DropKey for Vision Transformer

Bonan Li*, Yinhan Hu*, Xuecheng Nie‡, Congying Han†, Xiangjian Jiang‡, Tiande Guo*, Luoqi Liu‡

*University of Chinese Academy of Sciences  ‡MT Lab, Meitu Inc.  ‡University of Cambridge

{libonan, hancy, tdguo}@ucas.ac.cn, huyinhan16@mails.ucas.ac.cn
{nxc, llq5}@meitu.com, xj265@cam.ac.uk

Abstract

In this paper, we focus on analyzing and improving the dropout technique for self-attention layers of Vision Transformer, which is important while surprisingly ignored by prior works. In particular, we conduct researches on three core questions: First, what to drop in self-attention layers? Different from dropping attention weights in literature, we propose to move dropout operations forward ahead of attention matrix calculation and set the Key as the dropout unit, yielding a novel dropout-before-softmax scheme. We theoretically verify that this scheme helps keep both regularization and probability features of attention weights, alleviating the overfitting problem to specific patterns and enhancing the model to globally capture vital information; Second, how to schedule the drop ratio in consecutive layers? In contrast to exploit a constant drop ratio for all layers, we present a new decreasing schedule that gradually decreases the drop ratio along the stack of self-attention layers. We experimentally validate the proposed schedule can avoid overfitting in low-level features and missing in high-level semantics, thus improving the robustness and stableness of model training; Third, whether need to perform structured dropout operation as CNN? We attempt patch-based block-version of dropout operation and find that this useful trick for CNN is not essential for ViT. Given exploration on the above three questions, we present the novel DropKey method that regards Key as the drop unit and exploits decreasing schedule for drop ratio, improving ViTs in a general way. Comprehensive experiments demonstrate the effectiveness of DropKey for various ViT architectures, e.g. T2T, VOLO, CeiT and DeiT, as well as for various vision tasks, e.g., image classification, object detection, human-object interaction detection and human body shape recovery.

1. Introduction

Vision Transformer (ViT) [6] has achieved great success for various vision tasks, e.g., image recognition [7, 12, 20, 34, 35], object detection [1], human body shape estimation [18], etc. Prior works mainly focus on researches of patch division, architecture design and task extension. However, the dropout technique for self-attention layer, which plays the essential role to achieve good generalizability, is surprisingly ignored by the community.

Different from the counterpart for Convolutional Neural Networks (CNNs), the dropout in ViT directly utilizes the one in original Transformer designed for Natural Language Processing, which sets attention weights as the manipulation unit with a constant dropout ratio for all layers. Despite of its simplicity, this vanilla design faces three major problems. First, it breaks the probability distribution of attention weights due to the averaging operation on non-dropout units after softmax normalization. Although this regularizes the attention weights, it still overfits specific patterns locally due to the failure on penalizing score peaks, as shown in Fig. 1 (a) and (b); Second, the vanilla design is sensitive to the constant dropout ratio, since high ratio occurs missing of semantic information in high-level representations while low ratios overfitting in low-level features, resulting in the unstable training process; Third, it ignores the structured characteristic of input patch grid to ViT, which plays an effective role to improve performance with blockwise dropout in CNNs. These three problems degrade the performance and limit the generalizability of ViTs.

Motivated by this, we propose to analyze and improve
the dropout technique in self-attention layer, further pushing forward the frontier of ViTs for vision tasks in a general way. Specifically, we focus on three core aspects:

**What to drop in self-attention layer** Different from dropping attention weights as in the vanilla design, we propose to set the Key as the dropout unit, which is essential input of self-attention layer and significantly affects the output. This moves the dropout operation forward before calculating the attention matrix as shown in Fig. 1 (c) and yields a novel dropout-before-softmax scheme. This scheme regularizes attention weights and keeps their probability distribution at the same time, which intuitively helps penalize weight peaks and lift weight foots. We theoretically verify this property via implicitly introducing an adaptive smoothing coefficient for the attention operator from the perspective of gradient optimization by formulating a Lagrange function. With the dropout-before-softmax scheme, self-attention layers can capture vital information in a global manner, thus overcoming the overfitting problem to specific patterns occurred in the vanilla dropout and enhancing the model generalizability as visualization of feature map in Fig. 1 (c). For the training phase, this scheme can be simply implemented by swapping the operation order of softmax and dropout in vanilla design, which provides a general way to effectively enhance ViTs. For inference phase, we conduct an additional finetune phase to align the expectations to training phase, further improving the performance.

**How to schedule the drop ratio** In contrast to exploiting a constant drop ratio for all layers, we present a new linear decreasing schedule that gradually decreases the drop ratio along the stack of self-attention layers. This schedule leads to a high drop ratio in shallow layers while the low one in deep layers, thus avoiding overfittings to low-level features and preserving sufficient high-level semantics. We experimentally verify the effectiveness of the proposed decreasing schedule for drop ratio to stable the training phase and improve the robustness.

**Whether need to perform structured drop** Inspired by the DropBlock [10] method for CNNs, we implement two structured versions of the dropout operation for ViTs: the block-version dropout that drops keys corresponding to contiguous patches in images or feature maps; the cross-version dropout that drops keys corresponding to patches in horizontal and vertical stripes. We conduct thorough experiments to validate their efficacy and find that the structure trick useful for CNN is not essential for ViT, due to the powerful capability of ViT to grasp contextual information in full image range.

Given exploration on the above three aspects, we present a novel DropKey method that utilizes Key as the drop unit and decreasing schedule for drop ratio. In particular, DropKey overcomes drawbacks of the vanilla dropout technique for ViTs, improving performance in a general and effective way. Comprehensive experiments on different ViT architectures and vision tasks demonstrate the efficacy of DropKey. Our contributions are in three folds: First, to our best knowledge, we are the first to theoretically and experimentally analyze dropout technique for self-attention layers in ViT from three core aspects: drop unit, drop schedule and structured necessity; Second, according to our analysis, we present a novel DropKey method to effectively improve the dropout technique in ViT. Third, with DropKey, we improve multiple ViT architectures to achieve new SOTAs on various vision tasks.

### 2. Related Work

**Vision Transformers** Inspired by the Transformer architectures in NLP, some works introduce attention block to replace convolution layers to model long-range dependencies [6, 11, 17, 29, 30, 31, 37]. ViT[6] is the pioneering work which splits an image to non-overlapping patches and then these patches are fed to a transformer to evaluate attention scores. Prior works mainly focus on patch division, architecture design and tasks extension. For example, CvT[31] proposes a hybrid architecture to mix convolutional with attention layers to introduce local inductive bias. DropConnect [28] randomly masks a subset of weights within the model. For RNNs, the first prominent research on dropout is presented in [8], which is equipped with a learnable dropout rate. For GNNs, to alleviate over-fitting and over-smoothing issues, DropEdge [23] randomly removes part of edges in the input stage at each training epoch and theoretically demonstrates the effect of the proposed method. For ViTs, most original publicly code of multiple popular vision transformer simply intuitively apply dropout operation on attention matrix.

**Dropout** Dropout is a common technique for improving the generalizability of neural networks, e.g., CNNs [16], RNNs [21] and GNNs [24]. For CNNs, [10] points out the lack of success of dropout for convolutional layers is due to dense information flowing and then proposes a form of structured Dropout. DropConnect [28] randomly masks a subset of weights within the model. For RNNs, the first prominent research on dropout is presented in [8], which is equipped with a learnable dropout rate.

**DropAttention** [36] presents a novel way for transformer in NLPs. It refers to perform randomly drop on the attention matrix. There are two two main forms of applying DropAttention for the transformer training, DropAttention-unit and DropAttention-elements. Similar to the standard Dropout [25], DropAttention-unit randomly drops the unit, that is, these dropped token will not be used as attend patches for each query patch. As the general form of DropAttention-unit, DropAttention-elements randomly drops elements in attention weights matrix which
is similar to DropConnect [28]. Different from standard Dropout [25], DropAttention introduce re-normalize to guarantee the sum of attention weights to 1 and help training process to be more steady. However, there is no further discussion and theoretical analysis on this phenomenon.

3. Method

Our generic DropKey is inspired by DropAttention and proposed for self-attention operator in vision transformer. The main idea of DropKey is to adaptively adjust attention weight to obtain a smoother attention vector. In this section, we start by introducing the theoretical explanation of DropKey, then the implementation will be explained in detail.

3.1. Methodology

As discussed in the above section, transformer-based model tends to rely on local features rather than general global information. To alleviate this issue, we propose to reduce local-bias by encouraging models to learn a smoother attention weight for each patch. To this end, we attempt to reduce the attention weight of the patch which has a large attention weight, and vice versa. Nevertheless, it is tedious and difficult to achieve this by explicitly setting rules. In this work, we found DropKey achieves the above implicitly by normalizing the attention vector that has performed the dropout operation. Specifically, by given an image $I \in \mathbb{R}^{H \times W \times C}$, where $H$ denotes height, $W$ denotes width and $C$ denotes channels, Vision Transformer architecture starts by dicing it into $n_h \times n_w$ patches $x \in \mathbb{R}^{n_c}$. Then, $x$ is used as input of self-attention layer and the output $o$ can be computed as follows

$$o = \sum_{j=1}^{n_hn_w} \left( \frac{d_jp_j}{\sum_{j=1}^{n_hn_w} d_jp_j} \right) v_j$$

$$p_j = \frac{\exp(q_j^T k_j / \text{scale})}{\sum_{j=1}^{n_hn_w} \exp(q_j^T k_j / \text{scale})}$$

where $q_i = F_q(x), k_i = F_k(x), v_i = F_v(x)$, and $q_j, k_j, v_j$ denote query of $i^{th}$ patch, key of $j^{th}$ patch, value of $j^{th}$ patch and $o$ denotes the output of one patch. $F_q(\cdot), F_k(\cdot), F_v(\cdot)$ denote projection layers with weights of dimensions $n_c \times n_c$. scale denotes scaling factor and is set to $\sqrt{n_c}$. $d$ denotes dropout ratio and $d_j \sim \text{Bernoulli}(1 - d)$. Here, for convenience, we focus on one head and omit the index of patch. By performing re-normalize, DropKey has the ability to adaptively adjust attention weight to smoother. Next, we provide theoretical analysis of our method to demonstrate its effectiveness.

Here, we formulate the expectation output of model via introducing DropKey in training stage as:

$$E_{d_j, 1 \leq j \leq n_hn_w}[o] = \sum_{j=1}^{n_hn_w} \frac{d_jp_j}{\sum_{j=1}^{n_hn_w} d_jp_j} v_j = \sum_{j=1}^{n_hn_w} c_jv_j$$

where $c_j = \frac{d_j}{\sum_{j=1}^{n_hn_w} d_jp_j} > 0$ is a smoothing coefficient which is related to $d_j$ and $p_j$.

Note that $c$ in Equ (3) can be considered to add an additional smoothing prior to distribution $p$, i.e., $c_s < c_i$ when $p_s > p_i, 1 \leq s, t \leq n_hn_w$. The specific proof is as follows:

$$c_s - c_i = \sum_{j=1}^{n_hn_w} \frac{d_j}{\sum_{j=1}^{n_hn_w} d_jp_j} - \sum_{j=1}^{n_hn_w} \frac{d_j}{\sum_{j=1}^{n_hn_w} d_jp_j} = \sum_{j=1}^{n_hn_w} \frac{d_j}{\sum_{j=1}^{n_hn_w} d_jp_j} - \sum_{j=1}^{n_hn_w} \frac{d_j}{\sum_{j=1}^{n_hn_w} d_jp_j} + 1 - \sum_{j=1}^{n_hn_w} \frac{d_j}{\sum_{j=1}^{n_hn_w} d_jp_j} + 1 \leq 0$$

It is natural to find $c$ serves as a factor to implicitly encourage the model properly to reduce the consideration of the patch with large $p$ and improve the effectiveness of the patch with small attention weight. Meanwhile, $c$ can be adjusted adaptively according to the distribution of samples in the training stage without any manual design. Nevertheless, a noteworthy problem is that since we removed DropKey in the inference phase, the output expectations in training and inference stage are inconsistent, which will decrease the performance. In Section 3.2, we propose two methods to alleviate this problem.
Having explored the implicit regularization effects of DropKey, we also demonstrate it from the perspective of gradient optimization. For simplicity, we start by considering a simple but universal optimization objective as follows:

$$
\min_{p_j, v_j} \frac{1}{2} \sum_{j=1}^{N} p_j v_j - y ||^2 \text{ s.t. } p_j > 0, \sum_{j=1}^{N} p_j = 1, 1 \leq j \leq N
$$

where $p_j \in \mathbb{R}$, $v_j \in \mathbb{R}^r$ denotes learnable parameters and $y \in \mathbb{R}^r$ denotes target. Since all parameters are randomly initialized, it can be assumed that the initial solution is far from the optimal solution, that is, $p_1^{(0)} = p_2^{(0)} = \cdots = p_N^{(0)} = \frac{1}{N}, |p_j^{(0)}| \ll M$, $1 \leq j \leq N$. Consequently, the first additive term plays the leading role in Eq. (11). Due to $\frac{\partial L}{\partial p_j} < \frac{\partial L}{\partial p_i}$, the update speed of $p_s$ is faster than $p_i$, and then lead $p_s > p_i$, $\frac{\partial L}{\partial p_s} < \frac{\partial L}{\partial p_i}$. Based on the above analysis, it can be concluded that a larger $p$ would promote $\beta$ larger, while a larger $\beta$ would further promote $p$ larger. Finally, the output of model would be controlled by a few sparse blocks. Conversely, DropKey avoids suffering local-bias by introducing the parameter $c$ to enforce distribution $p$ to be smoother.

### 3.2. Implementation

**What to drop?** Different from DropAttention, we integrate Dropout and re-normalize into one stage by dropping key rather than weight. At each training iteration, DropKey masks a certain rate of keys of the input key map by random. It is worth noting that we generate masked key map for each query, instead of sharing the same masked key map for all query vectors. Specifically, given the query $Q \in \mathbb{R}^{n_h \times n_w \times n_c}$, key $K \in \mathbb{R}^{n_h \times n_w \times n_c}$ and value $V \in \mathbb{R}^{n_h \times n_w \times n_c}$ of a feature map, it first computes the dot product of the query with all keys and divide each by scaling factor $scale = \sqrt{n_c}$. Subsequently, we randomly generate a mask matrix $D \in \mathbb{R}^{1 \times (n_h \times n_w \times n_h \times n_w)}$ with drop ratio $d$ to enforce some elements of the similarity matrix to be -inf. The formulation of $D$ as follows:

$$
d_j = \begin{cases} 
0 & \text{with probability } 1 - d \\
-\infty & \text{with probability } d
\end{cases}
$$

Finally, the attention weight matrix is computed by given masked similarity matrix as input. Formally, we compute the outputs of a patch as:

$$
a = \sum_{j=1}^{n_h \times n_w} p_j v_j
$$

$$
p_j = \text{exp}(d_j + \frac{k_j^2}{scale}) \frac{\text{exp}(d_j + \frac{k_j^2}{scale})}{\sum_{j=1}^{n_h \times n_w} \text{exp}(d_j + \frac{k_j^2}{scale})}
$$

where $q_j, k_j, v_j$ denote query of $i^{th}$ patch, key of $j^{th}$ patch, value of $j^{th}$ patch. It is not difficult to find that Eq. (13) and Eq. (14) is the same equivalent form of Eq. (1) and Eq. (2). In particular, as shown in Algorithm 1, we only need to add two lines of code to pure attention to implement DropKey. Obviously, compared with Dropout, DropKey only changes the position of introducing mask.

**Scheduled Drop Ratio** Vision transformer always consists of few self-attention blocks to gradually learn high-dimensional features. Generally, early layers operate low-level visual features and deeper layers aim to model spatially coarse but complex information. Hence, we attempt to set smaller drop ratio for deeper layers to avoid missing important objects-relevant information. Specifically, instead of stochastically dropping with some fixed probability in Dropout, we gradually decrease number of dropped keys over the layers during training stage. We find that this scheduled drop ratio not only work well for DropKey, but also significantly improves the performance of Dropout.
Algorithm 1 Attention with DropKey code

```python
# N: token number, D: token dim
# Q: query (N, D), K: key (N, D), V: value (N, D)
# use_DropKey: whether use DropKey
# mask_ratio: ratio to mask
# m_r = torch.ones_like(attn) * mask_ratio

def Attention(Q, K, V, use_DropKey, mask_ratio):
    """mask_ratio: ratio to mask
    use_DropKey: whether use DropKey
    Q: query (N, D), K: key (N, D), V: value (N, D)
    N: token number, D: token dim"

    if use_DropKey == True:
        # use DropKey as regularizer
        m_r = torch.ones_like(attn) * mask_ratio
        attn = attn.softmax(dim=-1)

    attn = Q * (Q.shape[1] ** -0.5) @ K.transpose(-2, -1)

    return x

    x = attn @ V
```

Structured DropKey Although structured drop has been detailedly explored in convolution network, there is no work to study the impact of structured drop on vision transformer. In this section, we implement two structured forms of DropKey which are named DropKey-Block and DropKey-Cross (see Figure 2). For DropKey-Block, inspired by DropBlock [10], we drop contiguous patches in a square-shape window of feature maps. Actually, we re-set drop ratio \( d_{\text{block}} \) for each patch as:

\[
d_{\text{block}} = \frac{d}{s^2 (n_h - s + 1)(n_w - s + 1)}
\]

where \( d \) denotes the probability of dropping a patch in DropKey. The valid seed region is \((n_h - s + 1)(n_w - s + 1)\) where \( s \) denotes the size of window and \( n_h, n_w \) denotes height, weight of feature map.

The cross-shape window attention is proved to achieve strong modeling capability [5] which also indicates the information in this structure is correlated. For DropKey-Cross, we discard features in a cross window to prevent information flow through self-attention. In our implementation, we drop the rows and columns of valid seed and re-set the drop ratio \( d_{\text{cross}} \) for each patch as:

\[
d_{\text{cross}} = \frac{d}{s(n_h + n_w - s)(n_h + n_w - s + 1)}
\]

Note that the formulations for block- and cross-version are only the approximation, because there will be some overlapped when perform drop.

Align Expectation As mentioned in Section 3.1, the misaligned expectations have a certain negative impact on the model, so we attempt to use two methods to align the expectation. The first one is to use Monte Carlo method to estimate \( c \). We perform multiple random drop and calculate the attention weight matrix after each drop operation. Finally, the average of calculated multiple weight matrices is applied as the input for the next step. For the second one, we take the inspiration from [9] and propose to finetune the model without DropKey, as an extra stage after DropKey training. We experimentally verify that the second strategy performs better with experiment in Section 4.1.1.

4. Experiments

We conduct experiments on three tasks, image classification, object detection, human-object interaction detection and human mesh recovery, to show the efficacy and generalizability of our DropKey for improving ViTs.

4.1. Image Classification

Datasets We conduct the following experiments with T2T [34], VOLO [35], CeiT [33] and DeiT [27] for image classification on CIFAR10 [13], CIFAR100 [15] and ImageNet [4]. a) We conduct the ablation study to demonstrate the effects of introducing modified DropAttention to three T2T and three VOLO backbone architectures on CIFAR10 and CIFAR100. b) We validate the modified DropAttention by training T2T and VOLO from scratch on ImageNet.

Training By default, models are trained with random initialization on all datasets. Due to the missing training setting for T2T [34] and VOLO [35], we conduct grid search on learning rate and epoch number to make the vanilla benchmarks achieve the best performance. All implementation specifics can be seen in Appendix. For ImageNet, our training recipe follows [34, 35]. Specifically, we set 50 epochs and keep learning rate as 10\(^{-5}\) for finetune stage for all datasets.

4.1.1 Ablations on CIFAR10 and CIFAR100

Does DropKey boost generalization? T2T [34] and VOLO [35] are the widely used vision transformer architecture for image recognition. However, the DropAttention applied in these architectures does not apply re-normalize operation. In the following experiments, we plug in the DropKey to the family of T2T/VOLO and compare results with the version that use Dropout. The results in Figure 3 indicate DropKey can consistently increase the performance of various architectures on CIFAR10 and CIFAR100. On CIFAR100 with T2T19, for example, DropKey gains 1.7% improvement for the adaptive coefficient yields further improvement. Additionally, we found that drop ratio is also an essential factor affecting performance. Take T2T14 with DropKey as an example, the accuracy initially increases with increasing drop ratio until it achieves its peak accuracy of 79.2% at drop ratio = 0.3 and it declines upon further increase of the drop ratio. These suggest that the introduction of DropKey would be beneficial to the network for avoiding suffering overfits. However, a larger drop ratio can lead to an inability to capture valid information about the object.
Similar to DropKey, the performance of Dropout is also related to drop ratio.

**How does the choice of scheduled strategy of drop ratio impact accuracy?** The ablation in Table 1 analyzes the accuracy of DropKey by adjusting the strategy of setting drop ratio. Specifically, we validate three strategy as follows: a) Constant: drop ratio is constant over self attention layers. b) Scheduled↑: drop ratio is linearly increased over self attention layers. c) Scheduled↓: drop ratio is linearly decreased over self attention layers. Firstly, we find that Scheduled↓ outperform Constant whether using re-normalize or not. Secondly, introducing drop ratio with Scheduled↑ would seriously affect the performance of the model and even perform worse than the pure vision transformer. This phenomenon is in line with our expectation, since deeper layers often contain high-level semantic information which is essential to perform classification. A large drop ratio in deep layer will increase the risk of losing important features which would make model difficult to converge to a generalized solution. Hence, we set drop ratio with Scheduled↓ in all following experiments.

**Is Dropkey helpful to capture global information?** The contribution of DropKey is to generate a smoother attention weight matrix to encourage models to focus on global features. Class token has been proved to be effective in aggregating the information of the whole image, so we measure the smoothness of the attention distribution by calculating the entropy of its attention weight vector. Specifically, we define the entropy of the attention vector of class token as follows $E_i = -\sum_j p_{ij} \log p_{ij}$ where $p_{ij}$ denotes the attention weight of $j^{th}$ attending patch in $i^{th}$ head.

Note that a small entropy denotes that this head focuses on sparse patches. For convenience, the mean of entropy of multi-head as $E$ is reported in Figure 4. Clearly, it reads that while both Dropout and DropKey are able to improve the value of entropy, the improvement by DropKey is more significant. Meanwhile, when the drop rate increases, the entropy increase accordingly. In addition, to further verify the necessity of the adaptive coefficient for capturing general global information, we visualize the attention map (Figure 5) from class token in the last transformer layer. Obviously, the model with DropKey smoothly assigns attention weight to the area related to the instance. Finally, we further verify that our method can alleviate local-bias via the fact that transformers with DropKey perform robustly in occluded scenarios.

**Is structured drop is useful to vision transformer?** In this section, we analyze whether structured masking is useful for vision transformer via DropKey-Block and DropKey-Cross. Specifically, we swept over window size from 3 to 5 for DropKey-Block and from 1 to 3 for DropKey-Cross. Compared with the DropKey, the structured drop degrade the performance of the model in all case and classification accuracy will decrease with the further increase of the window size. One possible reason for this phenomenon is that a larger drop ratio in shallow transformer layer results in losing vital information of object. However, a larger drop ratio is the key to avoid overfit to low-level feature. Therefore, we encourage set a larger drop ratio rather than the introduction of structured drop.

**How does align expectations impact accuracy?** As discussed above, the mismatched expectations problem arises when directly use trained model with DropKey to test. Herein, to address above issue, we explore the impact of two methods, Monte Carlo and finetune, on the performance of network. The results in Table 3 indicates that accuracy can be improved regardless of the expectation alignment operation. Additionally, the accuracy of Monte Carlo increases with increasing sampling iterations. However, huge sampling iterations will lead to unacceptable computational cost. To fairly compare with Dropout, we also finetune it with the same training hyper-parameters as DropKey and then note that finetune only brings insignificant performance improvement or even over-fitting. In conclusion, alignment expectation of training and inference stage indeed can further improve the accuracy of the model.

4.1.2 DropKey vs Dropout on ImageNet

We provide the accuracy under other different drop ratio for each backbone with DropKey on ImageNet, and contrast them with existing State of the Arts (SOTA) equipped with Dropout, including T2T, VOLO, CeiT and DeiT in Table 4. We have these findings: (1) Clearly, our DropKey obtains significant enhancement against pure backbone. Meanwhile, We note that Dropout can only bring a little boost
Table 1. Comparison of models with Dropout or DropKey on CIFAR10 and CIFAR100 when introducing different scheduled strategy of drop ratio. The drop ratio is set as 0.3 for all models.

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scheduled↑</td>
<td>Constant</td>
</tr>
<tr>
<td>T2T14 [34] + Dropout</td>
<td>96.7</td>
<td>96.9</td>
</tr>
<tr>
<td>T2T14 [34] + DropKey</td>
<td>97.0</td>
<td>97.4</td>
</tr>
<tr>
<td>T2T19 [34] + Dropout</td>
<td>96.6</td>
<td>97.1</td>
</tr>
<tr>
<td>T2T19 [34] + DropKey</td>
<td>97.1</td>
<td>97.2</td>
</tr>
<tr>
<td>T2T24 [34] + Dropout</td>
<td>96.4</td>
<td>97.0</td>
</tr>
<tr>
<td>T2T24 [34] + DropKey</td>
<td>96.6</td>
<td>97.2</td>
</tr>
<tr>
<td>VOLOd1 [35] + Dropout</td>
<td>95.8</td>
<td>96.4</td>
</tr>
<tr>
<td>VOLOd1 [35] + DropKey</td>
<td>96.2</td>
<td>96.7</td>
</tr>
<tr>
<td>VOLOd2 [35] + Dropout</td>
<td>95.6</td>
<td>96.1</td>
</tr>
<tr>
<td>VOLOd2 [35] + DropKey</td>
<td>95.7</td>
<td>96.6</td>
</tr>
<tr>
<td>VOLOd3 [35] + Dropout</td>
<td>96.0</td>
<td>96.3</td>
</tr>
<tr>
<td>VOLOd3 [35] + DropKey</td>
<td>96.2</td>
<td>96.4</td>
</tr>
</tbody>
</table>

Figure 4. The histogram of entropy on CIFAR100. dr denotes drop ratio.

Figure 5. Visualization on CIFAR100.

Table 2. Comparison of different structured drop on CIFAR100. s denotes window size and drop ratio is set as 0.3 for all models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Random</th>
<th>Block</th>
<th>Cross</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s=3</td>
<td>s=5</td>
<td>a=1</td>
</tr>
<tr>
<td>T2T14 [34] + Dropout</td>
<td>78.4</td>
<td>78.1</td>
<td>77.1</td>
</tr>
<tr>
<td>T2T14 [34] + DropKey</td>
<td>79.2</td>
<td>78.7</td>
<td>77.5</td>
</tr>
<tr>
<td>VOLOd1 [35] + Dropout</td>
<td>78.6</td>
<td>78.0</td>
<td>76.7</td>
</tr>
<tr>
<td>VOLOd1 [35] + DropKey</td>
<td>79.2</td>
<td>78.5</td>
<td>77.5</td>
</tr>
</tbody>
</table>

Table 3. Comparison of models w/ or w/o Align Expectations on CIFAR100. Pure denotes the model without align expectation and drop ratio is set as 0.3 for all models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pure</th>
<th>Monte Carlo</th>
<th>Finetune</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
<td>6000</td>
<td></td>
</tr>
<tr>
<td>T2T14 [34] + Dropout</td>
<td>78.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T2T14 [34] + DropKey</td>
<td>78.8</td>
<td>78.9</td>
<td>79.1</td>
</tr>
<tr>
<td>T2T19 [34] + Dropout</td>
<td>78.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T2T19 [34] + DropKey</td>
<td>79.3</td>
<td>79.5</td>
<td>80.2</td>
</tr>
</tbody>
</table>

Table 2. Comparison of different structured drop on CIFAR100. s denotes window size and drop ratio is set as 0.3 for all models.

4.2. Object Detection in COCO

In this section, we verify the generalizability and effectiveness of DropKey for object detection on COCO dataset [19]. We apply DETR [1] framework for the experiments. Specifically, DETR is composed of the backbone, encoder and decoder. We followed the model architecture, anchor definition and training recipe in [1] to build DETR+Dropout and DETR+DropKey. In Table 5, we report the results of DETR [1] with Dropout or DropKey in terms of AP. It can be seen that, our DropKey significantly outperforms existing Dropout. In detail, DropKey achieve +0.7, +1.1, +0.4, +0.3, +0.8 and +0.7 higher AP than the Dropout, which is regarded as a remarkable boost considering the challenge on this benchmark. Another observation is that large objects benefit more from DropKey than small objects. These results suggest that DropKey has the advantage over Dropout.

4.3. Human-Object Interaction Detection in HICO-DET

Since the task of scene graph and relation understanding is quite sensitive to global context learning, so we also verify the effectiveness of DropKey in QPIC [26] for Human-Object Interaction Detection in HICO-DET [2]. QPIC is a transformer-based feature extractor that can effectively aggregate contextually important information. We respec-

32106
Table 4. Comparisons on backbones with Dropout or DropKey on ImageNet. dr denotes drop ratio.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Model</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2T14 [34]</td>
<td>81.76 ± 0.05</td>
<td>VOLOd1 [35]</td>
<td>84.27 ± 0.12</td>
</tr>
<tr>
<td>T2T14 [34] + Dropout(dr=0.05)</td>
<td>81.65 ± 0.04</td>
<td>VOLOd1 [35] + Dropout(dr=0.05)</td>
<td>84.35 ± 0.11</td>
</tr>
<tr>
<td>T2T14 [34] + Dropout(dr=0.1)</td>
<td>81.53 ± 0.08</td>
<td>VOLOd1 [35] + Dropout(dr=0.1)</td>
<td>84.31 ± 0.04</td>
</tr>
<tr>
<td>T2T14 [34] + DropKey(dr=0.05)</td>
<td>82.04 ± 0.14</td>
<td>VOLOd1 [35] + DropKey(dr=0.05)</td>
<td>84.53 ± 0.03</td>
</tr>
<tr>
<td>T2T14 [34] + DropKey(dr=0.1)</td>
<td>81.93 ± 0.09</td>
<td>VOLOd1 [35] + DropKey(dr=0.1)</td>
<td>84.39 ± 0.02</td>
</tr>
<tr>
<td>T2T19 [34]</td>
<td>82.56 ± 0.09</td>
<td>VOLOd2 [35]</td>
<td>85.24 ± 0.05</td>
</tr>
<tr>
<td>T2T19[34] + Dropout(dr=0.05)</td>
<td>83.63 ± 0.05</td>
<td>VOLOd2 [35] + Dropout(dr=0.05)</td>
<td>85.22 ± 0.08</td>
</tr>
<tr>
<td>T2T19[34] + Dropout(dr=0.1)</td>
<td>82.68 ± 0.04</td>
<td>VOLOd2 [35] + Dropout(dr=0.1)</td>
<td>85.21 ± 0.03</td>
</tr>
<tr>
<td>T2T19[34] + DropKey(dr=0.05)</td>
<td>82.71 ± 0.02</td>
<td>VOLOd2 [35] + DropKey(dr=0.05)</td>
<td>85.33 ± 0.12</td>
</tr>
<tr>
<td>T2T19 [34] + DropKey(dr=0.1)</td>
<td>82.94 ± 0.04</td>
<td>VOLOd2 [35] + DropKey(dr=0.1)</td>
<td>85.38 ± 0.06</td>
</tr>
<tr>
<td>CeiT-T [33]</td>
<td>81.92 ± 0.06</td>
<td>DeiT-B [27]</td>
<td>81.94 ± 0.02</td>
</tr>
<tr>
<td>CeiT-T [33] + Dropout(dr=0.05)</td>
<td>81.90 ± 0.03</td>
<td>DeiT-B [27] + Dropout(dr=0.05)</td>
<td>81.91 ± 0.04</td>
</tr>
<tr>
<td>CeiT-T [33] + Dropout(dr=0.1)</td>
<td>81.83 ± 0.06</td>
<td>DeiT-B [27] + Dropout(dr=0.1)</td>
<td>81.87 ± 0.05</td>
</tr>
<tr>
<td>CeiT-T [33] + DropKey(dr=0.05)</td>
<td>82.14 ± 0.05</td>
<td>DeiT-B [27] + DropKey(dr=0.05)</td>
<td>82.23 ± 0.01</td>
</tr>
<tr>
<td>CeiT-T [33] + DropKey(dr=0.1)</td>
<td>82.27 ± 0.06</td>
<td>DeiT-B [27] + DropKey(dr=0.1)</td>
<td>82.25 ± 0.08</td>
</tr>
</tbody>
</table>

Table 5. Comparison of DETR with Dropout and DropKey on COCO validation set.

<table>
<thead>
<tr>
<th>Model</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>APs</th>
<th>APm</th>
<th>APl</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETR [1]</td>
<td>41.9</td>
<td>62.1</td>
<td>44.2</td>
<td>20.6</td>
<td>45.7</td>
<td>60.7</td>
</tr>
<tr>
<td>DETR [1] + Dropout</td>
<td>42.2</td>
<td>62.3</td>
<td>44.3</td>
<td>20.8</td>
<td>45.9</td>
<td>61.1</td>
</tr>
<tr>
<td>DETR [1] + DropKey</td>
<td>42.9</td>
<td>63.4</td>
<td>44.7</td>
<td>21.1</td>
<td>46.7</td>
<td>61.8</td>
</tr>
</tbody>
</table>

Table 6. Comparison of METRO with Dropout and DropKey on HUMBI.

<table>
<thead>
<tr>
<th>Model</th>
<th>mPVE↓</th>
<th>mPJPE↓</th>
<th>PA-mPJPE↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRO [18]</td>
<td>57.8</td>
<td>51.5</td>
<td>39.9</td>
</tr>
<tr>
<td>METRO [18] + Dropout</td>
<td>57.9</td>
<td>51.6</td>
<td>39.7</td>
</tr>
<tr>
<td>METRO [18] + DropKey</td>
<td>57.1</td>
<td>51.0</td>
<td>39.5</td>
</tr>
</tbody>
</table>

Table 7. Comparison of QPIC with Dropout and DropKey on HICO-DET.

<table>
<thead>
<tr>
<th>Model</th>
<th>full (Default)</th>
<th>rare (Default)</th>
<th>non-rare (Default)</th>
<th>full (Known object)</th>
<th>rare (Known object)</th>
<th>non-rare (Known object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPIC-ResNet101 [26] + Dropout (dr=0.1)</td>
<td>29.96</td>
<td>24.03</td>
<td>31.63</td>
<td>32.42</td>
<td>26.01</td>
<td>34.31</td>
</tr>
<tr>
<td>QPIC-ResNet101 [26] + Dropout (dr=0.3)</td>
<td>30.02</td>
<td>24.17</td>
<td>31.72</td>
<td>32.49</td>
<td>26.13</td>
<td>34.41</td>
</tr>
<tr>
<td>QPIC-ResNet101 [26] + DropKey (dr=0.3)</td>
<td>30.87</td>
<td>24.63</td>
<td>32.24</td>
<td>33.21</td>
<td>26.66</td>
<td>34.94</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper, we explore the drop unit, drop schedule and structured necessity of the dropout technique in ViT. Specifically, we propose to set Key as the drop unit, which yields a novel dropout-before-softmax scheme. We theoretically and experimentally verify that this scheme can regularize attention weights and meanwhile keep them as a probability distribution simultaneously, helping capture vital patterns in a global manner and overcome local-bias problems that occurred to vanilla dropout. In addition, we present a new decreasing schedule for drop ratio, which stabilizes the training phase by avoiding overfitting in low-level features and maintaining sufficient high-level features. Moreover, we also experimentally show that structured dropout is not necessary for ViT. We distill the above analysis as a novel DropKey method, which plays as an improved version of dropout for ViT. Comprehensive experiments with different architectures on various vision tasks demonstrate the effectiveness of the proposed DropKey for enhancing ViTs.

Acknowledgements

This paper is supported by the National key research and development program of China (2021YFA1000403), and the National Natural Science Foundation of China (Nos. U19B2040, 11991022).
References


[7] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 1, 2


[31] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu,


