preferences and performance comparison of these methods.
- A new dropout method named HiddenKey is devised within our framework, exhibiting superior effectiveness and sufficiency in mitigating LoRA's susceptibility to overfitting.

2 Preliminaries

In this section, we revisit three transformer-specific dropout methods shown in Figure 1, laying the foundation for the subsequent analysis.

DropAttention. DropAttention (Zehui et al., 2019) is the first dropout method specially designed for self-attention mechanism. It randomly masks elements or key columns of attention weights, encouraging the utilization of multiple contextualized features instead of overfitting some specific patterns. Following Eq. 1 and 2, normalized rescaling replaces the traditional one to guarantee the sum of attention weights to be one, and achieves better performance for multiple NLP classification tasks.

$$\overline{w}_j = m \cdot w_j, \quad m \sim \mathrm{Bernoulli}(p), \tag{1}$$

$$w'_j = \frac{\overline{w}_j}{\mathrm{NoGrad}(\sum_{j=0}^{l-1} \overline{w}_j)}, \tag{2}$$

where p, l, w_j, $\overline{w}_j$, and w'_j denote the dropout rate, sequence length, original, masked, and rescaled attention weights. $\mathrm{NoGrad}()$ and $\mathrm{Bernoulli}()$ represent the gradient stopping operator and sampling from the Bernoulli distribution, respectively[1].

DropKey. As a dropout-before-softmax scheme, DropKey (Li et al., 2023) takes attention logits g_j instead of weights as the basic units, as formulated in Eq. 3. Since the subsequent $\mathrm{softmax}()$ ensures the sum of weights to be one, rescaling is no longer necessary.

$$g'_j = m + g_j, \;\; m = \begin{cases} 0, & \text{with probability } 1-p \\ -\infty, & \text{with probability } p \end{cases} \tag{3}$$

HiddenCut. In contrast, HiddenCut (Chen et al., 2021) focuses on preventing the co-adaptation of hidden representations in the feed-forward module. The core idea is to cut single contiguous span, which may contain more semantic information and be more difficult to be restored. Besides, JS loss is applied to encourage the perturbed representations to be as close to those in inference as possible.

[1]Here we omit the subscript t for clarity. Although whether the $\mathrm{NoGrad}()$ operator exists or not significantly impacts the performance of DropAttention, it is overlooked in the original paper. We present it here and will discuss both cases in detail.

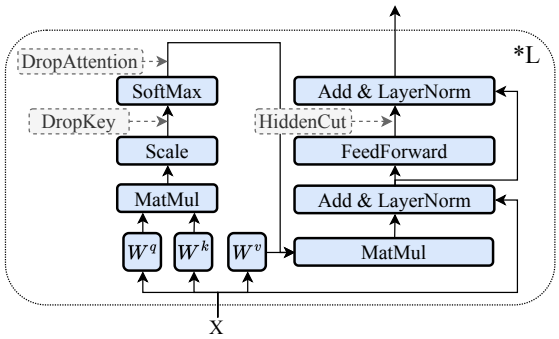
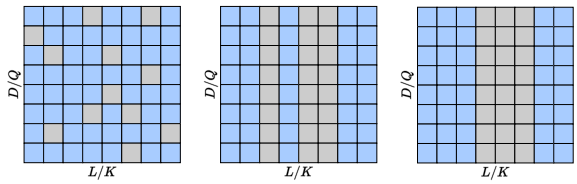


Figure 1: Illustration of transformer architecture and typical transformer-specific dropout methods, namely DropKey, DropAttention, and HiddenCut.



(a) element (b) column (c) span

Figure 2: Three structural sampling strategies, namely element, column, and span. The grey and blue cells represent masked and remaining entries, respectively. In HiddenCut, rows and columns denote sequence length (L) and hidden dimension (D), while representing keys (K) and queries (Q) in DropKey and DropAttention.

3 Method

Firstly, we conduct a comparative analysis of the above methods. Based on their similarity and differences, we then propose a unified framework along dropping position, structural pattern and compensation measure. Finally, this framework guides us to derive a new dropout method named HiddenKey, which exhibits superior performance empirically.

3.1 Mathematical and Empirical Comparison

Equivalent Forwarding between DropKey and DropAttention. Despite the different details between DropKey and DropAttention, we show their mathematical equivalence in forwarding. Let g_u and g_m denote the unmasked and masked attention logits, while w_u and w_m represent the corresponding attention weights[2]. For DropKey, we have

$$g'_m := -\infty, \quad g'_u := g_u, \quad w'_m = 0, \tag{4}$$

$$w'_u = \frac{\exp(g'_u)}{\sum_{i=0}^{l-1} \exp(g'_i)}, \tag{5}$$

while for DropAttention, we have

$$w'_m := 0, \quad w'_u = \frac{\exp(g_u)}{\sum_{i=0}^{l-1} \exp(g_i)} \cdot \frac{1}{\sum_{i=0}^{l-1} \overline{w}_i}. \tag{6}$$

[2]Only one masked element is considered here, but masking multiple elements shares the same analysis.

Proved by Eq. 13 in Appendix C, Eq. 5 and Eq. 6 are strictly equal to each other. Hence, the final attention weights (i.e., w'_u and w'_m) of DropKey are the same as those of DropAttention, and so is the following computation. Notably, normalized rescaling plays an indispensable role in establishing this equivalence, which diminishes the differences between these two methods during the forward pass.

Variation in Back-Propagation between DropKey and DropAttention. Due to the equivalent forward pass, the corresponding values of $\frac{\partial O}{\partial w'_u}$ and $\frac{\partial O}{\partial w'_m}$ remain the same for DropKey and DropAttention, where O denotes the objective function. Meanwhile, because of the identical computation before attention logits, the analysis of back-propagation focuses on the four partial derivatives of w'_u and w'_m with respect to g_u and g_m, respectively. For DropKey, we have

$$\frac{\partial w'_u}{\partial g_u} = \exp(g_u) \cdot \frac{\sum_{i=0,\neq m}^{l-1} \exp(g_i) - \exp(g_u)}{(\sum_{i=0,\neq m}^{l-1} \exp(g_i))^2}. \tag{7}$$

For DropAttention with $\mathrm{NoGrad}()$, we have

$$\frac{\partial w'_u}{\partial g_m} = -\frac{\exp(g_u) \cdot \exp(g_m)}{\sum_{i=0}^{l-1} \exp(g_i) \cdot \sum_{i=0,\neq m}^{l-1} \exp(g_i)}, \tag{8}$$

$$\frac{\partial w'_u}{\partial g_u} = \frac{\exp(g_u) \cdot \sum_{i=0,\neq u}^{l-1} \exp(g_i)}{\sum_{i=0}^{l-1} \exp(g_i) \cdot \sum_{i=0,\neq m}^{l-1} \exp(g_i)}. \tag{9}$$

As for other partial derivatives, their gradient flow is disrupted by dropping operations. When the corresponding elements of attention logits and weights are masked in DropKey and DropAttention, the derivative of w'_u with respect to g_u has proportional relation, as shown in Eq. 10 and proven by Eq. 14. Provably, k is always less than 1 and continuously decreases with the increase of g_m. In other words, compared to DropAttention with $\mathrm{NoGrad}()$, DropKey can adaptively lower the gradients when a large attention logit g_m is discarded. This can provide DropKey with dropping-dependent compensation capability, thereby stabilizing the training process. For DropAttention with $\mathrm{NoGrad}()$, the partial derivative of w'_u with respect to g_m is non-zero and that with respect to g_u depends on the value of g_m, even if w_m is masked and g_m is not used for computation. This implies that a larger dropout rate can introduce more gradient noise, which is further validated by the inferior performance in Sec. 4. In contrast, DropAttention without $\mathrm{NoGrad}()$ shares the same back-propagation with DropKey, thereby exhibiting identical behaviors. Hence, unless otherwise stated, we will refer

to DropAttention with $\mathrm{NoGrad}()$ as DropAttention, and include DropAttention without $\mathrm{NoGrad}()$ under DropKey for simplicity.

$$(\frac{\partial w'_u}{\partial g_u})^{\text{DropKey}} = k \cdot (\frac{\partial w'_u}{\partial g_u})^{\text{DropAttention}}, \quad (10)$$
$$k = \frac{1 - \frac{\exp(g_u)}{\sum_{i=0,\neq m}^{l-1} \exp(g_i)}}{1 - \frac{\exp(g_u)}{\sum_{i=0}^{l-1} \exp(g_i)}}$$

Comparison with HiddenCut. The commonality among these methods is that they all need to select a specific type of data, decide what patterns to mask, and consider how to reduce the gap between training and inference phases. In contrast, their divergences are two-fold. First, their distinct dropping positions and patterns lead to different rescaling operators. Identical to the vanilla dropout, element-wise HiddenCut amplifies hidden representations by a factor of $1/(1-p)$ for consistent scales between training and testing, while normalized rescaling is adopted by DropAttention. Due to the subsequent $\mathrm{softmax}()$, DropKey no longer utilizes any rescaling method. The other difference is that DropAttention and DropKey can be regarded as operations on weight matrices, which are utilized for the weighted summation of value vectors. Instead, HiddenCut operates directly in the hidden representations.

In summary, the comparative analysis of these methods highlights their similarities and differences, leading to the identification of key dimensions for designing a dropout method: dropping position, structural pattern and compensation measure. Subsequently, these elements are incorporated into our unified framework for further analysis.

3.2 A Unified Framework

Based on the above comparative analysis, we identify three key dimensions for a dropout method. Here we elaborate them further and instantiate these dropout methods along them below.

Dropping Position. For better generalization, a robust model needs to learn noise-resilient features. Hence, dropping position, determining where to inject noise, emerges as a primary consideration in designing dropout methods. For example, dropping inputs acts like data augmentation, dropping outputs encourages an ensemble of sub-classifiers, and dropping intermediate representations disrupts the co-adaptation of neighboring neurons. For a transformer layer depicted in Figure 1, DropKey, DropAttention and HiddenCut respectively drop attention logits, weights and hidden representations, covering the self-attention mechanism and feed-forward module. Additionally, the same dropping position may perform differently in full finetuning and LoRA scenarios. In full finetuning, weights directly associated with the dropping position are continuously adjusted for better noise resilience, while this adaptation is more implicit for LoRA. Specifically, LoRA, typically applied to the key and value projection matrices (Hu et al., 2021), requires multiple intermediate calculations to influence attention logits and weights, while even requires inter-module computation for hidden representations. This disparity may potentially affect the effectiveness of existing dropout methods in LoRA scenarios. Notably, distinct dropping positions do not necessarily indicate differences. In specific cases, different positions may also exhibit similar features, as discussed in Sec. 3.1.

Structural Pattern. Structural pattern means the style of units deactivated randomly, and determines how the co-adaptation of neurons is disrupted, thereby affecting the semantic information learned by these units. For example, as shown in Figure 2(b), if column pattern is adopted in DropKey, each value vector tend to possess as much contextual information as possible so that the output vectors are minimally affected by the masked key columns. Different patterns also result in varying levels of difficulty in recovery (Zehui et al., 2019). Generally, the span pattern is more challenging than the column style, while the element one is the simplest. Given the limited trainable parameters, LoRA may struggle to handle the strong disturbances introduced by complex patterns. Therefore, it may exhibit different preferences for structural patterns from full finetuning. Besides, different optimal patterns may be required for distinct positions, which will be thoroughly discussed in Sec. 4.

Compensation for Training and Inference Gap. For better performance and deterministic outputs, dropout is disabled in inference by default. However, this is not consistent with the training stage and can lead to a gap between the actual and ideal performance. Hence, another key consideration is how to close the training and inference gap. Apart from rescaling associated with each method intrinsically, R-drop (Wu et al., 2021) leverages Eq. 11, bidirectional KL divergence loss, to enforce the output distributions to be more dropout-insensitive so

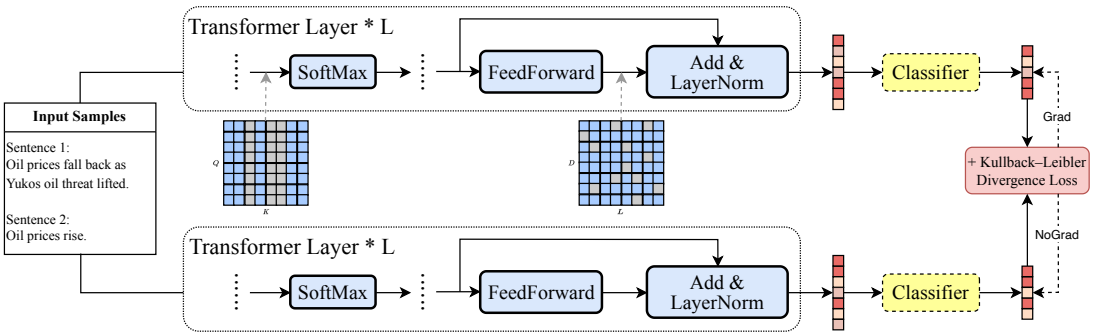


Figure 3: Illustration of HiddenKey. It respectively drops columns and elements of attention logits and hidden representations, and augments bidirectional KL loss to minimize the training and inference gap implicitly.

that the gap can be minimized implicitly. Alternatively, HiddenCut replaces it with JS loss shown in Eq. 12. With negligible tunable parameters, LoRA is more easily optimized to reach its performance ceiling. This compressed optimization space may potentially render some existing schemes ineffective, which is also verified in the following sections.

$$\mathcal{L}_{KL} = \frac{1}{2}(D_{\mathrm{KL}}(P_1 \| P_2) + D_{\mathrm{KL}}(P_2 \| P_1)), \quad (11)$$

$$\mathcal{L}_{JS} = D_{\mathrm{KL}}(P_1 \| \overline{P}), \quad (12)$$

where P_1, P_2, and $\overline{P}$ represent two different output distributions in the training stage and one in inference with the same input, respectively. For the sake of symmetry, KL loss calculates the bidirectional distances, while JS loss uses the inference distribution as reference.

3.3 HiddenKey

The proposed unified framework not only enables us to analyze the critical choices along each dimension and their mutual influences, but also guides us to design new dropout methods. As shown in Figure 3, we propose "HiddenKey", which drops the attention logits column-wisely in the attention mechanism and hidden representations element-wisely in the feed-forward module along the dropping position and structural pattern dimensions. As for the compensation measure to minimize the training and inference gap, two forward passes in parallel are performed so that an extra KL loss is deployed to enhance the similarity of output distributions. For classification tasks, the representations produced by the classifier are used, while those produced by the last transformer layer are used for regression tasks. Furthermore, the superiority over all the aforementioned methods will be extensively analyzed on diverse tasks and models below.

4 Experiments

4.1 General Setup

Models and Datasets. We implement comprehensive analysis on multiple tasks and models with LoRA. The models start from RoBERTa-large (Liu et al., 2019) and GPT2-Medium (Li and Liang, 2021), and scale up to LLaMA2-7B (Touvron et al., 2023a). Besides, both NLU and NLG tasks are covered. For NLU tasks, we utilize six datasets from GLUE benchmark (Wang et al., 2018): **SST-2** (Socher et al., 2013), **RTE** (Wang et al., 2018), **MRPC** (Dolan and Brockett, 2005), **STS-B** (Cer et al., 2017), **CoLA** (Warstadt et al., 2018), and **QNLI** (Rajpurkar et al., 2018). These datasets are selected to cover diverse tasks and sizes, including single sentence, similarity, paraphrase and inference. For NLG tasks, we follow Hu et al. (2021) and focus on **E2E** (Novikova et al., 2017) and **WebNLG** (Gardent et al., 2017). More details can be found in Appendix D.

Baseline. Due to the widespread popularity, we use vanilla LoRA as the baseline, and keep most of its configurations. Notably, low-rank decomposition with a rank of 8 and scalar of 16 is applied to the key and value projection matrices. This results in trainable parameters of 0.79M in the Roberta-large model, accounting for 0.22% of the total parameters[3]. In comparison, these values are 0.39M and 0.11% for GPT2-Medium, while 4.19M and 0.06% for LLaMA2-7B. More detailed configurations are demonstrated in the Appendix E.

4.2 Main Results

We first experiment with RoBERTa-large on four NLU datasets, and present the results in Table 1

[3]The classifier parameters are excluded here due to their varying numbers for different tasks.

Position	Pattern / Compen.	RTE Acc.	MRPC Acc.	STS-B Pearson.	STS2 Acc.	Avg.
Full Finetuning*	-	86.60	90.90	92.40	96.40	91.58
Baseline	-	$84.48_{\pm 0.98}$	$89.95_{\pm 0.50}$	$91.96_{\pm 0.48}$	$95.99_{\pm 0.25}$	90.60
HiddenCut	element	$87.00_{\pm 1.14}$	$90.69_{\pm 0.42}$	$91.94_{\pm 0.28}$	$96.10_{\pm 0.42}$	91.43
	column	$86.64_{\pm 0.80}$	$90.20_{\pm 0.80}$	$91.96_{\pm 0.11}$	$96.22_{\pm 0.19}$	91.26
	span	$86.64_{\pm 1.63}$	$90.69_{\pm 0.22}$	$92.05_{\pm 0.35}$	$96.10_{\pm 0.30}$	91.37
DropKey	element	$87.00_{\pm 1.08}$	$90.93_{\pm 1.06}$	$92.21_{\pm 0.21}$	$96.22_{\pm 0.25}$	91.59
	column	$87.36_{\pm 1.70}$	$90.93_{\pm 0.40}$	$92.25_{\pm 0.13}$	$96.22_{\pm 0.24}$	91.69
	span	$86.28_{\pm 0.94}$	$90.69_{\pm 0.69}$	$92.21_{\pm 0.21}$	$96.22_{\pm 0.25}$	91.35
DropAttention	element	$85.56_{\pm 11.73}$	$90.20_{\pm 3.07}$	$92.03_{\pm 0.27}$	$95.76_{\pm 0.30}$	90.89
	column	$85.56_{\pm 1.80}$	$90.20_{\pm 0.71}$	$92.11_{\pm 0.28}$	$95.87_{\pm 0.21}$	90.94
	span	$86.28_{\pm 0.60}$	$89.95_{\pm 0.61}$	$92.21_{\pm 0.36}$	$96.10_{\pm 0.39}$	91.14
HiddenKey^{-}	-	$87.70_{\pm 0.91}$	$90.90_{\pm 0.72}$	$92.28_{\pm 0.19}$	$96.22_{\pm 0.13}$	91.78
	+ KL	$88.10_{\pm 1.60}$	$\mathbf{91.20}_{\pm 0.90}$	$\mathbf{92.30}_{\pm 0.11}$	$\mathbf{96.44}_{\pm 0.20}$	**92.01**
	+ JS	$87.70_{\pm 1.72}$	$90.90_{\pm 0.47}$	$92.24_{\pm 0.21}$	$96.22_{\pm 0.24}$	91.77
+ input	-	$\mathbf{88.50}_{\pm 2.11}$	$90.70_{\pm 1.03}$	$92.11_{\pm 0.14}$	$96.33_{\pm 0.27}$	91.16
+ output	-	$87.70_{\pm 2.24}$	$90.70_{\pm 1.20}$	$92.19_{\pm 0.11}$	$96.22_{\pm 0.15}$	90.95

Table 1: Performance of various dropping positions, structural patterns and compensation methods for RoBERTa-large model on RTE, MRPC, STS-B and SST-2 datasets. "input" and "output" refer to the dropout of input and output representations, respectively. The subscripts denote the standard deviation, while bold indicates the best performance. "Compen." and "Avg." are abbreviations for compensation measures and the average results across four datasets.

and Figure 4. Generally, almost all methods can outperform the baseline with a large margin. This demonstrates that despite limited trainable parameters, LoRA still suffers from overfitting and these transformer-specific dropout methods can alleviate this problem. We claim that limited trainable parameters of LoRA still enable relatively large model capacity. This can stem from two aspects: (1) Even if the proportion is negligible, the number of tunable parameters remains significant due to the large size of foundation models. As mentioned earlier, there are still 0.79M tunable parameters, even if they only account for 0.22% of the whole model. (2) Coupled with the base models, the expressiveness of these parameters is enlarged extremely, as evidenced by the remarkable performance in Hu et al. (2021). This excessive model capacity contributes to the susceptibility to overfitting, despite only a negligible portion of trainable parameters.

Different dropping positions prefer distinct structural patterns. As shown in Table 1, the optimal structure for DropKey is "column", which deactivates specific keys across all queries within a head, thereby breaking the co-adaptation of value vectors and achieving better performance. Oppositely, Li et al. (2023) confirms the ineffectiveness of structural patterns in multiple CV tasks. This divergence may arise from that NLP tasks have a more semantically explicit token segmentation, while this property is absent for CV tasks. In comparison, HiddenCut only has one representation sequence instead of multiple ones in the multi-head self-attention module. Hence, "column" and "span" modes may erase too much information, especially when semantically important representations, such as emotional and negation ones, are masked. This could introduce excessive noise and even incorrect input-label pairs for more limited LoRA scenarios, and explains why element-wise HiddenCut achieves better performance on average, different from the span style for full finetuning (Chen et al., 2021).

These dropout methods exhibit different characteristics in LoRA scenarios, and combining different positions can yield further improvement. Specifically, with a small dropout rate, all methods perform very similarly, fluctuating around the baseline. However, as the dropout rate increases, DropKey consistently achieves the best performance on four datasets, followed by HiddenCut. This might be partially attributed to the closer proximity of DropKey to LoRA. In contrast, despite the similar dropping positions and the same forward pass

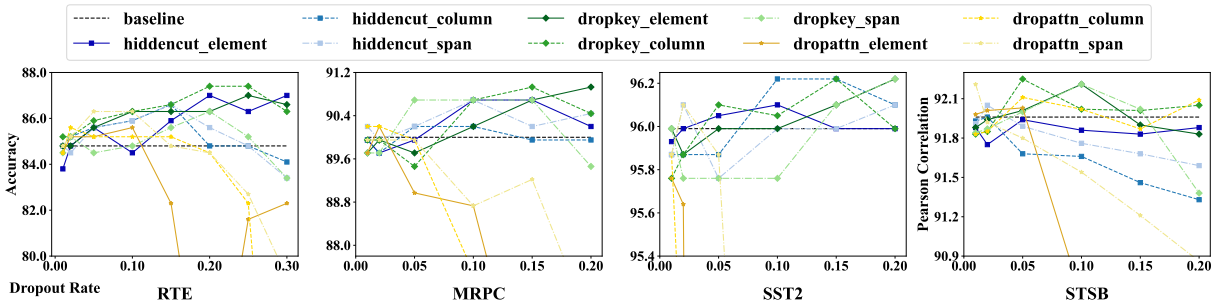


Figure 4: Performance of RoBERTa-large with different dropout methods on four NLU datasets, namely RTE, MRPC, SST-2 and STS-B. Markers and line styles differentiate various dropping positions, while the shades of color represent the structural patterns. Pearson correlation is reported for STS-B, while accuracy is utilized for others.

as DropKey, DropAttention produces the worst results. This confirms our earlier analysis in Sec. 3.1 that $\mathrm{NoGrad}()$ operator leads to larger gradient noise in back-propagation and rapid performance degradation as the dropout rate increases. Considering their best performance, we further combine element-wise HiddenCut with column-wise DropKey, named HiddenKey$^-$. On average, it achieves additional improvement over any single dropout mechanism. We also attempt to combine DropAttention, but it does not result in any benefits.

As for the compensation measures to narrow the gap between training and inference stages, KL loss consistently achieves better performance than JS loss. Specifically, compared to HiddenKey$^-$ (i.e. HiddenKey without any additional loss), the introduction of KL loss always provides extra performance gains on all the datasets. In contrast, JS loss does not have an apparent impact on the results, even if Chen et al. (2021) claims its effectiveness in full finetuning settings. This difference may arise from the more capacity-limited LoRA scenarios and superb dropout methods, which jointly squeeze the potential improvement space for augmented loss. Therefore, with the validated superiority, KL loss is incorporated into HiddenKey along the third dimension of our proposed framework. Due to the optimal practice along each dimension, HiddenKey steadily achieves the best performance among all the above methods and datasets.

4.3 Complementarity with Input and Output Dropout

In addition to DropKey, DropAttention and HiddenCut, which cover the transformer layer, cutoff is also applied to input embedding sequences for data augmentation (Shen et al., 2020), and standard dropout is used to the output representations for a more robust classifier. To comprehensively explore the impact of dropout on the entire model, we further investigate whether these methods could further enhance the transformer-specific dropout. The results at the end of Table 1 suggest that neither of these methods achieve consistent improvement over HiddenKey$^-$ across all the datasets, and both of their average performance suffers a slight decrease. This indicates that HiddenKey has predominantly captured the performance gains achieved through dropout methods, while dropping input or output does not contribute steady complementarity. This sufficiency hints that finetuning with

Model	Method	BLEU ↑	NIST ↑	METEOR ↑	ROUGE_L ↑	CIDEr ↑
GPT2-Medium	Full Finetuning*	68.20	8.620	46.20	71.00	2.470
	Baseline	$68.50_{\pm 0.90}$	$8.615_{\pm 0.09}$	$46.43_{\pm 0.26}$	$71.08_{\pm 0.25}$	$2.490_{\pm 0.02}$
	HiddenCut	$69.22_{\pm 0.44}$	$8.700_{\pm 0.05}$	$46.66_{\pm 0.11}$	$71.39_{\pm 0.07}$	$2.491_{\pm 0.01}$
	DropKey	$68.78_{\pm 0.75}$	$8.651_{\pm 0.08}$	$46.53_{\pm 0.24}$	$71.40_{\pm 0.33}$	$2.486_{\pm 0.01}$
	HiddenKey$^-$	$69.35_{\pm 0.48}$	$8.726_{\pm 0.04}$	$46.60_{\pm 0.29}$	$71.61_{\pm 0.26}$	$2.510_{\pm 0.00}$
	HiddenKey	$\mathbf{69.76}_{\pm 0.51}$	$\mathbf{8.765}_{\pm 0.08}$	$\mathbf{46.80}_{\pm 0.11}$	$\mathbf{71.78}_{\pm 0.06}$	$\mathbf{2.511}_{\pm 0.03}$
LLaMA2-7B	Baseline	$66.71_{\pm 0.65}$	$8.463_{\pm 0.09}$	$44.82_{\pm 0.26}$	$70.10_{\pm 0.46}$	$2.371_{\pm 0.01}$
	HiddenKey	$\mathbf{69.02}_{\pm 0.64}$	$\mathbf{8.725}_{\pm 0.08}$	$\mathbf{45.84}_{\pm 0.13}$	$\mathbf{71.17}_{\pm 0.13}$	$\mathbf{2.456}_{\pm 0.00}$

Table 2: Results of GPT2-Medium and LLaMA2-7B with various dropout methods on E2E NLG Challenge dataset.

Method	A			S			U		
	BLEU ↑	METEOR ↑	TER ↓	BLEU ↑	METEOR ↑	TER ↓	BLEU ↑	METEOR ↑	TER ↓
Full Finetuning*	46.50	0.380	0.530	64.20	0.450	0.330	27.70	0.300	0.760
Baseline	$54.78_{\pm 0.16}$	$0.411_{\pm 0.00}$	$0.395_{\pm 0.00}$	$62.30_{\pm 0.47}$	$0.420_{\pm 0.04}$	$0.331_{\pm 0.00}$	$45.53_{\pm 0.21}$	$0.376_{\pm 0.00}$	$0.464_{\pm 0.00}$
HiddenCut	$55.06_{\pm 0.18}$	$0.411_{\pm 0.00}$	$0.391_{\pm 0.00}$	$62.43_{\pm 0.21}$	$\mathbf{0.442}_{\pm 0.00}$	$0.329_{\pm 0.00}$	$46.11_{\pm 0.20}$	$0.377_{\pm 0.00}$	$0.458_{\pm 0.00}$
DropKey	$55.22_{\pm 0.34}$	$0.411_{\pm 0.00}$	$0.389_{\pm 0.00}$	$62.47_{\pm 0.17}$	$0.441_{\pm 0.00}$	$0.328_{\pm 0.00}$	$46.39_{\pm 0.75}$	$0.378_{\pm 0.00}$	$0.455_{\pm 0.01}$
HiddenKey$^-$	$55.26_{\pm 0.20}$	$0.411_{\pm 0.00}$	$0.388_{\pm 0.00}$	$\mathbf{62.57}_{\pm 0.24}$	$0.441_{\pm 0.00}$	$0.328_{\pm 0.00}$	$46.36_{\pm 0.34}$	$0.378_{\pm 0.00}$	$0.454_{\pm 0.00}$
HiddenKey	$\mathbf{55.27}_{\pm 0.21}$	$\mathbf{0.413}_{\pm 0.00}$	$\mathbf{0.386}_{\pm 0.00}$	$62.49_{\pm 0.18}$	$0.441_{\pm 0.00}$	$\mathbf{0.326}_{\pm 0.00}$	$\mathbf{46.48}_{\pm 0.46}$	$\mathbf{0.381}_{\pm 0.00}$	$\mathbf{0.452}_{\pm 0.00}$

Table 3: Results of GPT2-Medium finetuned with different dropout methods on WebNLG dataset. "A", "S" and "U" correspond to the "All", "Seen" and "Unseen" categories in the test set, respectively.

HiddenKey only is enough in LoRA scenarios.

4.4 Superiority on More NLU and NLG Tasks

Method	CoLA	QNLI
	Matthew.	Acc.
baseline	$67.96_{\pm 0.25}$	$94.23_{\pm 0.17}$
HiddenKey	$\mathbf{69.91}_{\pm 0.52}$	$\mathbf{95.04}_{\pm 0.11}$

Table 4: Results of RoBERTa-large finetuned with HiddenKey on CoLA and QNLI datasets.

More NLU Datasets. We further generalize HiddenKey to two extra NLU datasets, namely CoLA and QNLI. As shown in Table 4, HiddenKey steadily achieves 1.95 and 0.81 performance improvement over baselines on both of the datasets. These reconfirm the superior performance of HiddenKey in NLU tasks.

NLG datasets. Following Hu et al. (2021), we also experiment with GPT2-Medium on NLG tasks. As shown in Table 2, HiddenKey consistently outperforms full finetuning, LoRA baseline and other dropout methods over all the five metrics on E2E NLG Challenge dataset. Similarly in Table 3, on the "All", "Seen" and "Unseen" subsets of the WebNLG dataset, HiddenKey gains 7/9 wins over all other methods on BLEU, METEOR and TER metrics. Hence, HiddenKey exhibits a performance surge across diverse metrics, datasets and their subsets for NLG tasks, as it has shown for NLU tasks.

4.5 Performance Boost on LLMs

With the dominance of LLMs, we also extend the application of HiddenKey to LLaMA2-7B, one of the most popular and open-sourced LLMs, on both NLU and NLG tasks. As shown in Table 5, models finetuned with HiddenKey outperform those without HiddenKey by a large margin on RTE and MRPC datasets. Similarly, HiddenKey consistently

Method	RTE	MRPC
	Acc.	Acc.
baseline	$88.45_{\pm 0.79}$	$88.73_{\pm 0.56}$
HiddenKey	$\mathbf{90.25}_{\pm 1.05}$	$\mathbf{89.46}_{\pm 0.60}$

Table 5: Results of LLaMA2-7B finetuned with HiddenKey on RTE and MRPC datasets.

exhibits significant superiority on E2E NLG dataset across all metrics over baseline, shown at the end of Table 2. This indicates that HiddenKey can also function well with LLMs on diverse tasks.

4.6 Ablation Study

Based on our framework, we eliminate the components of HiddenKey to demonstrate the necessity of each dimension. As illustrated in Table 1, 2 and 3, the substantial boost of HiddenKey$^-$ over previous methods and baselines on both NLU and NLG tasks indicates the significance of dropping positions and patterns in mitigating the susceptibility to overfitting in LoRA scenarios. Moreover, HiddenKey also consistently outperforms HiddenKey$^-$, emphasizing the importance of appropriate compensation measures. These results provide strong evidence for the effectiveness of our framework.

5 Conclusion

We investigate the possible contradiction between the limited trainable parameters of LoRA and overfitting associated excessive parameter redundancy. After confirming the overfitting-prone property of LoRA, we analyze existing dropout methods theoretically and empirically, and further introduce a unified framework for thorough comparison. This also guides us to derive a new dropout method, HiddenKey. With its superiority and sufficiency across multiple models and datasets, HiddenKey deserves to be the recommended dropout method to alleviate overfitting in LoRA-based scenarios.

6 Limitation

The main limitation of this work is the potentially longer training duration incurred by the Bidirectional Kullback-Leibler (KL) divergence loss. Specifically, the calculation of the KL loss requires the output distributions of two forward passes. In our implementation, as shown in Figure 3, we only perform back-propagation on one of the branches, resulting in approximately 50% longer training time compared to the original training process. However, we argue that this can be greatly reduced by parallelizing the two forward passes with multiple processes. Alternatively, both branches can be back-propagated simultaneously or sequentially, before merging their gradient updates. This pipeline can be regarded as utilizing the same batch of samples twice, thereby roughly halving the number of iterations and resulting in similar total training time, which is left for future work. Furthermore, it is worth noting that the training cost is one-time, and the introduction of KL loss can significantly improve models' performance, which is highly beneficial for performance-critical scenarios. On the other hand, for training cost-sensitive scenarios, using only HiddenKey$^-$ (i.e. HiddenKey without KL loss) can still outperform the baselines. Hence, we claim that despite the potential increase in training duration, HiddenKey and HiddenKey$^-$ do provide available options for different scenarios.

7 Ethics Statement

We strictly follow the ACL Code of Ethics during the research. To the best of our knowledge, there are no foreseeable potential risks in the methods we introduced. We report the computing infrastructure for all computational experiments presented in the paper. The transparent statistics on our results and detailed configuration of our experimental setup, including best-found hyperparameter values, are well stated. Besides, we will also release the code upon publication for publicly available reproducibility with minimal effort.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Jiaao Chen, Dinghan Shen, Weizhu Chen, and Diyi Yang. 2021. Hiddencut: Simple data augmentation for natural language understanding with better generalizability. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4380–4390.

Yuxuan Chen, Rongpeng Li, Zhifeng Zhao, Chenghui Peng, Jianjun Wu, Ekram Hossain, and Honggang Zhang. 2023. Netgpt: A native-ai network architecture beyond provisioning personalized generative services. *arXiv preprint arXiv:2307.06148*.

WilliamB. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Bonan Li, Yinhan Hu, Xuecheng Nie, Congying Han, Xiangjian Jiang, Tiande Guo, and Luoqi Liu. 2023. Dropkey for vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22700–22709.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th*

Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 61–68.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Michael Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Cornell University - arXiv,Cornell University - arXiv*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *Proceedings of the SIGDIAL 2017 Conference, pages 201-206, Saarbrücken, Germany, 15-17 August 2017*.

OpenAI. 2023. Gpt-4 technical report.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*.

Richard Socher, Alex Perelygin, JeanY. Wu, Jason Chuang, ChristopherD. Manning, AndrewY. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *Empirical Methods in Natural Language Processing,Empirical Methods in Natural Language Processing*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. 2023. Multilora: Democratizing lora for better multi-task learning. *arXiv preprint arXiv:2311.11501*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments.

Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021. R-drop: Regularized dropout for neural networks. *Advances in Neural Information Processing Systems*, 34:10890–10905.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Lin Zehui, Pengfei Liu, Luyao Huang, Junkun Chen, Xipeng Qiu, and Xuanjing Huang. 2019. Dropattention: A regularization method for fully-connected self-attention networks. *arXiv preprint arXiv:1907.11065*.

A Related work

During the finetuning phase, full-finetuning involves updating all model parameters, resulting in a slightly modified version. However, with the rapid development of large language models (LLMs), this approach becomes increasingly impractical due to the high storage and inference expenses, particularly in multitask and personalized settings (Wang et al., 2023; Chen et al., 2023). As lightweight alternatives, parameter-efficient finetuning (PEFT) methods only introduce or retrain a negligible portion of parameters, sharing most of the parameters while preserving competitive performance as full-finetuning (Houlsby et al., 2019; Lester et al., 2021; Hu et al., 2021). For instance, Houlsby et al. (2019) inserts and exclusively updates new adapters between pretrained layers, achieving remarkable performance with limited trainable parameters. However, this method increases the model's depth and incurs higher time latency. Lester et al. (2021) prefixes a learnable prompt to the input and feeds this longer sequence into the frozen model. Nevertheless, this approach reduces the available sequence length and is empirically shown to be sensitive to initialization. Similarly, Li and Liang (2021) attaches prefixed tokens to the key and value sequences, addressing the first drawback but still suffering from the latter one. In contrast, BitFit (Zaken et al., 2021) only adjusts the biases, effectively avoiding the aforementioned problems. However, its limited capacity leads to inferior performance. More recently, LoRA (Hu et al., 2021) imposes a low-rank decomposition on weight updates, which can be optionally merged into the original weights during inference, and avoids all the aforementioned issues.

Dropout (Hinton et al., 2012) randomly deactivates each neuron with a specific probability during training, which can prevent the co-adaptation of neurons and has been extended to improve the performance of transformer models (Zehui et al., 2019; Chen et al., 2021; Li et al., 2023). Specifically, Zehui et al. (2019) proposes the first variant specially designed for self-attention mechanism, DropAttention, which drops the attention weights randomly and applies normalized rescaling to ensure their sum to be one. Instead, HiddenCut (Chen et al., 2021) applies contiguous span-style masks to hidden representations in the feed-forward module. Recently, Li et al. (2023) introduces a drop-before-softmax scheme, HiddenKey, which drops key units before the softmax layer so that the sum of attention weights can be kept as one automatically. However, it only focuses on computer vision tasks, while totally neglecting NLP tasks that emphasizes semantics and linguistic information. During inference, dropout is usually disabled by default for better performance and deterministic outputs. However, this is not consistent with the training stage and can lead to a gap between the actual and ideal performance. In order to address this divergence, R-Drop (Wu et al., 2021) minimizes the bidirectional Kullback-Leibler divergence between the output distributions of two forward passes with dropout for more noise-resilient outputs. In comparison, Shen et al. (2020) narrows this gap by applying Jensen-Shannon Divergence loss to enforce consistent representations between outputs with and without dropout.

B Overfitting-Prone Property of LoRA

As an illustrative example, Figure 5 shows the evaluation accuracy of LoRA with different ranks on the RTE dataset. This clearly indicates that with the increase of the rank and trainable parameters, the performance of LoRA initially improves and then deteriorates due to progressively excessive parameter redundancy, demonstrating the susceptibility to overfitting in LoRA scenarios.

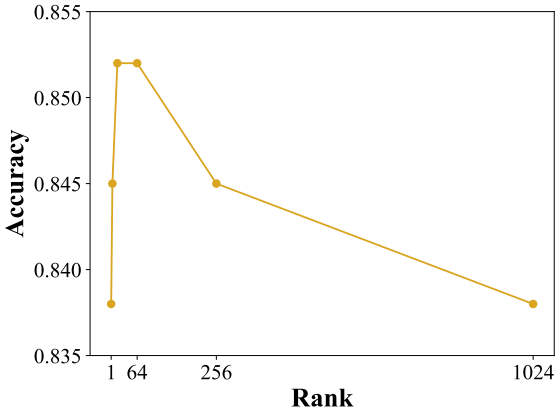


Figure 5: Evaluation accuracy of LoRA with respect to the rank on RTE dataset.

C Mathematical Proofs

We prove the mathematical equivalence of w'_u for DropKey and DropAttention as follows:

$$\begin{aligned} & \frac{\exp(g_u)}{\sum_{i=0}^{l-1}\exp(g_i)} \cdot \frac{1}{\sum_{i=0}^{l-1}\overline{w}_i} \\ = & \frac{\exp(g_u)}{\sum_{i=0}^{l-1}\exp(g_i)} \cdot \frac{1}{1-w_m} \\ = & \frac{\exp(g_u)}{\sum_{i=0}^{l-1}\exp(g_i)} \cdot \frac{1}{1-\frac{\exp(g_m)}{\sum_{i=0}^{l-1}\exp(g_i)}} \end{aligned} \tag{13}$$

$$
\begin{aligned}
&= \frac{\exp(g_u)}{\sum_{i=0}^{l-1}\exp(g_i) - \exp(g_m)} \\
&= \frac{\exp(g_u)}{\sum_{i=0,\neq m}^{l-1}\exp(g_i)} \\
&= \frac{\exp(g'_u)}{\sum_{i=0}^{l-1}\exp(g'_i)}
\end{aligned}
$$

The proportional relationship of $\frac{\partial w'_u}{\partial g_u}$ between DropKey and DropAttention can be derived with the following equation:

$$
\begin{aligned}
&\frac{(\frac{\partial w'_u}{\partial g_u})^{\text{DropKey}}}{(\frac{\partial w'_u}{\partial g_u})^{\text{DropAttention}}} \\
&= \frac{\exp(g_u)\cdot(\sum_{i=0,\neq m}^{l-1}\exp(g_i) - \exp(g_u))}{(\sum_{i=0,\neq m}^{l-1}\exp(g_i))^2} \\
&\quad \cdot \frac{\sum_{i=0}^{l-1}\exp(g_i)\cdot\sum_{i=0,\neq m}^{l-1}\exp(g_i)}{\exp(g_u)\cdot\sum_{i=0,\neq u}^{l-1}\exp(g_i)} \\
&= \frac{\sum_{i=0,\neq m}^{l-1}\exp(g_i) - \exp(g_u)}{\sum_{i=0,\neq m}^{l-1}\exp(g_i)} \\
&\quad \cdot \frac{\sum_{i=0}^{l-1}\exp(g_i)}{\sum_{i=0,\neq u}^{l-1}\exp(g_i)} \\
&= \frac{1 - \frac{\exp(g_u)}{\sum_{i=0,\neq m}^{l-1}\exp(g_i)}}{1 - \frac{\exp(g_u)}{\sum_{i=0}^{l-1}\exp(g_i)}}
\end{aligned} \tag{14}
$$

Denoting k as the result of Eq. 14, we have

$$
\begin{aligned}
k &< \frac{1 - \frac{\exp(g_u)}{\sum_{i=0,\neq m}^{l-1}\exp(g_i)+\exp(g_m)}}{1 - \frac{\exp(g_u)}{\sum_{i=0}^{l-1}\exp(g_i)}} \\
&= 1
\end{aligned} \tag{15}
$$

D Dataset Details

For NLU tasks, (i) Stanford Sentiment Treebank (**SST-2**) (Socher et al., 2013) is an English sentiment classification benchmark for a single sentence task, predicting whether the sentiment of movie reviews is positive or not. (ii) Recognizing Textual Entailment (**RTE**) (Wang et al., 2018) presents an inference task that predicts the entailment relation between two sentences. (iii) Microsoft Research Paraphrase Corpus (**MRPC**) (Dolan and Brockett, 2005) predicts the semantic equivalence between two sentences, while (iv) Semantic Textual Similarity Benchmark (**STS-B**) (Cer et al., 2017) predicts the similarity between two sentences. The later two tasks are involved with comparing and assessing the similarity and paraphrasing of two sentences. Notably, compared to the other classification tasks, STS-B performs a regression task and thus encompasses a broad range of tasks, enhancing the generalizability of our conclusions. Besides, additional experiments are further conducted on (v) Corpus of Linguistic Acceptability (**CoLA**) (Warstadt et al., 2018), which aims to predict whether a sentence is linguistically acceptable or not, and (vi) Question Natural Language Inference (**QNLI**) (Rajpurkar et al., 2018), which predicts whether a sentence is the answer to a given question. For NLG tasks, we focus on (vii) **E2E** NLG Challenge (Novikova et al., 2017) and (viii) **WebNLG** (Gardent et al., 2017). The former consists of sets of slot-value pairs along with multiple corresponding natural language references in the restaurant domain, while the later is a dataset where models generate the corresponding description in form of natural language text given a sequence of SUBJECT-PROPERTY-OBJECT triples.

As for the evaluation metrics, we report the Pearson correlation for STS-B, Matthew's correlation for CoLA, and accuracy for other NLU datasets. For NLG tasks, BLEU, NIST, METEOR, ROUGE-L and CIDEr are used on the E2E NLG Challenge dataset, while BLEU, METEOR and TER are evaluated separately for "Unseen", "Seen" and "All" categories in the test set of the WebNLG dataset.

E Hyperparameter Configuration

As shown in Table 6 and 7, we mainly follow the setup of LoRA (Hu et al., 2021) with as minimal changes as possible. However, based on our pre-experiments, significant fluctuations of the results are observed when models are trained with the original epochs, even if only random seeds change. Therefore, we increase the number of training epochs for more steady results. We also use the regular initialization instead of the MNLI checkpoint for LoRA modules. Different from RoBERTa-large and GPT2-Medium models, we employ FP16 mixed precision training for LLaMA2-7B to reduce the memory consumption, and set the epoch to one. Besides, we utilize greedy search with length penalty of 1.0 and "no repeat n-gram size" of 0 for inference, which empirically outperforms the settings of GPT2-Medium.

For the specific parameters in our experiments, we disable dropout in baselines and iterate all available dropout rate from {0.01, 0.02, 0.05, 0.1, 0.15, 0.2} for various dropout methods, which is expanded with {0.25, 0.3} for clearer trend of performance in RTE dataset. To the best of our knowledge, neither of HiddenCut, DropKey and DropAttention implements experiments with a ca-

Model	RoBERTa-large						LLaMA2-7B	
Dataset	RTE	MRPC	STS-B	SST-2	CoLA	QNLI	RTE	MRPC
Optimizer	AdamW						AdamW	
Weight Decay	0.1						0.1	
Warmup Ratio	0.06						0.06	
LR Schedule	Linear						Linear	
Learning Rate	4E-4	3E-4	3E-4	4E-4	2E-4	2E-4	5E-4	
Epoch	30	30	10	10	40	10	10	8
Batch Size	64	32	32	64	32	32	64	32
Mac Seq. Len.	512	512	128	512	128	512	512	
LoRA Rank	$r_q = r_v = 8$						$r_q = r_v = 8$	
LoRA Scalar	16						16	

Table 6: Hyperparameters for RoBERTa-large and LLaMA2-7B models with LoRA on NLU datasets.

Dataset	E2E NLG Challenge	WebNLG
Training		
Optimizer	AdamW	
Weight Decay	0.01	
Warmup Step	500	
LR Schedule	Linear	
Learning Rate	2E-4	
Epoch	5	
Batch Size	8	
Label Smooth	0.1	
LoRA Rank	$r_q = r_v = 4$	
LoRA Scalar	32	
Inference		
Beam Size	10	
Length Penalty	0.9	0.8
No Repeat N-Gram Size	4	
Repetition Penalty	1.0	

Table 7: Hyperparameters for GPT2-Medium with LoRA on NLG datasets.

sual decoder-only transformer model before. Based on our empirical observation, applying any of these methods can only produce limited improvement or even degradation on both NLU and NLG tasks, and the results are extremely sensitive to the dropout rate. This phenomenon might be caused by fragile shallow forwarding pass. In other words, noise introduced by dropout methods can be amplified with the propagation and diminish the benefits brought by dropout. Hence, we only introduce the dropping in the latter half of layers in decoder-only models and the apparent performance improvement emerges again. Besides, our pre-experiments demonstrate that a weight between 0.01 and 10 for KL and JS loss generally yields the best results. Therefore, we iterate the weight within {0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10}. All experiments are repeated 5 times on a NVIDIA V100 GPU to calculate the median values for NLU tasks, while the average values of three runs on a NVIDIA A100 GPU is reported for NLG tasks.

F Finetuning dynamics

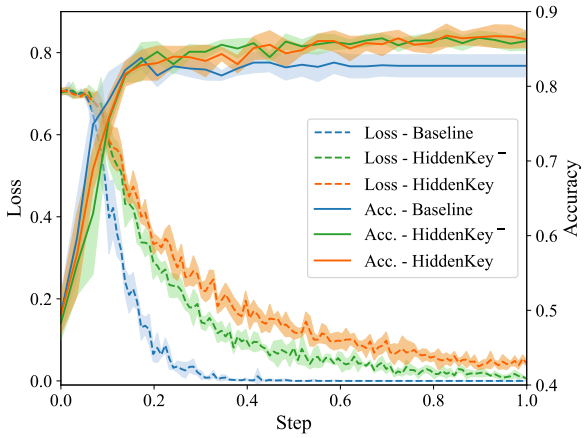


Figure 6: Finetuning loss and evaluation accuracy for baseline, HiddenKey$^-$ and HiddenKey.

Beyond the superior performance of HiddenKey, we also visualize the finetuning dynamics for a deeper understanding. Figure 6 presents the average dynamic curves of training loss and evaluation accuracy across five random seeds for multiple methods on the RTE dataset. Compared to the baseline whose training loss rapidly converges to near zero, the introduction of HiddenKey$^-$ (i.e. column-wise DropKey and element-wise HiddenCut) slows down this process and leads to larger final loss. However, large final loss does not mean inferior performance. Specifically, after reaching a fair peak

value, accuracy of the baseline deteriorates with the continuous loss decline. This hints that the models suffer from overfitting, which further supports our earlier analysis. In contrast, HiddenKey^{-} reaches the peak accuracy slightly slowly but remains superior to the baseline. With the additional KL loss, the accuracy keeps fluctuating upwards and achieves the best performance. It can be anticipated that a longer finetuning process would result in higher accuracy for HiddenKey. In summary, LoRA-based PEFT scenarios are still overfitting-prone, while HiddenKey can provide excellent model regularization in such settings, and continues improving the performance when further finetuning is allowed.