We need to talk about random seeds

Anonymous ACL submission

Abstract

Modern neural network libraries all take as a hyperparameter a random seed, typically used to determine the initial state of the model parameters. This position piece argues that there are some safe uses for random seeds: as part of the hyperparameter search to select a good model, creating an ensemble of several models, or measuring the sensitivity of the training algorithm to the random seed hyperparameter. It argues that some uses for random seeds are risky: using a fixed random seed for "replicability" and varying only the random seed to create score distributions for performance comparison. An analysis of 85 recent publications from the ACL Anthology shows that more than 50% contain risky uses of random seeds.

1 Introduction

Modern neural network libraries all take as a hyperparameter a *random seed*, a number that is used to initialize a pseudorandom number generator. That generator is typically used to determine the initial state of model parameters, but may also affect optimization in other ways, such as selecting which units to mask under dropout, or selecting which instances of the training data go into each minibatch during gradient descent. Like any hyperparameter, neural network random seeds can have a large or small impact on model performance depending on the specifics of the architecture and the data. Thus, it is important to tune the random seed hyperparameter as we would any other hyperparameter that affects optimization, such as learning rate or regularization strength.

Such tuning is especially important with the pre-trained transformer architectures currently popular in NLP (BERT, Devlin et al., 2019; RoBERTa Liu et al., 2019; etc.), which are quite sensitive to their random seeds (Risch and Krestel, 2020; Dodge et al., 2020; Mosbach et al., 2021). Several solutions to this problem have been proposed, including specific optimizer setups (Mosbach et al., 2021), ensemble methods (Risch and Krestel, 2020), and explicitly tuning the random seed like other hyperparameters (Dodge et al., 2020).

The NLP community thus has some awareness of the problems that random seeds present, but it is inconsistent in its approaches to solving those problems. The remainder of this position piece first presents a taxonomy of different ways that neural network random seeds are used in the NLP community, explaining which uses are safe and which are risky. It then reviews 85 articles recently published in the ACL Anthology, categorizing their random seed uses based on the taxonomy. This analysis shows that more than 50% of the articles include risky uses of random seeds, suggesting that the NLP community still needs a broader discussion about how we approach random seeds.

2 A taxonomy of random seed uses

This section highlights five common uses of neural network random seeds in the NLP community, and categorizes them as either safe or risky.

2.1 Safe use: Model selection

The random seed is a hyperparameter of a neural network architecture that determines where in the model parameter space optimization should begin. It may also affect optimization by determining the order of minibatches in gradient descent, or through mechanisms like dropout's random sampling of unit activations. As the random seed is a hyperparameter, it can and should be tuned just as other hyperparameters are. Unlike some other hyperparameters, there is no intuitive explanation of why one random seed would be better or worse than another, so the typical strategy is to try a number of randomly selected seeds. For example:

> *Instead, we compensate for the inherent randomness of the network by train-*

ing multiple models with randomized initializations and use as the final model the one which achieved the best performance on the validation set... (Björne and Salakoski, 2018)

The test results are derived from the 1-best random seed on the validation set. (Kuncoro et al., 2020)

2.2 Safe use: Ensemble creation

Ensemble methods are an effective way of combining multiple machine-learning models to make better predictions (Rokach, 2010). A common approach to creating neural network ensembles is to train the same architecture with different random seeds, and have the resulting models vote (Perrone and Cooper, 1995). For example:

In order to improve the stability of the RNNs, we ensemble five distinct models, each initialized with a different random seed. (Nicolai et al., 2017)

Our model is composed of the ensemble of 8 single models. The hyperparameters and the training procedure used in each single model are the same except the random seed. (Yang and Wang, 2019)

2.3 Safe use: Sensitivity analysis

Sometimes it is useful to demonstrate how sensitive a neural network architecture is to a particular hyperparameter. For example, Santurkar et al. (2018) shows that batch normalization makes neural network architectures less sensitive to the learning rate hyperparameter. Similarly, it may be useful to show how sensitive neural network architectures are to their random seed hyperparameter. For example:

We next (§3.3) examine the expected variance in attention-produced weights by initializing multiple training sequences with different random seeds... (Wiegreffe and Pinter, 2019)

Our model shows a lower standard deviation on each task, which means our model is less sensitive to random seeds than other models. (Hua et al., 2021)

2.4 Risky use: Single fixed seed

NLP articles sometimes pick a single fixed random seed, claiming that this is done to improve consistency or replicability. For example:

An arbitrary but fixed random seed was used for each run to ensure reproducibility... (Le and Fokkens, 2018)

For consistency, we used the same set of hyperparameters and a fixed random seed across all experiments. (Lin et al., 2020)

Why is this risky? First, fixing the random seed does not guarantee replicability. For example, the tensorflow library has a history of producing different results even given the same random seeds, especially on GPUs (Two Sigma, 2017; Kanwar et al., 2021). Second, not tuning the random seed hyperparameter has the same drawbacks as not tuning any other hyperparameter: performance will be an underestimate of the performance the architecture is capable of with a tuned model.

What should one do instead? The random seed should be tuned as any other hyperparameter. Dodge et al. (2020), for example, show that doing so leads to simpler models exceeding the published results of more complex state-of-the-art models on multiple GLUE tasks (Wang et al., 2018). If compute resources are scarce, it is reasonable to restrict the space of hyperparameters explored (and thus the number of random seeds explored). This can be done with techniques such as random hyperparameter search (Bergstra and Bengio, 2012) where n hyperparameter settings are sampled from the space of all hyperparameter settings (with random seeds treated the same as all other hyperparameters) and the value of n is tuned to match the availability of compute resources. In an extremely resource-limited scenario, this could result in selecting only a single value of the random seed or only a single value of some other hyperparameter. This might be acceptable given the constraints, especially if accompanied by an explicit acknowledgement of the risks of underestimating performance.

2.5 Risky use: Performance comparison

It is a good idea to compare not just the point estimate of a single model's performance, but distributions of model performance, as comparing performance distributions may result in more reliable conclusions (Reimers and Gurevych, 2017; Dodge et al., 2019; Radosavovic et al., 2020). However, it has sometimes been suggested that such distributions can be obtained by training the same architecture and varying only the random seed. For example:

> *We re-ran both implementations multiple times, each time only changing the seed value of the random number generator. . . The score distribution. . .* (Reimers and Gurevych, 2017)

> *Indeed, the best approach is to stop reporting single-value results, and instead report the distribution of results from a range of seeds.* (Crane, 2018)

Why is this risky? If the goal is to compare the best possible model trainable from one architecture to the best possible model trainable from another architecture, as in the case of leaderboard comparisons, then varying random seeds is generating a bunch of suboptimal models for that comparison. If the goal is to compare the family of models that result from training one architecture to the family of models that result from training another architecture, then varying only the random seed is generating a small biased slice of the family, since the family consists of the model variations across all hyperparameter settings, not just random seeds.

What should one do instead? If the goal is to compare the best possible models trainable from different architectures, then the random seed needs to be tuned just as we would for any other hyperparameter. It's still a good idea to compare distributions, rather than point estimates, so standard statistical techniques can be applied. For example, bootstrap samples may be drawn from the test set, and evaluating a model on each of those samples will give a distribution over the model's expected performance (Dror et al., 2018). Comparing these distributions will give a statistically sound estimate of whether the best model found for one neural network architecture outperforms the best model found for another. If the goal is instead to compare families of models, then it makes sense to train many versions of the same architecture, but they should be sampled to vary across all hyperparameters, not just the random seed hyperparameter[1]. Comparing these distributions will give a statistically sound estimate of whether one architecture (and not just the best-tuned instance of that architecture) is better than another.

[1]Occasionally, the random seed might be the only hyperparameter, e.g., an extreme black box machine-learning scenario where the only way to get model variants is to vary the order in which training data instances are fed to the model. In such cases, it would be acceptable to vary only the random seed.

Type	Purpose	Count
Safe	Model selection	12
Safe	Ensemble creation	13
Safe	Sensitivity analysis	12
Safe sub-total		37
Risky	Fixed seed	24
Risky	Performance comparison	24
Risky sub-total		48

Table 1: Uses of neural network random seeds for 85 ACL Anthology articles.

3 Random seed uses in ACL

Having introduced both safe and risky uses of neural network random seeds, we now turn to the current state of NLP with respect to such seeds.

On 29 Jun 2021, I searched the ACL Anthology for articles containing the phrases "random seed" and "neural network"[2]. The ACL Anthology search interface returns a maximum of 10 pages of results, with 10 results per page, so I collected 100 search results. Non-articles (entire proceedings, author pages, supplementary material) were excluded, as were articles where the random seeds were not used to initialize a neural network (e.g., they were used only for dataset selection). The result was 85 articles, from publications between 2015 and 2021.

I read each of the articles and categorized its use of random seeds into one of the five purposes introduced in section 2. While it is conceptually possible for an article to fall into more than one category (e.g., having both ensembles and sensitivity analysis) the vast majority of articles I read fell into a single category, typically with just a single sentence where *random seed* was used. For the tiny fraction of articles where more than one category applied, since my goal only was to get a rough distribution of random seed use, I selected a "primary" category arbitrarily from the categories present. The supplementary material for this article includes a spreadsheet detailing each article reviewed, its category of random seed use, and a snippet of text from the article justifying my assignment of that category.

Table 1 shows the distribution of articles across the different random seed purposes. More than half

[2]`https://www.aclweb.org/anthology/search/?q=%22random+seed%22+%22neural+network%22`

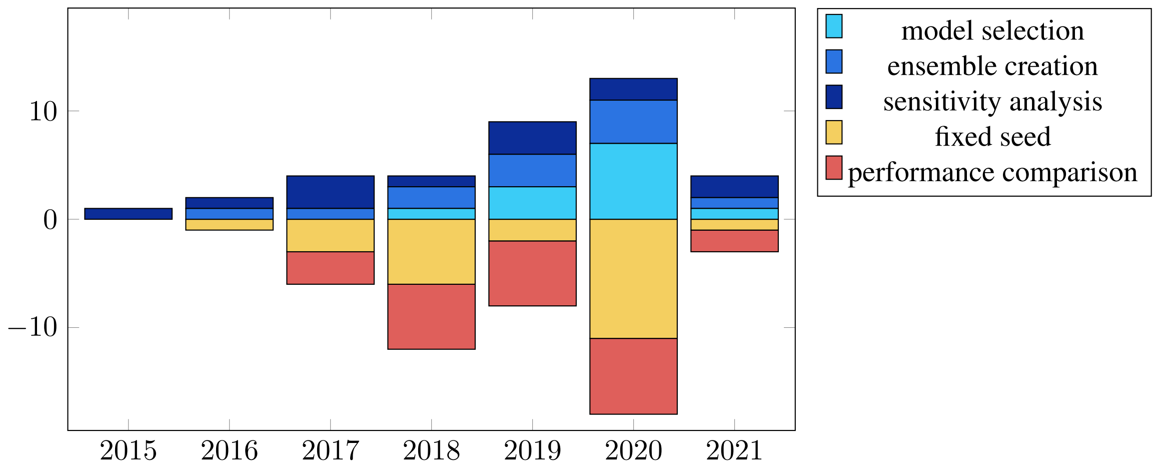

Figure 1: Uses of neural network random seeds by year for 85 ACL Anthology articles.

of the articles (48) include a risky use of random seeds, with 24 using a single fixed seed and 24 using only random seeds to generate distributions for performance comparisons. This suggests that NLP researchers are often using neural network random seeds without the necessary care.

One might wonder if the NLP community is getting better over time, that is, if risky uses are on the decline as NLP researchers become more familiar with neural networks research. Figure 1 shows that this is not the case: though the volume of articles that matched the query varies from year to year, for most years the number of risky uses of random seeds is similar to the number of safe uses. This suggests that NLP researchers continue to have trouble distinguishing safe from risky uses of neural network random seeds.

4 Discussion

We have seen that risky uses of neural network random seeds – using only a fixed seed or generating performance distributions for model comparisons by varying only random seeds – are still widespread within the NLP community. The analysis in section 3 is probably a conservative estimate of the problem. The query used in the analysis matched articles only if they had the explicit phrases “neural network” and “random seed” both within the article. That means the search did not return articles on neural networks where no “random seed” was mentioned, yet in such cases it is likely that a single fixed seed was used. Therefore the proportion of fixed seed papers in our sample is likely an underestimate of the proportion in the true population[3].

How do we move the NLP community away from risky uses of neural network random seeds? Hopefully, this article can help to start the necessary conversations, but clearly it is not an endpoint in and of itself. Part of the responsibility must fall on mentors in the NLP community, such as university faculty and industry research leads, to ensure that they are training their mentees about these topics. Part of the responsibility will fall on reviewers of NLP articles, who can identify misuses of neural network random seeds and flag them for revision. And of course part of the responsibility falls on NLP authors themselves to make sure they understand the nuances of neural network hyperparameters like random seeds and the ways in which they should and should not be used.

5 Conclusion

This position piece has introduced a simple taxonomy of common uses for neural network random seeds in the NLP literature, describing three safe uses (model selection, ensemble creation, and sensitivity analysis) and two risky uses (single fixed seed and varying only the random seed to generate distributions for performance comparison). An analysis of 85 articles from the ACL Anthology showed that more than half of these recent NLP articles include risky uses of neural network random seeds. Hopefully, highlighting this issue can help the NLP community to improve our mentorship and training and move away from risky uses of neural network random seeds in the future.

[3]This query also can’t address another interesting issue that is out of scope for the current article: quantifying how many articles don’t tune any hyperparameters at all.

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Jari Björne and Tapio Salakoski. 2018. Biomedical event extraction using convolutional neural networks and dependency parsing. In *Proceedings of the BioNLP 2018 workshop*, pages 98–108, Melbourne, Australia. Association for Computational Linguistics.

Matt Crane. 2018. Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Hang Hua, Xingjian Li, Dejing Dou, Chengzhong Xu, and Jiebo Luo. 2021. Noise stability regularization for improving BERT fine-tuning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3229–3241, Online. Association for Computational Linguistics.

Pankaj Kanwar, Reed Wanderman-Milne, and Duncan Riach. 2021. RFC: Random numbers in TensorFlow 2.0 by wangpengmit - pull request #38 - tensorflow/community.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Minh Le and Antske Fokkens. 2018. Neural models of selectional preferences for implicit semantic role labeling. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yan-Jie Lin, Hong-Jie Dai, You-Chen Zhang, Chung-Yang Wu, Yu-Cheng Chang, Pin-Jou Lu, Chih-Jen Huang, Yu-Tsang Wang, Hui-Min Hsieh, Kun-San Chao, Tsang-Wu Liu, I-Shou Chang, Yi-Hsin Connie Yang, Ti-Hao Wang, Ko-Jiunn Liu, Li-Tzong Chen, and Sheau-Fang Yang. 2020. Cancer registry information extraction via transfer learning. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 201–208, Online. Association for Computational Linguistics.

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.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Garrett Nicolai, Bradley Hauer, Mohammad Motallebi, Saeed Najafi, and Grzegorz Kondrak. 2017. If you can't beat them, join them: the University of Alberta system description. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 79–84, Vancouver. Association for Computational Linguistics.

Michael P. Perrone and Leon N. Cooper. 1995. *When networks disagree: Ensemble methods for hybrid neural networks*, pages 342–358. World Scientific.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Julian Risch and Ralf Krestel. 2020. Bagging BERT models for robust aggression identification. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 55–61, Marseille, France. European Language Resources Association (ELRA).

Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Two Sigma. 2017. A workaround for non-determinism in TensorFlow.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. 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)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Liner Yang and Chencheng Wang. 2019. The BLCU system in the BEA 2019 shared task. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 197–206, Florence, Italy. Association for Computational Linguistics.