Appendix for "R-Drop: Regularized Dropout for Neural Networks"

Anonymous Author(s)
Affiliation
Address
email

A Detailed Experimental Settings

We provide more detailed settings for the experiments of each task in this part.

A.1 Neural Machine Translation

For all the NMT tasks, we use the public datasets from IWSLT competitions[1] and WMT competitions[2]. We tokenize all the datasets with byte-pair-encoding (BPE) [10] approach with the dictionary built jointly upon the source and target sentence pairs except the IWSLT17 En↔Zh translation dataset that is built separately. After tokenization, the resulted vocabularies for IWSLT datasets are near $10k$, while for WMT datasets, the vocabulary size is about $32k$.

To train the Transformer based NMT models, we use `transformer_iwslt_de_en` configuration for IWSLT translations, which has 6 layers in both encoder and decoder, embedding size 512, feed-forward size $1,024$, attention heads 4, dropout value 0.3, weight decay 0.0001. For the WMT experiments, the `transformer_vaswani_wmt_en_de_big` setting has 6 layers in encoder and decoder, embedding size $1,024$, feed-forward size $4,096$, attention heads 16, dropout value 0.1, attention dropout 0.1 and relu dropout 0.1. The training is optimized with Adam [5] with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$. The learning rate scheduler is `inverse_sqrt` with default learning rate 0.0005 and warmup steps $4,000$. Label smoothing [11] is adopted with value 0.1. Our code implementation is based on open-source Fairseq[3]. We train the IWSLT translations on 1 GEFORCE RTX 3090 card and the WMT translations on 8 GEFORCE RTX 3090 cards.

To evaluate the performance, we use `multi-bleu.perl`[4] to evaluate IWSLT14 En↔De and all WMT tasks for a fair comparison with previous works [15, 8]. For other NMT tasks, we use `sacre-bleu`[5] [9] for evaluation. When inference, we follow [12] to use beam size 4 and length penalty 0.6 for WMT14 En→De, beam size 5 and penalty 1.0 for other tasks.

A.2 Abstrative Summarization

For summarization, we take the pre-trained BART [6] model as backbone and fine-tune on the CNN/DailyMail dataset[6]. BART is a pre-trained sequence-to-sequence model based on the masked source input and autoregressive target output, which contains 12 layers of Transformer encoder and 12 layers of Transformer decoder, the embedding size is $1,024$, and the feed-forward size is $4,096$.

[1] `https://iwslt.org/`

[2] `https://www.statmt.org/wmt14/translation-task.html`

[3] `https://github.com/pytorch/fairseq/tree/master/examples/translation`

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

[5] `https://github.com/mjpost/sacrebleu`

[6] `https://github.com/abisee/cnn-dailymail`

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Hyper-parameter	CoLA	MRPC	RTE	SST-2	MNLI	QNLI	QQP	STS-B
Learning Rate	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
Max Update	5336	2296	3120	20935	123873	33112	113272	3598
Max Sentence (Batch)	16	16	8	32	32	32	32	16
Dropout	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Coefficient α	0.5	1.0	1.0	1.0	0.5	1.0	0.5	1.0

Table 1: Hyper-parameters when fine-tuning our models on GLUE benchmark.

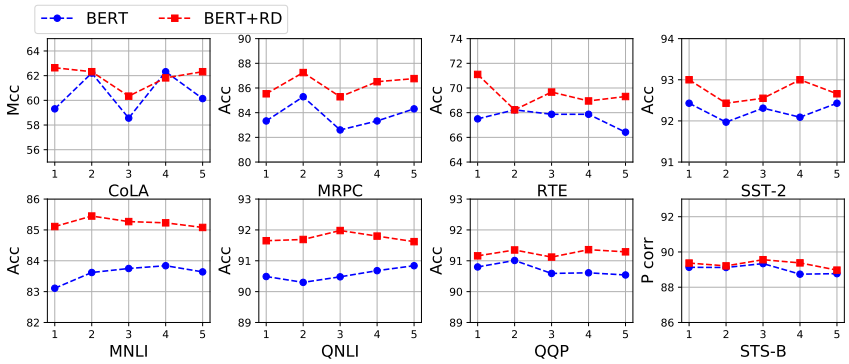

Figure 1: Results on 8 GLUE tasks with different random seed.

Dropout value is 0.1. During fine-tuning, we follow the hyper-parameters used in [6]. The pre-trained model and the backbone implementations are all from Fairseq[7]. The training is conducted on 8 GEFORCE RTX 3090 GPU cards.

A.3 Language Modeling

For language modeling, we train on the Transformer decoder [12] and Adaptive Input Transformer [1] models. The configuration for Transformer is `transformer_lm_gpt`, which contains 12 layers with embedding size 768 and feed-forward size $3,072$, attention heads 12. Dropout and attention dropout are 0.1. For Adaptive Input Transformer, the configuration is `transformer_lm_wiki103` with 16 layers, embedding size $1,024$, feed-forward size $4,096$, attention heads 16, dropout 0.3, attention dropout 0.1, gelu dropout 0.1 and adaptive softmax dropout 0.2. We train Transformer model for $50k$ steps and Adaptive Input Transformer for $286k$ steps. The development is based on the code base Fairseq[8]. The training is on 8 Tesla V100 GPU cards.

A.4 Language Understanding

For language understanding tasks, we follow the popular pre-training and fine-tuning methodology, and the fine-tuned sets are the GLUE [13] benchmark. We follow previous works [2, 7] to work on the 8 tasks, including singe-sentence classification tasks (CoLA, SST-2), sentence-pair classification tasks (MNLI, QNLI, RTE, QQP, MRPC), and sentence-pair regression task (STS-B). The detailed data statistics can be found from the original paper [13].

The pre-trained BERT-base model is the Transformer [12] encoder network, which contains 12 layers with embedding size 768, feed-forward size $3,072$ and attention heads 12. Correspondingly, the Roberta-large model contains 24 layers with embedding size $1,024$, feed-forward size $4,096$ and attention heads 16. During fine-tuning, we use Adam [5] as our optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-6}$, and L_2 weight decay of 0.01. We select the learning rate in range $\{5 \times 10^{-6}, 10^{-5}\}$ and batch size in $\{8, 16, 32\}$. Other hyper-parameter settings are mostly same as previous works [7]. The

[7] `https://github.com/pytorch/fairseq/tree/master/examples/bart`
[8] `https://github.com/pytorch/fairseq/tree/master/examples/language_model`

pre-trained model and the backbone implementations are all from Huggingface Transformers[9]. We report the specific settings of several important hyper-parameters in Table 1, including the dropout value. The fine-tuning experiments are conducted on 1 GEFORCE RTX 3090 GPU card.

Further, to give a clear comparison of our R-Drop based fine-tuning and vanilla fine-tuning, we plot the performance changes from different random seeds over the pre-trained BERT model on each GLUE task. The curves are shown in Figure 1. We can see that consistent improvements are achieved on different random seeds, which means our R-Drop can robustly help improve the mode generalization and model performance.

MSE Regularization Our R-Drop is presented under the KL-divergence between two distributions. To extend our method into the regression task, such as STS-B in GLUE, we introduce the MSE-based regularization. For input data (x, y), we forward the x two times similarly as in classification and obtain the two predicted values y_1' and y_2'. Then we regularize these two predicted values with MSE as follow:

$$\mathcal{L}_{mse_r} = ||y_1' - y_2'||_2, \tag{1}$$

and we add $\mathcal{L}_{mse_r}$ with conventional MSE loss: $\mathcal{L}_{mse} = ||y - y_1'||_2 + ||y - y_2'||_2$. The final optimization objective is:

$$\mathcal{L} = \mathcal{L}_{mse} + \alpha \mathcal{L}_{mse_r}. \tag{2}$$

A.5 Image Classification

The image classification task is evaluated with the recent popular Vision Transformer (ViT) [4] model, which is the same as Transformer but with the image patch data as input. We take the two publicly released models[10], ViT-B/16 and ViT-L/16, which are pre-trained on ImageNet-21k [3] dataset with $21k$ classes and $14M$ images in total. ViT-B/16 is a Transformer model with 12 Transformer encoder layers, embedding size 768, feed-forward size $3,072$ and attention heads 12, while ViT-L/16 with 24 layers, $1,024$ embedding size, $4,096$ feed-forward size and 16 attention heads. We only conduct the fine-tuning stage experiments on CIFAR-100 and ImageNet. Note that the ImageNet results are computed without additional techniques (Polyak averaging and 512 resolution images) used to achieve results in [4]. During fine-tuning, the dropout values are 0.1 for both models. Fine-tuning is on 8 GEFORCE RTX 3090 GPU cards.

B Theoretical Discussion of R-Drop

For self-contained, we reformulate the constrained optimization problem as:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_\xi[-\log \mathcal{P}_\xi^w(y_i|x_i)], \tag{3}$$

$$\textit{s.t.} \quad \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}_{\xi^{(1)}}^w(y_i|x_i)||\mathcal{P}_{\xi^{(2)}}^w(y_i|x_i)))] = 0. \tag{4}$$

For a multi-layer perceptron (MLP) with dropout, its output is represented as:

$$\mathcal{P}_\xi(y|x) = \texttt{softmax}(v h_{\xi^L}^L); \quad h_{\xi^l}^l = \xi^l \odot \sigma(w^l h_{\xi^{l-1}}^{l-1}), l = 1, \cdots, L; \quad h_{\xi^0}^0 = \xi_0 \odot x, \tag{5}$$

where $\sigma(\cdot)$ is the activation function.

We also consider the MLP with skip/residual connections (which is more popular than plain MLP), whose output after dropout is represented as:

$$\mathcal{P}_\xi^{res}(y|x) = \texttt{softmax}(v h_{\xi^L}^L); \quad h_{\xi^l}^l = \xi^l \odot \sigma(w^l h_{\xi^{l-1}}^{l-1}) + h_{\xi^{l-1}}^{l-1}, l = 1, \cdots, L; \quad h_{\xi^0}^0 = \xi_0 \odot x. \tag{6}$$

[9] https://github.com/huggingface/transformers

[10] https://github.com/jeonsworld/ViT-pytorch

For a Transformer model, the output can be represented as:

$$z = SA(Q, K, V) = \texttt{softmax}\left(\frac{(Qw_1)(Kw_2)^T}{\sqrt{d}}\right)Vw_3, \tag{7}$$

$$f(z) = v_2 \max(0, v_1 z), \tag{8}$$

where v_1, v_2, w_1, w_2, w_3 are learnable parameters, $SA(\cdot)$ stands for self-attention operation, and $f(\cdot)$ is the position-wise feed-forward network as described in Transformer [12]. We only consider dropout inside the self-attention layer, i.e.,

$$z_\xi = SA(Q, K, V) = \left(\xi_2 \odot \texttt{softmax}\left(\frac{((\xi_1 \odot Q)w_1)((\xi_1 \odot K)w_2)^T}{\sqrt{d}}\right)\right)((\xi_1 \odot V)w_3), \tag{9}$$

$$f(z_\xi) = v_2 \max(0, (v_1 z_{\xi_1})), \tag{10}$$

$$\mathcal{P}_\xi^{TF}(y|x) = \texttt{softmax}(f(z_\xi)), \tag{11}$$

where the size of random matrix ξ_1 equal to the size of Q[11], and the size of ξ_2 equal to the size of the value after softmax.

Now we provide the proof for the next proposition. Here, we consider a sampling strategy called dropout$_2$, which randomly samples p portion of hidden neurons at each layer. Thus, this strategy can keep the width for the sampled sub-structure and avoid extremely thin sub-structures.

Proposition B.1. *For a fully-connected neural network $\mathcal{P}^w(y|x)$, e.g., multilayer perceptron and Transformer, the constraint in Equation (4) is equivalent to constraining all the parameters of the network at the same layer to be equal.*

Proof: We consider two sub-structures $\mathcal{P}_1$ and $\mathcal{P}_2$ that differ from each other at only one hidden node. Without loss of generality, we suppose the hidden node that is only contained in $\mathcal{P}_1$ as h_i^l and the hidden node that is only contained in $\mathcal{P}_2$ as h_j^l.

Since we constrain the $KL(\mathcal{P}_1||\mathcal{P}_2) = 0, \forall x$, which is equivalent to $\mathcal{P}_1(y|x) = \mathcal{P}_2(y|x), \forall x$. Then we have $h_i^l = h_j^l, \forall x$. If not, we can construct an $x \in \mathbb{R}^d$ to make their final output unequal[12]. Because the two sub-structures share the other hidden nodes, the inputs of h_i^l and h_j^l are the same, i.e., $h_i^l = \sigma(\tilde{w}_i^l \tilde{h}^{l-1}), h_j^l = \sigma(\tilde{w}_j^l \tilde{h}^{l-1})$, where w_i denotes the ingoing weights of h_i^l and we use the subscript $\tilde{}$ to denote the hidden output and weight matrix after the dropout. Then, we have $\tilde{w}_i^l = \tilde{w}_j^l$. Traversing all the paired sub-structures (produced by dropout$_2$) that differ at only one hidden node, we can get that all the parameters at the same layer are equal.

Similarly, for the Transformer model, we can also get the results in the above Proposition, but it is established by removing the redundancy in weights. In detail, the weights w_1 and w_2 have redundant freedom due to the scaling invariant property of $Qw_1(Kw_2)^T$ [14], i.e., If we regard w_1 and w_2 as two linear layers of MLP, the ingoing weights of one hidden node are multiplying a scalar c and its outgoing weights are dividing by c, the resulted output will not change. To remove the redundancy, we fix the first column in w_1 (one ingoing weight for each hidden node) to have the same value. Then Proposition B.1 is established for w_1, w_2, w_3.

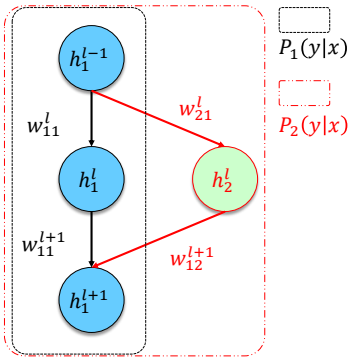

Figure 2: Example of MLP in 1).

Remark: Although Proposition B.1 is established for a different sampling strategy with original dropout, it will approach to the result for original dropout **with high probability** because the mean portion of the sampled neurons will be closed to p according to central limit theorem.

Proposition B.1 can be regarded as a result of original dropout in the sense of high probability. Next, we briefly discuss what will happen if we consider the original sampling strategy in dropout. Since

[11] Please note that for self-attention, $Q = K = V$.

[12] Here, we suppose the activation function σ satisfies that $w^2\sigma(w^1 x) = \tilde{w}^2\sigma(\tilde{w}^1 x), \forall x$ if and only if $w^2 = \tilde{w}^2, w_1 = \tilde{w}^1$.

Figure 3: Results of R-Drop and Transformer with double batch.

the original strategy contains more sub-structures, it actually brings a much stronger restriction to the model. For example, the model has the chance to degenerate to a zero mapping.

1) For MLP: If we consider the original sampling strategy in dropout, the constraint of Equation (4) is equivalent to constraining one layer of the MLP to be zero. We can consider an extreme sub-structure that there is only one hidden node after dropout and denote its output as:

$$\mathcal{P}_1(y|x) = \texttt{softmax}(vh_1^L); \quad h_1^l = \sigma(w_{11}^l h_1^{l-1}), l = 1, \cdots, L; \quad h_1^0 = x_1,$$

where h_1^l denotes the 1-th hidden node at layer l and w_{11}^l denotes the weights that connects h_1^{l-1} and h_1^l. Then we consider another sub-structure $\mathcal{P}_2(y|x)$ which has one more hidden node at layer l (denoted as h_2^l) than $\mathcal{P}_1(y|x)$, i.e.,

$$h_2^l = \sigma(w_{21}^l h_1^{l-1}); h_1^{l+1} = \sigma(w_{11}^{l+1} h_1^l + w_{12}^{l+1} h_2^l),$$

and other computations are the same as the calculation of $\mathcal{P}_1(y|x)$ (see the example in Figure 2). Therefore, the sub-structure $\mathcal{P}_2$ has two additional parameters w_{12}^{l+1} and w_{21}^l compared with the sub-structure $\mathcal{P}_1$, and they share weights on $w_{11}^l, l = 1, \cdots, L$. Because we constrain the $KL(\mathcal{P}_1||\mathcal{P}_2) = 0, \forall x$, which is equivalent to $\mathcal{P}_1(y|x) = \mathcal{P}_2(y|x), \forall x$. Therefore, we need $\sigma(w_{11}^{l+1} h_1^l + w_{12}^{l+1} h_2^l) = \sigma(w_{11}^{l+1} h_1^l), \forall x$. Then we must have $w_{12}^{l+1} = 0$ or $w_{21}^l = 0$. Similarly, we can traverse all such extreme cases to get the results.

2) For MLP with skip/residual connections: Using the similar technique in 1), if we consider the MLP with skip connection $\mathcal{P}_\xi^{res}(y|x)$ and the original sampling strategy in dropout, the constraint in Equation (4) is equivalent to constraining all the parameters equals zero. The network will degenerate to an identity mapping of x.

C More Studies

C.1 Batch Size Doubled Training

As discussed in Section 2.2, we implement the algorithm by repeating input data x once and concatenate the x with repeated one in the same mini-batch to forward once. This is similar to enlarge the batch size to be double size at each step. The difference is that half of the data are the same as the other half, while directly doubling the batch size, the data in the same mini-batch are all different. Therefore, we are still interested in the result of directly doubling the batch size to see the performance. We conduct experiments on IWSLT14 De→En translation with Transformer, and the batch size is enlarged from $4,096$ to be $8,192$. The result is 34.93 BLEU score. We can see that though slight improvement is achieved (compared to baseline 34.64), it falls far behind our strong performance 37.25. For the detailed training cost for each step, we represent the number here:

Transformer + Double Batch costs near 9ms per step, while Transformer + DR costs about 10ms per step. The additional cost is from the KL-divergence loss backward computation. We can see the cost is 1.1 times, which is a negligible cost. We also plot the valid BLEU curves along with the training for this study. The curves are shown in Figure 3. Compare to this batch size doubled training and our R-Drop, we can clearly see the advantage of R-Drop. With similar training cost, R-Drop gradually improves the performance to a much stronger one. In the figures, we also plot the curve for Transformer with the original batch size training (e.g., $4,096$) for a better comparison.

C.2 Importance of KL-divergence

Our method introduces a KL-divergence between the two distributions from the same sample. In this study, we specifically investigate on the importance of the KL-divergence loss. Thus, this ablation removes the $\mathcal{L}_{KL}$ loss between the two distributions and only keeps the $\mathcal{L}_{NLL}$ loss for training. Similar as other studies, we work on IWSLT14 De→En translation and the model is Transformer. The result is also 34.93 BLEU score (same as above enlarge batch size), it is slightly better than Transformer baseline (34.64), but far worse from our R-Drop based training result 37.25 BLEU score. This result well demonstrates the importance and effectiveness of our introduced KL-divergence loss.

References

[1] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2018.

[2] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2019.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, 2020.

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

[8] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, 2018.

[9] Matt Post. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, 2018.

[10] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, 2016.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, 2017.

[13] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018.

[14] Yue Wang, Yuting Liu, and Zhi-Ming Ma. The scale-invariant space for attention layer in neural network. *Neurocomputing*, 392:1–10, 2020.

[15] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*, 2019.