UniDrop: A Simple yet Effective Technique to Improve Transformer without Extra Cost

Zhen Wu[1]* **Lijun Wu**[2] **Qi Meng**[2] **Yingce Xia**[2] **Shufang Xie**[2]
Tao Qin[2] **Xinyu Dai**[1] **Tie-Yan Liu**[2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University
[2]Microsoft Research Asia
`wuz@smail.nju.edu.cn, daixinyu@nju.edu.cn`
`{Lijun.Wu,meq,yingce.xia,shufxi,taoqin,tyliu}@microsoft.com`

Abstract

Transformer architecture achieves great success in abundant natural language processing tasks. The over-parameterization of the Transformer model has motivated plenty of works to alleviate its overfitting for superior performances. With some explorations, we find simple techniques such as dropout, can greatly boost model performance with a careful design. Therefore, in this paper, we integrate different dropout techniques into the training of Transformer models. Specifically, we propose an approach named `UniDrop` to unite three different dropout techniques from fine-grain to coarse-grain, i.e., feature dropout, structure dropout, and data dropout. Theoretically, we demonstrate that these three dropouts play different roles from regularization perspectives. Empirically, we conduct experiments on both neural machine translation and text classification benchmark datasets. Extensive results indicate that Transformer with `UniDrop` can achieve around 1.5 BLEU improvement on IWSLT14 translation tasks, and better accuracy for the classification even using strong pre-trained RoBERTa as backbone.

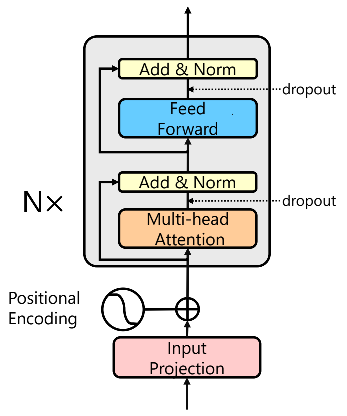
1 Introduction

In recent years, Transformer (Vaswani et al., 2017) has been the dominant structure in natural language processing (NLP), such as neural machine translation (Vaswani et al., 2017), language modeling (Dai et al., 2019) and text classification (Devlin et al., 2019; Liu et al., 2019). To further improve the model performance, there has been much effort in designing better architectures or introducing external knowledge into Transformer models (Wu et al., 2019; Lu et al., 2019; Kitaev et al., 2020; Ahmed et al., 2017; Hashemi et al., 2020), which increases computational costs or requires extra resources.

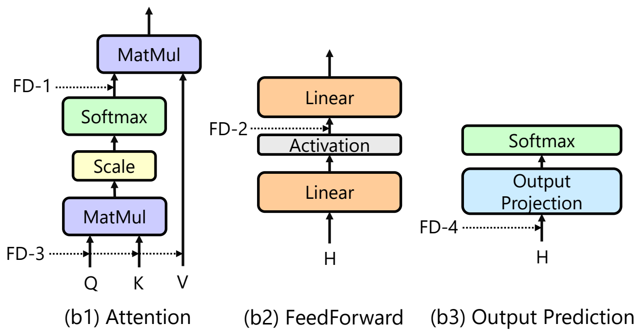
Despite the effectiveness of above strategies, the over-parameterization and overfitting is still a crucial problem for Transformer. Regularization methods such as weight decay (Krogh and Hertz, 1992), data augmentation (Sennrich et al., 2016a), dropout (Srivastava et al., 2014), parameter sharing (Dehghani et al., 2018; Xia et al., 2019) are all widely adopted to address overfitting. Among these regularization approaches, dropout (Srivastava et al., 2014), which randomly drops out some hidden units during training, is the most popular one and various dropout techniques have been proposed for Transformer. For example, Fan et al. (2020a) propose `LayerDrop`, a random structured dropout, to drop certain layers of Transformer during training. Zhou et al. (2020) alternatively propose `DropHead` as a structured dropout method for regularizing the multi-head attention mechanism. Both of them achieved promising performances. One great advantage of dropout is that it is free of additional computational costs and resource requirements. Hence we ask one question: *can we achieve stronger or even state-of-the-art (SOTA) results only relying on various dropout techniques instead of extra model architecture design or knowledge enhancement?*

To this end, in this paper, we propose `UniDrop` to integrate three different-level dropout techniques from fine-grain to coarse-grain, *feature dropout*, *structure dropout*, and *data dropout*, into Transformer models. Feature dropout is the conventional dropout (Srivastava et al., 2014) that we introduced before, which is widely applied on hidden representations of networks. Structure dropout is a coarse-grained control and aims to randomly drop some entire substructures or components from the whole model. In this work, we adopt the aforementioned `LayerDrop` (Fan et al., 2020a) as our structure dropout. Different from the previous two dropout methods, data dropout (Iyyer et al., 2015) is performed on the input data level, which serves as a data augmentation method by randomly dropping out some tokens in an input sequence.

* This work was done when Zhen Wu was a research intern at Microsoft Research Asia.



(a) Transformer architecture.



(b) Structure and overview of feature dropout.

Figure 1: Transformer structure and feature dropout applied in different Transformer components.

We first theoretically analyze different regularization roles played by the three dropout techniques, and we show they can improve the generalization ability from different aspects. Then, we provide empirical evaluations of the `UniDrop` approach. We conduct experiments on neural machine translation with 8 translation datasets, and text classification task with 8 benchmark datasets. On both sequence generation and classification tasks, experimental results show that the three dropouts in `UniDrop` can jointly improve the performance of Transformer.

The contributions of this paper can be summarized as follows:

- We introduce `UniDrop`, which unites three different dropout techniques into a robust one for Transformer, to jointly improve the performance of Transformer without additional computational cost and prior knowledge.
- We theoretically demonstrate that the three dropouts, i.e., feature dropout, structure dropout, and data dropout play different roles in preventing Transformer from overfitting and improving the robustness of the model.
- Extensive results indicate that Transformer models with `UniDrop` can achieve strong or even SOTA performances on sequence generation and classification tasks. Specifically, around 1.5 BLEU improvement on IWSLT14 translation tasks, and better accuracy for classification even using strong pre-trained model RoBERTa as backbone.

2 Background

Feature dropout (FD) and structure dropout (SD) are highly coupled with model architecture. Therefore, we briefly recap Transformer and refer the readers to Vaswani et al. (2017) for details.

As shown in Figure 1a, Transformer is stacked by several identical blocks, and each block contains two sub-layers, which are multi-head self-attention layer and position-wise fully connected feed-forward layer. Each sub-layer is followed by an `AddNorm` operation that is a residual connection `Add` (He et al., 2016) and a layer normalization `LN` (Ba et al., 2016).

Multi-head Attention sub-layer consists of multiple parallel attention heads, and each head maps the query $\mathbf{Q}$ and a set of key-value pairs $\mathbf{K}, \mathbf{V}$ to an output through a scale dot-product attention:

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V}, \qquad (1)$$

where d_k is the dimension of query and key, and $\frac{1}{\sqrt{d_k}}$ is a scaling factor. The outputs of these heads are then concatenated and projected again to result in the final values.

Position-wise Feed-Forward sub-layer applies two linear transformations with an inner ReLU (Nair and Hinton, 2010) activation:

$$\mathtt{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W_1} + \mathbf{b_1})\mathbf{W_2} + \mathbf{b_2}, \qquad (2)$$

where $\mathbf{W}$ and $\mathbf{b}$ are parameters.

The output of each sub-layer is then followed with `AddNorm`: $\mathtt{AddNorm}(\mathrm{x}) = \mathtt{LN}(\mathtt{Add}(\mathrm{x}))$.

3 UniDrop

In this section, we first introduce the details of the three different levels of dropout techniques we study, feature dropout, structure dropout and data dropout. Then we provide the theoretical analysis of these dropout methods on the regularization perspectives. Finally, we present our proposed `UniDrop` approach for training Transformer.

3.1 Feature Dropout

The feature dropout (FD), as a well-known regularization method, is proposed by Srivastava et al. (2014), which is to randomly suppress neurons of neural networks during training by setting them to 0 with a pre-defined probability p.

In practice, dropout is applied to the output of each sub-layer by default. Besides, Transformer also contains two specific feature dropouts for multi-head attention and activation layer of feed-forward network. In this work, we also explore their effects on the performance of Transformer.

- **FD-1** (attention dropout): according to Equation (1), we can obtain attention weight matrix $\mathbf{A} = \mathbf{Q}\mathbf{K}^\top$ towards value sequence $\mathbf{V}$. Our FD-1 is applied to the attention weight $\mathbf{A}$.

- **FD-2** (activation dropout): FD-2 is employed after the activation function between the two linear transformations of `FFN` sub-layer.

In addition to the above FDs for Transformer, we still find the risk of overfitting in pre-experiments. Therefore, we further introduce another two feature dropouts into the model architecture:

- **FD-3** (query, key, value dropout): FD-1 is used to improve generalization of multi-head attention. However, it is directly applied to the attention weights $\mathbf{A}$, where drop value $\mathbf{A}(i, j)$ means ignore the relation between token i and token j, thus a larger FD-1 means a larger risk of losing some critical information from sequence positions. To alleviate this potential risk, we add dropout to query, key, and value before the calculation of attention.

- **FD-4** (output dropout): we also apply dropout to the output features before linear transformation for softmax classification. Specifically, when dealing with sequence-to-sequence tasks such as machine translation, we add FD-4 to the output features of the last layer in the Transformer decoder, otherwise the last layer of the Transformer encoder.

The positions of each feature dropout applied in Transformer[1] are shown in Figure 1b.

3.2 Structure Dropout

There are three structure dropouts, respectively `LayerDrop` (Fan et al., 2020a), `DropHead` (Zhou et al., 2020) and `HeadMask` (Sun et al., 2020), which are specifically designed for Transformer.

Some recent studies (Voita et al., 2019; Michel et al., 2019) show multi-head attention mechanism is dominated by a small portion of attention heads. To prevent domination and excessive co-adaptation between different attention heads, Zhou et al. (2020) and Sun et al. (2020) respectively propose structured `DropHead` and `HeadMask` that drop certain entire heads during training. In contrast, `LayerDrop` (Fan et al., 2020a) is a higher-level and coarser-grained structure dropout. It drops some entire layers at training time and directly reduces the Transformer model size.

In this work, we adopt `LayerDrop` as the structure dropout to incorporate it into our `UniDrop`.

3.3 Data Dropout

Data dropout aims to randomly remove some words in the sentence with a pre-defined probability. It is often used as a data augmentation technique (Wei and Zou, 2019; Xie et al., 2020). However, directly applying vanilla data dropout is hard to keep the original sequence for training, which leads to the risk of losing high-quality training samples. To address this issue, we propose a *two-stage data dropout strategy*. Specifically, given a sequence, with probability p_k (a hyperparameter lies in $(0, 1)$), we keep the original sequence and do not apply data dropout. If data dropout is applied, for each token, with another probability p (another hyperparameter lies in $(0, 1)$), we will drop the token.

3.4 Theoretical Analysis

In this section, we provide theoretical analysis for feature dropout, structure dropout and data dropout, to show their different regularization effects. We first re-formulate the three dropout methods. For some probability p and layer representation $h \in \mathbb{R}^d$ (i.e., h is the vector of outputs of some layer), we

[1] We also explored other positions for feature dropout, but their performances are not so good (see Appendix A.3).

randomly sample a scaling vector $\xi \in \mathbb{R}^d$ with each independent coordinate as follows:

$$\xi_i = \begin{cases} -1 & \textit{with probability } p \\ \dfrac{p}{1-p} & \textit{with probability 1-p.} \end{cases} \quad (3)$$

Here, i indexes a coordinate of ξ, $i \in [1, ..., d]$. Then feature dropout can be applied by computing

$$h_{fd} = (\mathbf{1} + \xi) \odot h,$$

where $\odot$ denotes element-wised product and $\mathbf{1} = (1, 1, \cdots, 1)'$.

We use $F(h_{fd}(x))$ to denote the output of a model after dropping feature from a hidden layer and $\mathcal{L}$ to denote the loss function. Similar to Wei et al. (2020), we apply Taylor expansion to $\mathcal{L}$ and take expectation to ξ:

$$\begin{aligned} &\mathbb{E}_\xi \mathcal{L}(F(h_{fd}(x))) = \mathbb{E}_\xi \mathcal{L}(F((\mathbf{1}+\xi) \odot h(x))) \\ &\approx \mathcal{L}(F(h(x)) + \frac{1}{2}\mathbb{E}_\xi(\xi \odot h(x))^T D_h^2 \mathcal{L}(x)(\xi \odot h(x)) \\ &= \mathcal{L}(F(h(x)) + \frac{p}{2(1-p)} \sum_{j=1}^{d} D^2_{h_j,h_j}\mathcal{L}(x) \cdot h_j(x)^2, \quad (4) \end{aligned}$$

where $D_h^2\mathcal{L}$ is the Hessian matrix of loss with respect to hidden output h and $D^2_{h_j,h_j}\mathcal{L}(x)$ is the j-th diagonal element of $D_h^2\mathcal{L}$. Expect the original loss $\mathcal{L}(F(h(x)))$, the above formula shows that feature dropout implicitly regularize the term $\sum_{j=1}^{d} D^2_{h_j,h_j}\mathcal{L}(x) \cdot h_j(x)^2$, which relates to the trace of the Hessian.

For structure dropout, we use a 1-dim random scalar $\eta \in \mathbb{R}$ whose distribution is: $\eta = -1$ with probability p, and $\eta = 0$ with probability $1-p$. The structure dropout is similarly applied by computing $h_{sd} = (1+\eta) \cdot h$.

For input data $x \in \mathbb{R}^m$, here x is a sequence of tokens and m is the sequence length, we sample a random scaling vector $\beta \in \mathbb{R}^m$ with independent random coordinates where each coordinate is identically distributed as η. The input data after drop data becomes $x_{dd} = (\mathbf{1} + \beta) \odot x$.

Similar to feature dropout, we can obtain that data dropout implicitly optimizes the regularized loss as follows: $\mathcal{L}(F(h(x))) - p \cdot x^T \nabla_x \mathcal{L}(x) + p \cdot \sum_{j=1}^{m} D^2_{x_j,x_j}\mathcal{L}(x) \cdot x_j^2$, and structure dropout implicitly optimizes the regularized loss: $\mathcal{L}(F(h(x))) - p \cdot h(x)^T \nabla_h \mathcal{L}(x) + p \cdot \sum_{i,j=1}^{m} D^2_{h_i,h_j}\mathcal{L}(x) \cdot h_i(x)h_j(x)$, where $D^2_{h_i,h_j}\mathcal{L}(x)$ is the (i,j)-th element in Hessian matrix $D_h^2\mathcal{L}$.

Interpretation From the above analysis, we can conclude that feature dropout, structure dropout and data dropout regularize different terms of the model, and they can not be replaced by each other. (1) Because the hidden output will be normalized by layer normalization, the term $h(x)^T \nabla_h \mathcal{L}(x)$ equals to zero according to Lemma 2.4 in Arora et al. (2019). Therefore, structure dropout implicitly regularizes the term $\sum_{i,j=1}^{m} D^2_{h_i,h_j}\mathcal{L}(x)$. Hence, structure dropout can regularize the whole elements of Hessian of the model with respect to hidden output, while feature dropout only regularizes the diagonal elements of the Hessian. Thus, integrating structure dropout and feature dropout can regularize every component of Hessian with emphasizing the diagonal elements of the Hessian. (2) Since x is also normalized, the term $x^T \nabla_x \mathcal{L}(x)$ equals to zero according to Lemma 2.4 in Arora et al. (2019). Different from feature dropout and structure dropout, data dropout regularizes Hessian of loss with respect to input data.

Regularizing Hessian matrix with respect to both input and hidden output can improve model robustness and hence the generalization ability. We put more details in Appendix A.1.

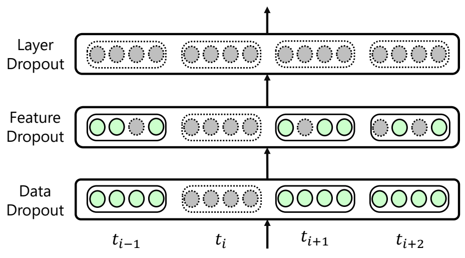


Figure 2: Different dropout components in `UniDrop`. The gray positions denote applying the corresponding dropout.

3.5 UniDrop Integration

From the above theoretical analysis, the three dropout techniques are performed in different ways to regularize the training of Transformer, each with unique property to improve the model generalization. Therefore, we introduce `UniDrop` to take the most of each dropout into Transformer. The overview of `UniDrop` is presented in Figure 2.

To better view each dropout in a model forward pass, we only show a three layers of architecture in Figure 2, and each layer with one specific dropout technique. The data dropout is applied in the input layer by dropping out some word embeddings (e.g., embedding of word t_i is dropped). In the middle layer, the feature dropout randomly drops several neurons in each word representations (e.g., the third neurons of word t_{i-1} is dropped). The last layer is directly dropped out through layer dropout[2].

4 Experiments

We conduct experiments on both sequence generation and classification tasks, specifically, neural machine translation and text classification, to validate the effectiveness of `UniDrop` for Transformer.

4.1 Neural Machine Translation

In this section, we introduce the detailed settings for the neural machine translation tasks and report the experimental results.

4.1.1 Datasets

We adopt the widely acknowledged IWSLT14 datasets[3] with multiple language pairs, including English↔German (En↔De), English↔Romanian (En↔Ro), English↔Dutch (En↔Nl), and English↔Portuguese-Brazil (En↔Pt-br), a total number of 8 translation tasks. Each dataset contains about 170k∼190k translation data pairs. The datasets are processed by Moses toolkit[4] and byte-pair-encoding (BPE) (Sennrich et al., 2016b) is applied to obtain subword units. The detailed statistics of datasets are shown in Appendix A.2.

4.1.2 Model

We use the `transformer_iwslt_de_en` configuration[5] for all Transformer models. Specifically, the encoder and decoder both consist of 6 blocks. The source and target word embeddings are shared for each language pair. The dimensions of embedding and feed-forward sub-layer are respectively set to 512 and 1024, the number of attention heads is 4. The default dropout (not our four feature dropout) rate is 0.3 and weight decay is 0.0001. All models are optimized with Adam (Kingma and Ba, 2015) and the learning rate schedule is same as in Vaswani et al. (2017). The weight of label smoothing (Pereyra et al., 2017) is set to 0.1.

For the Transformer models with our `UniDrop`, we set all feature dropout rates to 0.1. The structure dropout `LayerDrop` is only applied to the decoder with rate 0.1. For the data dropout, the sequence keep rate p_k and token dropout rate p are respectively 0.5 and 0.2. The other settings are the same as the configuration of the baseline Transformer.

To evaluate the model performance, we use beam search (Sutskever et al., 2014) algorithm to generate the translation results. The beam width is 5 and the length penalty is 1.0. The evaluation metric is the tokenized BLEU (Papineni et al., 2002) score with `multi-bleu.perl` script[6]. We repeat each experiment three times with different seeds and report the average BLEU.

4.1.3 Results

Table 1 shows the BLEU results of the Transformer baselines and models with different dropouts. Compared with baselines, we can see that the dropouts FD, SD, or DD all bring some improvements[7]. This observation verifies the existence of overfitting in the Transformer. In contrast, our model Transformer+`UniDrop` achieves the most improvements across all translation tasks, which demonstrates the effectiveness of `UniDrop` for the Transformer architecture. To further explore the effects of the three different grained dropouts in `UniDrop`, we conduct ablation studies and respectively remove the FD, SD, and DD from Transformer+`UniDrop`. The results in Table 1 show that three ablated models obtain lower BLEU scores compared to the full model. This observation validates the necessity of them for `UniDrop`. Among all ablation versions, the Transformer-`UniDrop` w/o FD obtains the least improvements. It is reasonable because FD actually contains four feature dropouts on different positions, which can effectively prevent Transformer from overfitting.

To show the superiority of `UniDrop`, we also compare the Transformer+`UniDrop` with several existing works on the widely acknowledged benchmark IWSLT14 De→En translation. These works improve machine translation from different aspects, such as the training algorithm design (Wang et al., 2019b), model architecture design (Lu et al., 2019; Wu et al., 2019) and data augmentation (Gao et al., 2019). The detailed results are shown in Table 2. We can see that the Transformer model with our `UniDrop` outperforms all previous works and achieve state-of-the-art performance, with 36.88

[2]Except the data dropout is only applied in the input layer, feature/structure dropout can be applied in each layer.

[3]`https://wit3.fbk.eu/mt.php?release=2014-01`

[4]`https://github.com/moses-smt/mosesdecoder/tree/master/scripts`

[5]`https://github.com/pytorch/fairseq`

[6]`https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

[7]The dropout rates of model Transformer+FD, Transformer+SD, Transformer+DD are tuned with IWSLT14 De→En dev set and respectively set to 0.2, 0.2, 0.3.

	En→De	De→En	En→Ro	Ro→En	En→Nl	Nl→En	Nn→Pt-br	Pt-br→En	Avg.	△
Transformer	28.67	34.84	24.74	32.14	29.64	33.28	39.08	43.63	33.25	-
+FD	29.61	36.08	25.45	33.12	30.37	34.50	40.10	44.74	34.24	+0.99
+SD	29.03	35.09	25.03	32.69	29.97	33.94	39.78	44.02	33.69	+0.44
+DD	28.83	35.26	24.98	32.76	29.72	34.00	39.50	43.71	33.59	+0.34
+`UniDrop`	**29.99**	**36.88**	**25.77**	**33.49**	**31.01**	**34.80**	**40.62**	**45.62**	**34.77**	+1.52
w/o FD	29.24	35.68	25.18	33.17	30.16	33.90	39.97	44.81	34.01	+0.76
w/o SD	29.92	36.70	25.59	33.26	30.55	34.75	40.45	45.60	34.60	+1.35
w/o DD	29.76	36.38	25.44	33.26	30.86	34.55	40.37	45.27	34.49	+1.24

Table 1: Machine translation results of the standard Transformer and our models on various IWSLT14 translation datasets. The "+FD", "+SD", "+DD", and "+`UniDrop`" denotes applying the feature dropout, structure dropout, data dropout, or `UniDrop` to the standard Transformer. The "w/o FD", "w/o SD" and "w/o DD" respectively indicate the removal of the feature dropout, structure dropout, or data dropout from the model Transformer+`UniDrop`. Avg. and △ denote the average results of the 8 translation tasks and improvements compared with the standard Transformer. Best results are in bold.

Approaches	BLEU
Adversarial MLE (Wang et al., 2019b)	35.18
DynamicConv (Wu et al., 2019)	35.20
Macaron (Lu et al., 2019)	35.40
IOT (Zhu et al., 2021)	35.62
Soft Contextual Data Aug (Gao et al., 2019)	35.78
BERT-fused NMT (Zhu et al., 2020)	36.11
MAT (Fan et al., 2020b)	36.22
MixReps+co-teaching (Wu et al., 2020)	36.41
Transformer	34.84
+`UniDrop`	**36.88**

Table 2: Comparison with existing works on IWSLT-2014 De→En translation task.

Approaches	En→De	Ro→En	Nl→En
MAT (Fan et al., 2020b)	29.90	-	-
MixReps+co-teaching (Wu et al., 2020)	29.93	33.12	34.45
Transformer	28.67	32.14	33.38
+`UniDrop`	**29.99**	**33.49**	**34.80**

Table 3: Comparison with existing works on IWSLT-2014 En→De, Ro→En, and Nl→En translation tasks.

BLEU score. Especially, it surpasses the BERT-fused NMT model (Zhu et al., 2020), which incorporates the pre-trained language model BERT, by a non-trivial margin. We also show some comparisons on IWSLT14 En→De, Ro→En, and Nl→En translations, the results are shown in Table 3.

According to the above results, `UniDrop` successfully unites the FD, SD, and DD, and finally improves the performance of Transformer on neural machine translation tasks, without any additional computation costs and resource requirements.

4.2 Text Classification

We also conduct experiments on text classification tasks to further demonstrate the effectiveness of `UniDrop` for the Transformer models.

4.2.1 Datasets

We evaluate different methods on the text classification task based on 8 widely-studied datasets, which can be divided into two groups. The first group is from GLUE tasks (Wang et al., 2019a), and they are usually used to evaluate the performance of the large-scale pre-trained language models after fine-tuning. The second group is some typical text classification datasets that are widely used in previous works (Voorhees and Tice, 1999; Maas et al., 2011; Zhang et al., 2015). The statistics of all datasets are shown in Appendix A.2.

4.2.2 Model

We employ RoBERTa$_{\mathrm{BASE}}$ (Liu et al., 2019) as the strong baseline and fine-tune it on the text classification datasets. Different from BERT$_{\mathrm{BASE}}$ (Devlin et al., 2019), RoBERTa$_{\mathrm{BASE}}$ is pre-trained with dynamic masking, full-sentences without NSP loss and a larger mini-batches. It has 12 blocks, and the dimensions of embedding and `FFN` are 768 and 3072, the number of attention heads is 12. When fine-tuning, we set the batch size to 32 and the max epoch to 30. Adam is applied to optimize the models with a learning rate of 1e-5 and a warm-up step ratio of 0.1. We employ the polynomial decay strategy to adjust the learning rate. The default dropout and weight decay are both set to 0.1.

When adding `UniDrop` to RoBERTa$_{\mathrm{BASE}}$, we empirically set feature dropout rate and `LayerDrop` rate to 0.1. For data dropout, the sequence keep rate p_k and token dropout rate p are respectively 0.5 and 0.1. The other settings are the same as in the baseline RoBERTa$_{\mathrm{BASE}}$. We use the standard accuracy to evaluate different methods on text classification tasks.

	MNLI	QNLI	SST-2	MRPC
BiLSTM+Attn, CoVe	67.9	72.5	89.2	72.8
BiLSTM+Attn, ELMo	72.4	75.2	91.5	71.1
BERT$_{\mathrm{BASE}}$	84.4	88.4	92.9	86.7
BERT$_{\mathrm{LARGE}}$	86.6	92.3	93.2	88.0
RoBERTa$_{\mathrm{BASE}}$	87.1	92.7	94.7	89.0
+UniDrop	**87.8**	**93.2**	**95.5**	**90.4**

Table 4: Accuracy on GLUE tasks (dev set). The models BiLSTM+Attn, CoVe and BiLSTM+Attn, ELMo are from Wang et al. (2019a). Best results are in bold.

	IMDB	Yelp	AG	TREC
Char-level CNN	-	62.05	90.49	-
VDCNN	-	64.72	91.33	-
DPCNN	-	69.42	93.13	-
ULMFiT	95.40	-	94.99	96.40
BERT$_{\mathrm{BASE}}$	94.60	69.94	94.75	97.20
RoBERTa$_{\mathrm{BASE}}$	95.7	70.9	95.1	97.6
+UniDrop	**96.0**	**71.4**	**95.5**	**98.0**

Table 5: Accuracy on the typical text classification datasets. Char-level CNN and VDCNN are from Zhang et al. (2015) and Conneau et al. (2017), DPCNN and ULMFiT are from Johnson and Zhang (2017) and Howard and Ruder (2018). Best results are in bold.

4.2.3 Results

Table 4 and Table 5 respectively show the accuracy of different models on GLUE tasks and typical text classification datasets.

Compared with the conventional BiLSTM and CNN based models, we can observe the pre-trained models, including ULMFiT, BERT, RoBERTa, achieve obvious improvements on most datasets. Benefiting from better training strategy, RoBERTa$_{\mathrm{BASE}}$ outperforms BERT$_{\mathrm{BASE}}$ and even BERT$_{\mathrm{LARGE}}$ on GLUE tasks.

We can see our proposed UniDrop further improve the performance RoBERTa$_{\mathrm{BASE}}$ on both small-scale and large-scale datasets. Specifically, UniDrop brings about 0.4 improvements of accuracy on the typical text classification datasets from Table 5. In contrast, RoBERTa$_{\mathrm{BASE}}$+UniDrop achieves more improvements on GLUE tasks. The experimental results on the 8 text classification benchmark datasets consistently demonstrate the facilitation of UniDrop for Transformer. We show more results and ablation study on text classification task in Appendix A.5.

5 Analysis

In this section, we use IWSLT14 De→En translation as the analysis task to investigate the capability of UniDrop to avoid overfitting, as well as the effects of different dropout components and dropout rates on UniDrop.

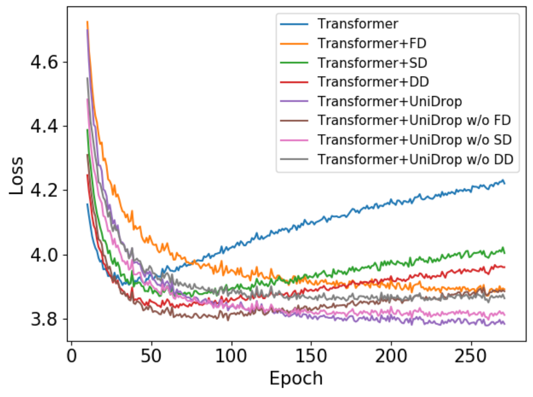


Figure 3: The dev loss of different models on IWSLT14 De→En translation task.

5.1 Overfitting

To show the superiority of UniDrop to prevent Transformer from overfitting, we compare the dev loss during training of Transformer, Transformer with each dropout technique, Transformer+UniDrop, and ablated models of Transformer+UniDrop. Figure 3 shows loss curves of different models.

We can observe that the standard Transformer is quickly overfitted during training, though it is equipped with a default dropout. In contrast, the feature dropout, structure dropout, and data dropout, as well as the combinations of any two dropouts (i.e., ablated models), greatly reduce the risk of overfitting to some extent. Among all compared models, our Transformer+UniDrop achieves the lowest dev loss and shows great advantage to prevent Transformer from overfitting. Besides, we also find that the dev loss of Transformer+UniDrop continuously falls until the end of the training. We stop it to keep training epochs of all models same for a fair comparison.

In Appendix A.4, we also plot the curves of training loss for the above models, together with the dev loss, to make a better understanding of the regularization effects from these dropout techniques.

5.2 Ablation Study

In Table 1, we have presented some important ablation studies by removing FD, SD, or DD from UniDrop. The consistent decline of BLEU scores demonstrates their effectiveness. Besides, we further investigate the effects of the two existing feature dropouts FD-1, FD-2, two new feature dropouts FD-3, FD-4, and our proposed two-stage

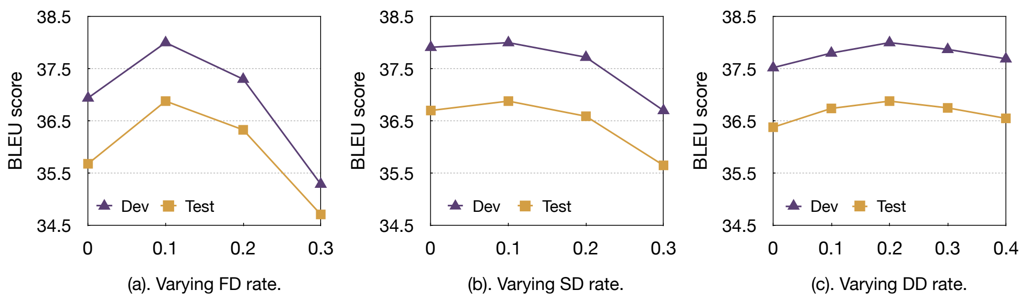


Figure 4: The BLEU scores of Transformer+`UniDrop` on IWSLT14 De→En translation dev set and test test, with varying the rates of FD, SD and DD respectively.

	De→En	En→De	Ro→En
Transformer	34.84	28.67	32.14
+`UniDrop`	36.88	29.99	33.49
w/o FD-1	36.72	29.84	33.33
w/o FD-2	36.57	29.76	33.28
w/o FD-3	36.59	29.83	33.31
w/o FD-4	36.65	29.59	33.24
w/o 2-stage DD	36.61	29.78	33.12

Table 6: Ablation study of data dropout and different feature dropouts on IWSLT14 De→En, En→De, and Ro→En translation tasks.

data dropout strategy on Transformer models. The experimental results are shown in Table 6.

From Table 6, we can see the four ablation models removing FDs underperform the full model Transformer+`UniDrop`, which means they can work together to prevent Transformer from overfitting. In multi-head attention module, FD-3 brings more BLUE improvement than FD-1. This comparison shows the insufficiency of only applying FD-1 for the Transformer architecture. The Transformer+`UniDrop` w/o 2-stage DD means we directly apply conventional data dropout to the sequence instead of our proposed 2-stage strategy. Compared with the full model, its performance also decreases. This shows the necessity of keeping the original sequence for data dropout.

5.3 Effects of Different Dropout Rates

To investigate the effects of FD, SD, and DD dropout rates on the `UniDrop`, we respectively vary them based on the setting (FD=0.1, SD=0.1, DD=0.2). When varying one dropout component, we keep other dropout rates unchanged. Figure 4 shows the corresponding results.

We can observe that the performance of each dropout for Transformer+`UniDrop` first increases then decreases when varying the dropout rates from small to large. Especially, varying the rate for FD dropout makes a more significant impact on the model performance since FD contains four feature dropout positions. In contrast, the DD is least sensitive to the dropout rate change, but it still plays a role in the model regularization.

6 Related Work

6.1 Dropout

Dropout is a popular regularization method for neural networks by randomly dropping some neurons during training (Srivastava et al., 2014). Following the idea, there are abundant subsequent works designing specific dropout for specific architecture, such as StochasticDepth (Huang et al., 2016), DropPath (Larsson et al., 2017), DropBlock (Ghiasi et al., 2018) for convolutional neural networks, Variational Dropout (Gal and Ghahramani, 2016), ZoneOut (Krueger et al., 2017), and Word Embedding Dropout (Gal and Ghahramani, 2016) for recurrent neural networks. Recently, the Transformer architecture achieves great success in a variety of tasks. To improve generalization of Transformer, some recent works propose `LayerDrop` (Fan et al., 2020a), `DropHead` (Zhou et al., 2020) and `HeadMask` (Sun et al., 2020) as structured regularizations, and obtain better performance than standard Transformer. Instead of designing a specific dropout for Transformer, in this work, we focus on integrating the existing dropouts into one `UniDrop` to further improve generalization of Transformer without any additional cost.

6.2 Data Augmentation

Data augmentation aims at creating realistic-looking training data by applying a transformation to a sample, without changing its label (Xie et al., 2020). In NLP tasks, data augmentation often refers to back-translation (Sennrich et al., 2016a), word replacing/inserting/swapping/dropout (Wei and Zou, 2019; Xie et al., 2020), etc. In this work, we adopt simple but effective word dropout as data level dropout in our `UniDrop`. We, additionally, design a two-stage data dropout strategy.

7 Conclusion

In this paper, we present an integrated dropout approach, `UniDrop`, to specifically regularize the Transformer architecture. The proposed `UniDrop` unites three different level dropout techniques from fine-grain to coarse-grain, feature dropout, structure dropout, and data dropout respectively. We provide a theoretical justification that the three dropouts play different roles in regularizing Transformer. Extensive results on neural machine translation and text classification datasets show that our Transformer+`UniDrop` outperforms the standard Transformer and various ablation versions. Further analysis also validates the effectiveness of different dropout components and our two-stage data dropout strategy. In conclusion, the `UniDrop` improves the performance and generalization of the Transformer without additional computational cost and resource requirement.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. Xinyu Dai and Lijun Wu are the corresponding authors. This work was partially supported by the NSFC (No. 61976114,61936012) and National Key R&D Program of China (No. 2018YFB1005102).

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*.

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. 2019. Theoretical analysis of auto rate-tuning by batch normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Angela Fan, Edouard Grave, and Armand Joulin. 2020a. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Yang Fan, Shufang Xie, Yingce Xia, Lijun Wu, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2020b. Multi-branch attentive transformer. *CoRR*, abs/2006.10270.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1019–1027.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5539–5544.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. 2018. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 10750–10760.

Helia Hashemi, Hamed Zamani, and W Bruce Croft. 2020. Guided transformer: Leveraging multiple external sources for representation learning in conversational search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and*

Development in Information Retrieval, pages 1131–1140.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Judy Hoffman, Daniel A Roberts, and Sho Yaida. 2019. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 646–661. Springer.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691.

Daniel Jakubovitz and Raja Giryes. 2018. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 562–570.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Anders Krogh and John A Hertz. 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron C. Courville, and Christopher J. Pal. 2017. Zoneout: Regularizing rnns by randomly preserving hidden activations. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. Fractalnet: Ultra-deep neural networks without residuals. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 14014–14024.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Zewei Sun, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2020. Alleviating the inequality of attention heads for neural machine translation. *CoRR*, abs/2009.09672.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5797–5808. Association for Computational Linguistics.

Ellen M. Voorhees and Dawn M. Tice. 1999. The TREC-8 question answering track evaluation. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*, volume 500-246 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dilin Wang, ChengYue Gong, and Qiang Liu. 2019b. Improving neural language modeling via adversarial training. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6555–6565. PMLR.

Colin Wei, Sham Kakade, and Tengyu Ma. 2020. The implicit and explicit regularization effects of dropout. *arXiv preprint arXiv:2002.12915*.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6381–6387.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Lijun Wu, Shufang Xie, Yingce Xia, Yang Fan, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2020. Sequence generation with mixed representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*.

Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5466–5473.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 649–657.

Wangchunshu Zhou, Tao Ge, Furu Wei, Ming Zhou, and Ke Xu. 2020. Scheduled drophead: A regularization method for transformer models. In *Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 1971–1980.

Jinhua Zhu, Lijun Wu, Yingce Xia, Shufang Xie, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2021. {IOT}: Instance-wise layer reordering for transformer structures. In *International Conference on Learning Representations*.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Appendix

A.1 Supplementary materials for theoretical analysis

In this section, we explain why regularizing Hessian matrix with respect to input or hidden output can improve model robustness and generalization.

We use $D_f^{\alpha}\mathcal{L}$ to denote the α-order derivatives of loss $\mathcal{L}$ with respect to f. If the hidden output is perturbed by ϵ, i.e., $\tilde{h} = h + \epsilon$, the k-th output F_k shifts to

$$F_k(h+\epsilon) = F_k(h) + \epsilon^T J_{F_k,h} + \frac{1}{2}\epsilon^T (D_h^2 F_k(h))\epsilon + o(\epsilon^2), \quad (5)$$

where $J_{F_k,h}(x)$ is the Jacobian between hidden output h and final output F_k.

Structure dropout regularizes all elements in Hessian matrix $D_h^2\mathcal{L}$. For Hessian matrix of loss function, we have $D_h^2\mathcal{L} = J_{F,h}^T(D_F^2\mathcal{L})J_{F,h} + \sum_k (D_{F_k}\mathcal{L})(D_h^2 F_k(h))$. Thus, regularizing all elements in $D_h^2\mathcal{L}$ means regularizing both $J_{F,h}$ and $D_h^2 F_k(h)$. As shown in Eq.5, regularizing this two terms can make $|F_k(h+\epsilon) - F_k(h)|$ smaller. Therefore, the robustness of the model is improved and the generalization ability of the model can also be improved (Hoffman et al., 2019; Jakubovitz and Giryes, 2018).

Feature dropout regularizes diagonal element of $D_h^2\mathcal{L}$. Using the approximation $D_h^2\mathcal{L} \approx J_{F,h}^T(D_F^2\mathcal{L})J_{F,h}$(Wei et al., 2020), regularizing diagonal elements $D_h^2\mathcal{L}$ equals to regularizing norm of Jacobian, i.e., $||J_{F,h}||_2$ if $D_F^2\mathcal{L}$ is roughly a diagonal matrix. For cross-entropy loss, $D_F^2\mathcal{L} = diag(z) - zz^T$, where z is the probability vector predicted by the model encoding the distribution over output class labels, the matrix $D_F^2\mathcal{L}$ can be approximated by a diagonal matrix. Thus, feature dropout mainly regularizes the first-order coefficient $J_{F_k,h}$ in Taylor expansion in Eq.5, which is different from structure dropout. Since Jacobian is an essential quantity for the generalization (Wei et al., 2020; Hoffman et al., 2019), emphasising this term is necessary for generalization although structure dropout can also regularize it.

Similar analysis can be applied to data dropout and we only need to replace hidden output h to the input x.

A.2 Statistics of Datasets

Table 7 and Table 8 respectively show the statistics of machine translation and text classification benchmark datasets we used to evaluate the `UniDrop` for Transformer.

Datasets	Train	Dev	Test
En↔De	160k	7k	7k
En↔Ro	180k	4.7k	1.1k
En↔Nl	170k	4.5k	1.1k
En↔Pt-br	175k	4.5k	1.2k

Table 7: Statistics for machine translation datasets.

Datasets	Classes	Train	Dev
MNLI	3	393k	20k
QNLI	2	105k	5.5k
SST-2	2	67k	0.9k
MRPC	2	3.7k	0.4k
Datasets	Classes	Train	Test
IMDB	2	25k	25k
Yelp	5	650k	50k
AG's News	4	120k	76k
TREC	6	5.4k	0.5k

Table 8: Statistics for text classification datasets.

For machine translation tasks, the four language pairs all contain around 170k∼190k training pairs.

Text classification experiments are conducted in GLUE tasks (Wang et al., 2019a) and typical text classification benchmarks datasets (Voorhees and Tice, 1999; Maas et al., 2011; Zhang et al., 2015). For GLUE tasks, we adopt the four datasets MNLI, QNLI, SST-2 and MRPC. They are used to evaluate the ability of models on language inference, sentiment classification and paraphrase detection. In typical text classification datasets, IMDB is binary film review classification task (Maas et al., 2011). Yelp and AG's News datasets are built by (Zhang et al., 2015), respectively for sentiment classification and topic classification. TREC is a question classification dataset consisting of 6 question types (Voorhees and Tice, 1999).

A.3 Dropout Attempts

Besides the different dropout methods introduced in Section 3, we also tried some other dropouts. We first introduce their settings. The 'QKV_proj' applies dropout to query, key, and value after linear projection. In contrast, FD-3 is to add dropout to query, key, and value before projection. Similarly, 'LogitsDrop' means that we use dropout after obtaining output logits from output projection layer. Compared to LogitsDrop, FD-4 directly applies dropout before the output projection layer.

	BLEU
Transformer	34.84
+FD-1, FD-2	35.46
+FD-1, FD-2, FD-3	36.10
+FD-1, FD-2, QKV_proj	35.75
+FD-1, FD-2, FD-4	36.15
+FD-1, FD-2, LogitsDrop	36.00
+FD-1, FD-2, FD-3, LogitsDrop	36.06
+FD-1, FD-2, FD-3, FD-4	36.48
+FD-1, FD-2, Encoder LayerDrop	35.24
+FD-1, FD-2, Decoder LayerDrop	35.99
+FD-1, FD-2, Encoder&Decoder LayerDrop	35.74
+FD-1, FD-2, EncoderDrop	35.64
+FD-1, FD-2, DD	36.09
+FD-1, FD-2, FD-3, FD-4, Decoder LayerDrop	36.61
+UniDrop	36.88

Table 9: The results of different dropouts on IWSLT14 De→En translation task.

'EncoderDrop' means that we randomly drop the whole information of Transformer encoder with a probability and only use previous outputs to generate the next token during training. Obviously, it is a language modeling task when dropping the encoder. 'Encoder LayerDrop' is that we apply LayerDrop only on the Transformer encoder. Table 9 shows the BLEU scores of different models on IWSLT-2014 De→En translation task. All dropout rates are tuned within $[0.1, 0.2, 0.3, 0.4]$ according to the performance of the dev set.

FD-1 and FD-2 are two existing feature dropouts for Transformer. We first use them and achieve better BLUE scores than the standard Transformer, which demonstrates the existence of serious overfitting in Transformer model. On this basis, we try to add further feature dropout to prevent Transformer from overfitting. However, we can see that QKK_proj achieves fewer improvements compared with FD-3. Similarly, LogitsDrop also underperforms FD-4. Therefore, we finally use FD-3 and FD-4 as our feature dropout components together with FD-1 and FD-2.

Among all structure dropout models, decoder LayerDrop outperforms all compared methods. In contrast, EncoderDrop only brings small improvements. Surprisingly, we can see that here the encoder LayerDrop actually has a negative effect on Transformer. Thus we integrate the promising decoder LayerDrop as structured dropout component into UniDrop.

	MNLI	QNLI	SST-2	MRPC
RoBERTa$_{\mathrm{BASE}}$	87.1	92.7	94.7	89.0
+UniDrop	87.8	93.2	95.5	90.4
w/o FD	87.3	92.9	94.8	90.1
w/o SD	87.5	93.1	95.1	89.5
w/o DD	87.7	93.1	95.0	89.5
RoBERTa$_{\mathrm{LAEGE}}$	89.8	94.3	96.3	90.4
+UniDrop	90.2	94.8	96.6	91.4
w/o FD	89.9	94.6	96.2	90.4
w/o SD	90.0	94.6	96.3	90.7
w/o DD	90.2	94.7	95.2	90.7

Table 10: Ablation Study on GLUE tasks (dev set). The "w/o FD", "w/o SD", "w/o DD" indicate respectively removing feature dropout, structure dropout, and data dropout from RoBERTa$_{\mathrm{BASE}}$+UniDrop or RoBERTa$_{\mathrm{LARGE}}$+UniDrop.

A.4 Loss Curves

Figure 5 shows the loss curves of different models during training. Overall, we can see that our Transfomer+UniDrop obtains the minimal gap of training loss and dev loss compared with other dropout models and the standard Transformer. This observation shows the better capability of UniDrop to prevent Transformer from overfitting. Benefitting from the advantage, Transfomer+UniDrop achieves the best generalization and dev loss on IWSLT14 De→En translation task.

A.5 Ablation Study on Text Classification

Table 10 show the accuracy of standard RoBERTa$_{\mathrm{BASE}}$ and RoBERTa$_{\mathrm{LARGE}}$, the models with UniDrop and corresponding ablated models on GLUE tasks. Compared the base

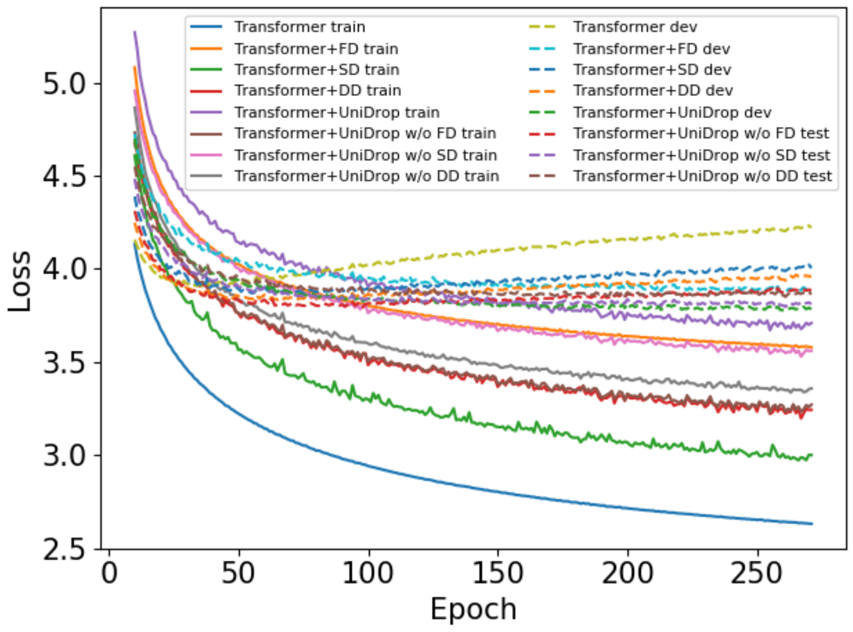


Figure 5: The training and dev loss of different models on IWSLT14 De→En translation task.

models RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$, we can observe that `UniDrop` further improves their performance on text classification tasks. After removing FD, SD, or DD from `UniDrop`, the corresponding accuracy has decreased more or less. The consistent declines again demonstrate the necessity of the feature dropout, structure dropout and data dropout for `UniDrop`.