FreeLB: Enhanced Adversarial Training for Natural Language Understanding
(2020)

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, Jingjing Liu

Summary

Contributions

The dominant process in NLP, is to first pre-train a language model using a self-supervised objective, then fine-tune the pre-trained model for a given task using labeled examples. However, although widely adopted, existing approaches for fine-tuning pre-trained model are generally unstable across different hyper-parameters settings and require many runs to obtain good results. To solve this, the authors proposed to use adversarial training to improve the generalization an stabilize the training by minimizing risk for label preserving input perturbations. More precisely, they propose FreeLB, an adversarial method that forces the model to produce consistent prediction over various adversarial perturbation added to the word embedding, thus promoting higher invariance in the embedding space and producing good results.

Method

In order to improve the generalization of these pre-trained language models (eg, BERT, ALVERT, T5, RoBERTa) on downstream language understanding tasks, the paper consists of enhancing their robustness in the embedding space during fine-tuning on these tasks. To this end, we need to create virtual adversarial examples in the embedding space and then train the model on them. However, creating actual adversarial examples for language is difficult since it remains unclear how to construct label preserving adversarial examples via work or character replacement without any human evaluations. To overcome this, we can simply create the effects of an adversarial examples rather than creating the adversarial example itself.

Let $Z = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n]$ be a sequence of one-hot representations of the input subwords, $\boldsymbol{V}$ as the subword embeddings, and the language model, or the encoder as $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{X})$ with $\boldsymbol{X} = \boldsymbol{V}\boldsymbol{Z}$. The objective to find the optimal parameter of the encoder θ' to minimize the maximum risk for any adversarial perturbation δ over the embeddings within a norm ball (to maintain a label preserving perturbation). The optimization objective can be written as follows

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{Z},y)\sim\mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|\leq\epsilon} L\left(f_{\boldsymbol{\theta}}(\boldsymbol{X}+\boldsymbol{\delta}), y\right) \right]$$

with $\mathcal{D}$ as the data distribution, y as the label and L as the loss function. However, in order to apply this double optimization (finding the maximum perturbation then minimizing the risk) is computationally expensive, and requires multiple (say K) backward and forward pass through the network per training iteration to achieve a high-level robustness, resulting in a large increase in the training time when training large language models. A possible solution proposed by FreeAT is to simplify this by only taking one descent step on the parameters together for ech K ascent steps on the perturbation, however, in this case, we have stale gradients where the perturbation δ_t at time step t might not be the perturbation corresponding to parameters θ_t but that of an older version of the parameters.

To obtain better solutions for the inner max and avoid such limitations, the authors propose FreeLB, which first performs K iteration to craft the adversarial perturbation, accumulated the gradients at each of the K iterations, and then the parameters are updated using the accumulated parameters.

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{Z},y)\sim\mathcal{D}} \left[\frac{1}{K} \sum_{t=0}^{K-1} \max_{\boldsymbol{\delta}_t \in \mathcal{I}_t} L\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{X}+\boldsymbol{\delta}_t\right), y\right) \right]$$

This is equivalent to replacing the input batch $\boldsymbol{X}$ with K-times virtually augmented input batch, where each new batch have embeddings $\boldsymbol{X}+\boldsymbol{\delta}_0, \ldots, \boldsymbol{X}+\boldsymbol{\delta}_{K-1}$.

One important consideration is using dropout, since the application of dropout in one of the K inner iterations results in optimizing the perturbation δ for a different network. To avoid this, the authors fix the dropout mask over all the inner K iterations.

Results

Method	MNLI (Acc)	QNLI (Acc)	QQP (Acc)	RTE (Acc)	SST-2 (Acc)	MRPC (Acc)	CoLA (Mcc)	STS-B (Pearson)
Reported	90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4
ReImp	-	-	-	85.61 (1.7)	96.56 (.3)	90.69 (.5)	67.57 (1.3)	92.20 (.2)
PGD	90.53 (.2)	94.87 (.2)	92.49 (.07)	87.41 (.9)	96.44 (.1)	90.93 (.2)	69.67 (1.2)	92.43 (7.)
FreeAT	90.02 (.2)	94.66 (.2)	92.48 (.08)	86.69 (15.)	96.10 (.2)	90.69 (.4)	68.80 (1.3)	92.40 (.3)
FreeLB	**90.61** (.1)	**94.98** (.2)	**92.60** (.03)	**88.13** (1.2)	**96.79** (.2)	**91.42** (.7)	**71.12** (.9)	**92.67** (.08)

Table 1: Results (median and variance) on the dev sets of GLUE based on the RoBERTa-large model, from 5 runs with the same hyperparameter but different random seeds. ReImp is our reimplementation of RoBERTa-large. The training process can be very unstable even with the vanilla version. Here, both PGD on STS-B and FreeAT on RTE demonstrates such instability, with one unconverged instance out of five.

Model	Score	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX
BERT-base[1]	78.3	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	65.1	34.2
FreeLB-BERT	79.4	54.5	93.6	88.1/83.5	87.7/86.7	72.7/89.6	85.7/84.6	91.8	70.1	65.1	36.9
MT-DNN[2]	87.6	**68.4**	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9/87.4	96.0	86.3	89.0	42.8
XLNet-Large[3]	88.4	67.8	**96.8**	93.0/90.7	91.6/91.1	74.2/90.3	90.2/89.8	98.6	86.3	**90.4**	47.5
RoBERTa[4]	88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	**98.9**	88.2	89.0	48.7
FreeLB-RoB	**88.8**	68.0	**96.8**	**93.1/90.8**	**92.4/92.2**	**74.8/90.3**	**91.1/90.7**	98.8	**88.7**	89.0	**50.1**
Human	87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-

Table 2: Results on GLUE from the evaluation server, as of Sep 25, 2019. Metrics are the same as the leaderboard. Number under each task's name is the size of the training set. FreeLB-BERT is the single-model results of BERT-base finetuned with FreeLB, and FreeLB-RoB is the ensemble of 7 RoBERTa-Large models for each task. References: [1]: (Devlin et al., 2019); [2]: (Liu et al., 2019a); [3]: (Yang et al., 2019); [4]: (Liu et al., 2019b).

	ARC-Easy		ARC-Challenge		ARC-Merge		CQA		
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Test (E)
RoBERTa (Reported)	-	-	-	-	-	-	78.43	72.1	72.5
RoBERTa (ReImp)	84.39	84.13	64.54	64.44	77.83	77.62	77.56	-	-
FreeLB-RoBERTa	**84.91**	84.81	65.89	65.36	78.37	78.39	**78.81**	**72.2**	**73.1**
AristoRoBERTaV7 (MTL)	-	85.02	-	66.47	-	78.89	-	-	-
XLNet + RoBERTa (MTL+Ens)	-	-	-	67.06	-	-	-	-	-
FreeLB-RoBERTa (MTL)	**84.91**	**85.44**	**70.23**	**67.75**	**79.86**	**79.60**	-	-	-

Table 3: Results on ARC and CommonsenseQA (CQA). ARC-Merge is the combination of ARC-Easy and ARC-Challenge, "MTL" stands for multi-task learning and "Ens" stands for ensemble. Results of XLNet + RoBERTa (MTL+Ens) and AristoRoBERTaV7 (MTL) are from the ARC leaderboards. Test (E) denotes the test set results with ensembles. For CQA, we report the highest dev and test accuracies among *all* models. The models with 78.81/72.19 dev/test accuracy (as in the table) have 71.84/78.64 test/dev accuracies respectively.

Methods	Vanilla	FreeLB-3*	FreeLB-3	YOPO-3-2	YOPO-3-3
RTE	85.61 (1.67)	87.14 (1.29)	**88.13 (1.21)**	87.05 (1.36)	87.05 (0.20)
CoLA	67.57 (1.30)	69.31 (1.16)	**71.12 (0.90)**	70.40 (0.91)	69.91 (1.16)
MRPC	90.69 (0.54)	90.93 (0.66)	**91.42 (0.72)**	90.44 (0.62)	90.69 (0.37)

Table 4: The median and standard deviation of the scores on the dev sets of RTE, CoLA and MRPC from the GLUE benchmark, computed from 5 runs with the same hyper-parameters except for the random seeds. We use FreeLB-m to denote FreeLB with m ascent steps, and FreeLB-3* to denote the version without reusing the dropout mask.

Methods	RTE			CoLA			MRPC		
	M-Inc (10^{-4})	M-Inc (R) (10^{-4})	N-Loss (10^{-4})	M-Inc (10^{-4})	M-Inc (R) (10^{-4})	N-Loss (10^{-4})	M-Inc (10^{-3})	M-Inc (R) (10^{-3})	N-Loss (10^{-3})
Vanilla	5.1	5.3	4.5	6.1	5.7	5.2	10.2	10.2	1.9
PGD	4.7	4.9	6.2	128.2	130.1	436.1	5.7	5.7	5.4
FreeLB	3.0	2.6	4.1	1.4	1.3	7.2	3.6	3.6	2.7

Table 5: Median of the maximum increase in loss in the vicinity of the dev set samples for RoBERTa-Large model finetuned with different methods. Vanilla models are naturally trained RoBERTa's. M-Inc: Max Inc, M-Inc (R): Max Inc (R). Nat Loss (N-Loss) is the loss value on clean samples. Notice we require *all* clean samples here to be correctly classified by all models, which results in 227, 850 and 355 samples for RTE, CoLA and MRPC, respectively. We also give the variance in the Appendix.

Method	MNLI (Acc)	QNLI (Acc)	QQP (Acc)	RTE (Acc)	SST-2 (Acc)	MRPC (Acc)	CoLA (Mcc)	STS-B (Pearson)
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8
RoBERTa-large	90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4
RoBERTa-FreeLB	90.6	95.0	**92.6**	88.1	96.8	91.4	71.1	92.7
ALBERT-xxlarge-v2	90.8	95.3	92.2	89.2	96.9	90.9	71.4	93.0
ALBERT-FreeLB	**90.9**	**95.6**	92.5	**89.9**	**97.0**	**92.4**	**73.1**	**93.2**

Table 6: Results (median) on the dev sets of GLUE from 5 runs with the same hyperparameter but different random seeds. RoBERTa-FreeLB and ALBERT-FreeLB are RoBERTa-large and ALBERT-xxlarge-v2 models fine-tuned with FreeLB on GLUE. All other results are copied from (Lan et al., 2020).