Learning Data Augmentation Schedules for Natural Language Processing

Daphné Chopard and **Matthias S. Treder** and **Irena Spasić**
School of Computer Science and Informatics, Cardiff University
Queen's Buildings, 5 The Parade
Cardiff, CF24 3AA, United Kingdom
{ChopardDA,TrederM,SpasicI}@cardiff.ac.uk

Abstract

Despite its proven efficiency in other fields, data augmentation is less popular in the context of natural language processing (NLP) due to its complexity and limited results. A recent study (Longpre et al., 2020) showed for example that task-agnostic data augmentations fail to consistently boost the performance of pretrained transformers even in low data regimes. In this paper, we investigate whether data-driven augmentation scheduling and the integration of a wider set of transformations can lead to improved performance where fixed and limited policies were unsuccessful. Our results suggest that, while this approach can help train better models in some settings, the improvements are unsubstantial. This negative result is meant to help researchers better understand the limitations of data augmentation for NLP.

1 Introduction

In recent years, data augmentation has become an integral part of many successful deep learning systems, especially in the fields of computer vision and speech processing (Krizhevsky et al., 2012; Jaitly and Hinton, 2013; Hannun et al., 2014; Ko et al., 2015). Traditionally, data augmentation approaches take the form of label-preserving transforms that can be applied to the training datasets to expand their size and diversity. The idea of generating synthetic samples that share the same underlying distribution as the original data is often considered a pragmatic solution to the shortage of annotated data, and has been shown to reduce overfitting and improve generalisation performance (Shorten and Khoshgoftaar, 2019). However, despite a sound theoretical foundation (Dao et al., 2019), this paradigm has not yet translated to consistent and substantial improvement in natural language processing (NLP) (Longpre et al., 2020).

The inability of NLP models to consistently benefit from data augmentations can be partially attributed to the general difficulty of finding a good combination of transforms and determining their respective set of optimal hyperparameters (Ratner et al., 2017), a problem that is exacerbated in the context of text data. Indeed, since the complexity of language makes text highly sensitive to any transformations, data augmentations are often tailored to a specific task or dataset and are only shown to be successful in specific settings.

In this work, we investigate whether automatically searching for an optimal augmentation *schedule* from a wide range of transformations can alleviate some of the shortcomings encountered when applying data augmentations to NLP. This endeavour follows the recent success of automated augmentation strategies in computer vision (Cubuk et al., 2019; Ho et al., 2019; Cubuk et al., 2020). In doing so, we extend the efforts to understand the limits of data augmentation in NLP.

2 Related Work

Although there exist recent surveys (Feng et al., 2021; Shorten et al., 2021) that offer a comprehensive review of related work, they do not provide a comparative analysis of the different data augmentation approaches and of their effect on learning performance. In general, the literature lacks general comparative studies that encompass the variety of tasks and datasets in NLP. Indeed, most of the existing text data augmentation studies either focus on a single approach in a specific setting or compare a small set of techniques on a specific task and dataset (Giridhara. et al., 2019; Marivate and Sefara, 2020). In addition, many of these comparisons have been conducted before the widespread adoption of contextualized representations.

Recently, Longpre et al. (2020) showed that, despite careful calibration, data augmentation yielded little to no improvement when applied to pretrained transformers even in low data regimes. While their comparative analysis is conducted on various classification tasks, it focuses on a limited set of aug-

Proceedings of the Second Workshop on Insights from Negative Results in NLP, pages 89–102
November 10, 2021. ©2021 Association for Computational Linguistics
ISBN 978-1-954085-93-0

mentation strategies that are applied independently and whose hyperparameters are optimized via a random search.

To further assess the effectiveness of data augmentation on pretrained transformers, we investigate in this work the paradigm of learning data augmentation strategies from data in the context NLP. The idea is to leverage the training data to automatically discover an optimal combination of augmentations and their hyperparameters at each epoch of the fine-tuning. First, we define a search space that consists of a variety of transformations before relying on the training data to learn an optimal schedule of probabilities and magnitudes for each augmentation. This schedule is later used to boost performance during fine-tuning.

In the following section, we present a variety of data augmentations and, for each category of transformations, we highlight a subset of augmentations that is representative of the category and that will constitute our search space.

3 Data Augmentation in NLP

In this study, we focus on *transformative* methods which apply a label-preserving transformation to existing data—rather than generative methods which create entirely new instances using generative models. Indeed their simplicity of use and their low computational cost make them good candidates for a wide deployment (Xu et al., 2016). In the last decade, we witnessed a widespread adoption of continuous vector representations of words, which can be easily fed to deep neural network architectures. As a result, transforms have been developed not only at the lexical level (i.e., words) but also at the latent semantic level (i.e., embeddings). This distinction is emphasized throughout this section.

Word Replacement A commonly used form of data augmentation in NLP is word replacement. At the lexical level, the most common approach consists in randomly replacing words with their synonyms (Zhang et al., 2015; Mueller and Thyagarajan, 2016; Vosoughi et al., 2016). Some variants include replacement based on other lexical relationships such as hypernymy (Navigli and Velardi, 2003) or simply words from the vocabulary (Wang et al., 2018; Cheng et al., 2018). Another popular approach consists in using a language model (LM) for replacement (Kolomiyets et al., 2011; Fadaee et al., 2017; Ratner et al., 2017). Because these transformations do not ensure the preservation of the sample class, Kobayashi (2018) suggested conditioning a bidirectional LM on the labels, an idea later revisited by Wu et al. (2019) who replaced the LM with a conditional BERT (Bidirectional Encoder Representations from Transformers).

At the latent semantic level, word replacement amounts to randomly replacing its embedding with some other vector. For instance, Wang and Yang (2015) choose the k-nearest-neighbour in the embedding vocabulary as a replacement for each word. Similar strategies were later adopted by Vijayaraghavan et al. (2016) and Zhang et al. (2019).

In this study, as replacement methods, we select both **synonym** and **hypernym** replacement as well as **contextual augmentation** at the word level and **nearest neighbour** at the embedding level.

Noising A simple yet effective form of augmentation that is often applied to images and audio samples is data noising. Not surprisingly, this type of data augmentation can also be found in NLP despite the discrete nature of text data.

In its simplest form, data noising when applied to text consists in inserting, swapping or deleting words at random (Wei and Zou, 2019). More generally, the process of ignoring a fraction of the input words is often referred to as *word dropout* (Iyyer et al., 2015) and can take multiple forms (Dai and Le, 2015; Zhang et al., 2016; Bowman et al., 2016; Xie et al., 2017; Li et al., 2017).

Sometimes, word replacement can also be thought of as a form of noising. For instance, replacing words at random with other words from the vocabulary introduces noise into the data (Xie et al., 2017; Cheng et al., 2018).

In contrast, at the distributed representation level, this type of augmentation often takes the form of added noise to the embeddings. Possible noising schemes include Gaussian noise (Kumar et al., 2016; Cheng et al., 2018), uniform noise (Kim et al., 2019), Bernoulli noise and adversarial noise (Zhang and Yang, 2018). Typically, noising is applied to every word embedding, but it can also be applied only to selected ones (Kim et al., 2019). Alternatively, as with word dropout, noise can be incorporated into the training by discarding, across all words, some embedding dimensions with a predefined probability (Dai and Le, 2015).

Noising strategies used in this study are based on **random deletion**, **random swap** and **random insertion** at the sentence level as well as **Gaussian noise** and **uniform noise** at the feature level.

Back-translation Back-translation provides a way for neural machine translation (NMT) systems to leverage monolingual data to increase the amount of parallel data (Sennrich et al., 2016; Edunov et al., 2018; Fadaee and Monz, 2018). Similarly, back-translation can be applied twice in a row (i.e., from English to another language and back to English) to generate new data points without the need for parallel corpora and can therefore find applications as a task-agnostic augmentation in other tasks such as text classification (Luque and Pérez, 2018; Aroyehun and Gelbukh, 2018), paraphrase generation (Mallinson et al., 2017) and question answering (Yu et al., 2018).

Here, we consider a wide range of intermediate languages to include **back-translation** into our search space

4 Automated Data Augmentation in NLP

In NLP, little effort has been put into developing strategies that can, given a task and a dataset, learn an optimal subset of data augmentation operations and their hyperparameters (Shorten et al., 2021). Yet, this idea has been very successful in computer vision (Cubuk et al., 2019; Ho et al., 2019; Cubuk et al., 2020). For instance, in the context of image classification, Cubuk et al. (2019) have proposed AutoAugment a procedure that automatically searches for optimal augmentation policies using reinforcement learning. Later, Population-Based Augmentation (PBA) (Ho et al., 2019)—an algorithm that views the data augmentation selection and calibration problem as a hyperparameter search and can thus leverage the Population-Based Training (PBT) method (Jaderberg et al., 2017) to find an optimal transformation schedule in an efficient way—was introduced as a more cost-effective yet competitive alternative to AutoAugment. Finally, Cubuk et al. (2020) introduced the RandAugment method which tackles some of the issues arising from these previous works.

In this study, we adapt the PBA framework to NLP in an attempt to learn an optimal schedule of data augmentation operations with optimized hyperparameters.

4.1 Population-Based Augmentation

Given a hyperparameter search space that consists of data augmentation operations along with their probability level (i.e., likelihood of being applied) and their magnitude level (i.e., strength with which they are applied), PBA works as follows: during a pre-defined number of epochs, k child models of identical architecture are trained in parallel for the task at hand on a given dataset. Periodically, the training is interrupted, all models are evaluated on a validation set and an "exploit-and-explore" procedure takes place. First, the worst-performing models (bottom 25%) copy the weights and hyperparameters of the best-performing models (top 25%) (**exploit**), then the hyperparameters are either slightly perturbed or uniformly resampled from all possible values (**explore**). At that point, training can continue. At the end of the training, a data augmentation policy schedule is extracted from the hyperparameters of the best performing child model. The obtained schedule can then be used to train from scratch a different model on the same task and the same dataset.

5 Experiments

Our implementation builds on the original PBA codes (Ho et al., 2019). We make the NLP adaptation of this framework publicly available[1].

5.1 Datasets

Two of the datasets suggested by Longpre et al. (2020) in their comparative analysis are used to conduct our experiments: SST-2 (Socher et al., 2013) and MNLI (Williams et al., 2018). The former is a corpus of movie reviews used for sentiment analysis (single sentence, binary classification), whereas the latter is a natural language inference corpus (two sentences, multi-class classification). Accuracy and mis-matched accuracy respectively are used for evaluation. We take $N = \{1500, 2000, 3000, 9000\}$ samples from the original train sets to constitute our train-validation sets. As test sets, we use the *full* original validation sets which—unlike the original test sets—contain publicly available ground-truths.

5.2 Hyperparameter Space

The hyperparameter space consists of 10 data augmentation operations highlighted in the previous section with associated probability and magnitude. For most replacement methods, the magnitude level can be thought of as a percentage of tokens on which the transformation is applied and, for noising transforms, it corresponds to the amount of noising. In the context of back-translation, however,

[1] https://github.com/chopardda/LDAS-NLP

	ACCURACY [%]			
	$N = 1500$	2000	3000	FULL
NO AUGM	88.22 ±0.69	88.11 ±0.41	88.76 ±0.46	90.49 ±0.49
SCHEDULE	**88.64** ±0.47	**88.60** ±0.68	**89.56** ±0.40	**90.91** ±0.43
DIFFERENCE	*+0.42*	*+0.49*	*+0.80*	*+0.42*

(a) SST-2 test dataset.

	MIS-MATCHED ACCURACY [%]			
	$N = 1500$	2000	3000	FULL
NO AUGM	**65.73** ±1.12	**67.60** ±2.21	**69.67** ±0.79	**74.26** ±0.35
SCHEDULE	64.78 ±1.20	66.21 ±0.77	68.71 ±0.49	73.95 ±0.47
DIFFERENCE	*-0.95*	*-1.39*	*-0.96*	*-0.31*

(b) MNLI test dataset.

Table 1: Performance on SST-2 and MNLI. The model is trained 10 times independently either without augmentation or with the augmentation schedule yielded by the search. Since a single search is conducted per value of N, the reported standard deviation measures the robustness of the training procedure for a single schedule. (For more details about the robustness of the augmentation search, see Section 6.1.)

magnitude relates to the quality of the translation according to BLEU-3 scores (Aiken, 2019). Details concerning the implementation of these augmentations as well as how exactly magnitude is defined for each one of them can be found in Appendix A.1.

5.3 Search

The search is conducted on 48 epochs using around 20% of the N data points for training and the rest for validation. Both the child models and the final model follow the original uncased BERT base architecture suggested by Devlin et al. (2019). The learning rate is chosen so as to slow down the fine-tuning without affecting the performance. Further details are provided in Appendix A.2.

6 Results

The main results can be found in Table 1a and Table 1b. The first row corresponds to the performance on the test set when training the model on all N samples without any augmentation. The second row in the table contains the result of the evaluation on the test set of the model trained on all N samples using the discovered schedules (i.e., for each data size, the schedule of the child model with the highest validation accuracy at the end of the search is used to train the final model). The same slowed-down learning rate and the same extended number of epochs are used across all experiments to allow for a fair comparison.

Overall, the improvements yielded by the optimized data augmentation schedules are inconsistent and unsubstantial (below 0.8%). Even though the incorporation of transforms has a small positive impact on the SST-2 dataset, it has the opposite effect on MNLI (i.e., the scores plummet by as much as 1.39%). A possible reason for these poor results might be due to the difference in settings between our experiments and the ones in the PBA study. Indeed, our search is conducted on 48 epochs as opposed to the 160 to 200 epochs suggested for image classification tasks and the exploit-and-explore procedure takes place after each epoch rather than after every 3 epochs. In addition, the size of the training datasets is very different. The size of our largest experiment is roughly the same as the size of the smallest dataset in (Ho et al., 2019).

Surprisingly, our data-driven search seems unable to reproduce the performance boost reported by Longpre et al. (2020) on the MNLI dataset, even though the augmentations considered in their work are part of our search space. This might be explained by the way augmentations are applied. In our study, we transform each training sample by applying up to 2 transformations in a row with probability p and magnitude m. In contrast, Longpre et al. add $N \times \tau$ augmented samples to the training set with $\tau \in \{0.5, 1, 1.5, 2\}$, meaning that the original examples are provided along with their augmented counterparts at each iteration.

6.1 Search Robustness

Given the stochastic nature of the searching process, the discovered schedule is bound to differ from one run to another. So far, we have run a single search for each experiment setting. In this section, we investigate whether the limited effect of automatic augmentation on the model performance may be caused by the stochasticity of the search. To that end, we run 10 independent searches on the SST-2 dataset with N=1500 and use each of the 10 discovered schedules to train a separate model. All the network hyperparameters are kept the same as in the previous section. Overall, the standard deviation over 10 independent schedules is 0.55%, which indicates that the performance of the training is robust across searches. Thus, the poor results observed in the previous section cannot be explained by the variability of schedules.

However, a closer look at these 10 individual schedules reveals that the chosen augmentation hyperparameters are very different from one run to another and that the search does not seems to favour any particular set of augmentation transforms. This may indicate that, in this setting, data augmentation acts more as a regulariser rather than a way to learn invariance properties and that, as a result, any kind of augmentation transform has a similar effect on performance. In view of these findings, it would be interesting to explore whether relying on a greater number of child models during the search could potentially yield less disparate schedules and improve the overall quality of the search. For the interested reader, the various schedules are displayed in Appendix A.5.

6.2 Validation size

As mentioned earlier, the limited impact of automatic data augmentation scheduling in our settings might be due to the small number of samples available for each experiment. In particular, one of the drawbacks of PBA is that a large portion of training data (approximately 80% as suggested by Ho et al. (2019)) has to be set aside to form a validation set that is used during the search to find optimal hyperparameters. For example, at N=1500 only 250 examples are used to learn the network weights during the search while the remaining 1250 samples are used for hyperparameter selection. As a result of this discrepancy, the selected data augmentation might be relevant when only 250 data points are available for training but less effective when learning with 1500 samples as is ultimately the case. In this section, we investigate whether the poor results observed in Table 1 can be attributed to the ratio chosen to split the available data into a train and a validation set. To that end, we run the search on the SST-2 dataset using different ratios to divide the N=1500 samples at hand. The results reported in Table 2 suggest that using different proportions of train and validation examples does not affect the effectiveness of the augmentation schedule in this setting. In fact, the performance remains the same even though the model is trained with schedules that were optimized using very different split ratios. This might be explained by the fact that both the train and the validation sets are too small to find optimal augmentation hyperparameters irrespective of the chosen split ratio. Alternatively, it is possible that the chosen dataset can simply not benefit from augmentation because of its nature. To verify this hypothesis, it would be interesting to extend this analysis to a wider range of datasets, including actual low-resource datasets.

TRAIN/VAL	250/1250	750/750	1250/250
ACCURACY	**88.64** ±0.47	**88.64** ±0.62	**88.76** ±0.59

Table 2: Performance on the SST-2 test set of the model trained on N=1500 samples with the schedule discovered using different proportions of validation and training sets. For each split ratio, the model is trained 10 times using the schedule yielded by a single search. The mean accuracy and standard deviation are reported.

7 Conclusion

The results suggest that augmentation schedules and data-driven parameter search do not provide a consistent and straightforward way to improve the performance of NLP models that use pretrained transformers. There are a few possible explanations for this phenomenon. First, the overall setup of the PBA approach (e.g., the need for large validation sets) might not be well suited for low-data regimes in NLP. A second but more likely reason is that transformers are already pre-trained on huge datasets and their representations may already be invariant to many of the transformations that are encoded into the data augmentation. A systematic investigation into the latter hypothesis is required, which, if proven, would show that data augmentation may be redundant when opting to use transformers to implement NLP solutions. A final reason might be that the search space we consider only contains transformative data augmentation techniques and omits generative ones, even though the latter have started to show some promising results.

References

Milam Aiken. 2019. An updated evaluation of google translate accuracy. Studies in linguistics and literature, 3(3):253–260.

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), pages 90–97.

Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL).

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1756–1766.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2019. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 113–123.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 702–703.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 3079–3087. Curran Associates, Inc.

David Dale. 2020. GitHub repository.

Tri Dao, Albert Gu, Alexander Ratner, Virginia Smith, Chris De Sa, and Christopher Ré. 2019. A kernel theory of modern data augmentation. In International Conference on Machine Learning, pages 1528–1537. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT (1).

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 489–500.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 567–573.

Marzieh Fadaee and Christof Monz. 2018. Back-translation sampling by targeting difficult words in neural machine translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 436–446.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. Findings of ACL.

Praveen Kumar Badimala Giridhara., Chinmaya Mishra., Reddy Kumar Modam Venkataramana., Syed Saqib Bukhari., and Andreas Dengel. 2019. A study of various text augmentation techniques for relation classification in free text. In Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,, pages 360–367. INSTICC, SciTePress.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In International Conference on Machine Learning, pages 2731–2741.

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), volume 1, pages 1681–1691.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. arXiv preprint arXiv:1711.09846.

Navdeep Jaitly and Geoffrey E Hinton. 2013. Vocal tract length perturbation (vtlp) improves speech recognition. In Proc. ICML Workshop on Deep Learning for Audio, Speech and Language, volume 117.

Hwa-Yeon Kim, Yoon-Hyung Roh, and Young-Gil Kim. 2019. Data augmentation by data noising for open-vocabulary slots in spoken language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 97–102.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In Sixteenth Annual Conference of the International Speech Communication Association.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 452–457.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pages 271–276. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In International conference on machine learning, pages 1378–1387.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 21–27.

Shayne Longpre, Yu Wang, and Chris DuBois. 2020. How effective is task-agnostic data augmentation for pretrained transformers? In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 4401–4411.

Franco M Luque and Juan Manuel Pérez. 2018. Atalaya at tass 2018: Sentiment analysis with tweet embeddings and data augmentation. In TASS@ SEPLN, pages 29–35.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 881–893.

Vukosi Marivate and Tshephisho Sefara. 2020. Improving short text classification through global augmentation methods. In International Cross-Domain Conference for Machine Learning and Knowledge Extraction, pages 385–399. Springer.

George A Miller. 1998. WordNet: An electronic lexical database. MIT press.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In Thirtieth AAAI Conference on Artificial Intelligence.

Roberto Navigli and Paola Velardi. 2003. An analysis of ontology-based query expansion strategies. In International Workshop & Tutorial on Adaptive Text Extraction and Mining held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practice of, page 42.

Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation. In Advances in neural information processing systems, pages 3236–3246.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):60.

Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. 2021. Text data augmentation for deep learning. Journal of Big Data, 8(1):1–34.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing, pages 1631–1642.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. DeepStance at SemEval-2016 task 6: Detecting stance in tweets using character and word-level CNNs. In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pages 413–419, San Diego, California. Association for Computational Linguistics.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2Vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 1041–1044. ACM.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In Proceedings of the 2015 Conference on

Empirical Methods in Natural Language Processing, pages 2557–2563.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 856–861.

Jason 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), pages 6383–6389.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional BERT contextual augmentation. In International Conference on Computational Science, pages 84–95. Springer.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1461–1470.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In International Conference on Learning Representations.

Dongxu Zhang, Tianyi Luo, and Dong Wang. 2016. Learning from LDA using deep neural networks. In Natural Language Understanding and Intelligent Applications, pages 657–664. Springer.

Dongxu Zhang and Zhichao Yang. 2018. Word embedding perturbation for sentence classification. arXiv preprint arXiv:1804.08166.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. 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 1031–1040.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In Advances in neural information processing systems, page 649–657.

A Appendix

A.1 Hyperparameter Search Space

As a reminder we consider the following 11 data augmentation methods:

- Synonym replacement
- Hypernym replacement
- Nearest neighbour
- Contextual augmentation
- Random insertion of a vocabulary word
- Random insertion of synonym
- Random deletion
- Random swap
- Gaussian noising
- Uniform noising
- Back-translation

The search space consists of augmentation operations with associated probability and magnitude. More specifically, this can be represented as a vector of 11 tuples (o_i, p_i, m_i) (i.e., one tuple for each transform). During the training, up to two data augmentation operations o_i are drawn uniformly at random for each training sample and applied with probability p_i and magnitude m_i. As suggested by Ho et al. (2019), we set the number of operations to $0, 1$ and 2 with probabilities $0.2, 0.3$ and 0.5 respectively. The operations are applied in the same order in which they are drawn. However, the two embedding-level noising operations (i.e., Gaussian noising and uniform noising) are always applied after the other augmentations since they must be applied in the middle of the graph, after the representation layers, whereas the other augmentations are applied directly on the input of the neural network.

To allow for a smooth parametrisation of the search space with large coverage, probabilities and magnitudes can take any values between 0 and 1: $p, m \in [0, 1]$. This is different from the original PBA algorithm where the parameters are limited to discrete values. The magnitude level, which represents the intensity with which each operation should be applied, is scaled down differently to fit the different operations. Maximal magnitude values are chosen so as to allow for a wide enough array of impactful values and their specific values for each augmentation are indicated in the corresponding paragraphs.

All transformation operations are detailed below, including implementation, and the ones that are applied directly on the input rather than on the embeddings are illustrated in Table 3.

Synonym Replacement The implementation follows the one suggested by the authors of the easy data augmentation (EDA) techniques (Wei and Zou, 2019) and uses the provided codes. First, the number of words that are replaced with one of their synonyms is determined as $n_i = \lfloor \hat{m}_i * |s_i| \rfloor$, with $\hat{m}$ the magnitude level scaled down between 0 and 0.25. Then stop-words are removed from the sample. While the number of words replaced is lower than n_i, one word is selected uniformly at random among the words that have not been replaced yet and is replaced with one of its synonyms. Synonyms are retrieved with WordNet (Miller, 1998). Note that since many words have multiple meanings, it is not rare that the chosen synonym carries a different meaning than the original word.

Hypernym Replacement The process for hypernym replacement is identical to that of synonym replacement in all respect except that hypernyms instead of synonyms are extracted using WordNet.

Nearest Neighbour At the beginning of each search (and at the beginning of the final training), we feed every training sample into the pretrained BERT and use the contextualized representation of each token to build a k-d tree. Note that this process is one-time only and is tailored to the train set. When applied to sample s_i, the *nearest-neighbour* operation first tokenizes the sample into WordPiece tokens using the BERT tokenizer before computing the number of tokens that will be replaced as follows: $n_i = \lfloor \hat{m}_i \cdot |s_i| \rceil$. Here, $|s_i|$ corresponds to the number of WordPiece tokens in sample s_i and $\hat{m}_i = 0.25 \cdot m_i$ is the scaled-down level of magnitude, which has a maximum value of 0.25 so that at most 25% of the tokens are replaced. Then, n_i WordPiece tokens at drawn uniformly at random. For each one of them, the 10 tokens with the nearest embeddings (in the context of sample s_i) are retrieved and one of them is selected for replacement using a geometric distribution with parameter $q = 0.5$. A geometric distribution ensures that the nearest neighbours have a higher chance to be selected as a replacement than the more distant ones. The implementation is based on the one provided by Dale (2020).

Contextual Augmentation The *contextual augmentation* transform replaces words by these pre-

	original sample ($m = 0$)	$m = 0.5$	$m = 1$
Synonym Replacement	a pretentious and ultimately empty examination of a sick and evil woman	a pretentious and ultimately empty examination of a sick and **malefic** woman	a **ostentatious** and ultimately empty **interrogatory** of a **retch** and evil woman
Hypernym Replacement	just another disjointed , fairly predictable psychological thriller	just another disjointed , fairly predictable psychological **adventure story**	just another **part** , fairly predictable psychological **heroic tale**
Nearest Neighbour	tormented by his heritage , using his storytelling ability to honor the many faceless victims	tormented by his heritage , using his storytelling ability to honor the many faceless **emotions**	tormented by his **fate** , using his **cinematic** ability to honor the many **unexplainable** victims
Contextual Augmentation	small - budget	**thin** - budget	**barely running** budget
Random Insertion (vocabulary)	actually manages to bring something new into the mix	actually manages *johns* to bring something new into the mix	*lest* actually manages to *goaltender* bring something new into the mix
Random Insertion (synonym)	is a gorgeous film - vivid with color , music and life	is a gorgeous film *brilliant* - vivid with color , music and life	*liveliness* is a gorgeous film - *medicine* vivid with color *exist* , music and life
Random Deletion	an adventurous young talent who finds his inspiration on the fringes of the american underground	an adventurous talent who finds his on the fringes of american underground	an young finds his on the of underground
Random Swap	the movie is widely seen and debated with appropriate ferocity and thoughtfulness	the movie seen widely is and debated with appropriate ferocity and thoughtfulness	movie the is widely seen ferocity thoughtfulness with appropriate and and debated
Back-translation	does n't know what it wants to be	Do not know what he wants to be	I don't know what you want to be

Table 3: Overview of the data augmentation transforms from our search space that operate directly on the input. This shows the outcome when the transformations are applied on samples from the SST-2 dataset with three different magnitude levels $m = 0, 0.5, 1$. Where relevant, tokens that have been replaced are highlighted in bold. In addition, newly inserted tokens are italicized whereas tokens that have changed places are underlined.

dicted by a language model (LM) conditioned on the labels. In this work, we use the implementation provided by Wu et al. (2019) which uses BERT as a conditional masked language model. At the beginning of the search, the model is fine-tuned on the training data on a task that applies extra label-conditional constraint to the traditional masked language objective. Once fine-tuned, the model can be used to infer masked words given a label. When applied to sample s_i, this operation replaces $n_i = \lfloor m_i \cdot |s_i| \rceil$ of the tokens with a mask, where $|s_i|$ is the number of tokens in the sample after applying the BERT tokenizer. Then, along with its label, the masked sample is fed to the fine-tuned conditional model which infers a vocabulary word for each of the masked tokens. These predictions are used as a replacement.

Random Insertion When applied to a sample s_i, the *random insertion* operation randomly adds a token to the sample. The number of tokens to insert n_i is set to a fraction of the length of s_i, namely $n_i = \lfloor \hat{m}_i \cdot |s_i| \rceil$, where $|s_i|$ is the number of tokens in s_i and $\hat{m}_i = 0.25 \cdot m_i$ is the scaled-down magnitude which ensures the number of inserted tokens does not exceed 25% of the original number of tokens and, by extension, that the new sample contains at most 20% of randomly inserted tokens. We include two independent variants: each inserted token is either a synonym of one of the tokens (selected uniformly at random) in s_i as suggested by Wei and Zou (2019) as part of their EDA techniques or is sampled uniformly at random from a subset of the BERT vocabulary. Note that we only consider words between index 1996 and 29611 of the vocabulary to exclude special and unused tokens as well as punctuation, digits and tokens with non-English characters. We also ignore tokens that start with "##" the special characters used to indicate a trailing WordPiece token. The position for the insertion of the new token in s_i is chosen uniformly at a random. The implementation uses the codes provided by Wei and Zou (2019).

Random Deletion The *random deletion* operation removes a fraction of the tokens from the sample. Each token is discarded with probability q_i, where $q_i = \hat{m}_i = 0.25 \cdot m_i$ which is the magnitude level scaled down between 0 and 0.25 to guarantee that at most half of the tokens are removed. This allows a wide range of values around the original intensity parameter of 0.1 suggested by Wei and Zou (2019) whose implementation we use.

Random Swap This augmentation swaps any two words from the sample s_i at random n_i times in a row, where $n_i = \lfloor \hat{m}_i * |s_i| \rfloor$ and $|s_i|$ is the number of tokens in sample s_i. The magnitude parameter m_i is scaled down to have a maximum value of 0.25 to ensure that at most 50% of the words are swapped. Once again, we use the implementation provided by Wei and Zou (2019).

Gaussian Noising The *Gaussian noising* operation is not applied on the input sequences but rather directly on the contextualized word representations. Let w_{ij} be the embedding of word j in sample s_i. Then, each embedding in the sample is transformed as follows:

$$\hat{w}_{ij} = w_{ij} + e_j, \quad e_{jk} \sim \mathcal{N}(0, \sigma^2) \quad , \qquad (1)$$

where $\sigma = m_i$ and e_j is a vector of the length of d (embedding dimension) with elements e_{jk} normally distributed with mean 0 and standard deviation m_i.

Uniform Noising Similarly to Gaussian noising, the *uniform noising* operation is applied directly on the contextualized embeddings. More specifically,

$$\hat{w}_{ij} = w_{ij} + e_j, \quad e_{jk} \sim \mathcal{U}(-m_i, m_i) \quad . \qquad (2)$$

Once again, e_{jk} $(0 \leq k < d)$ are the elements of noise vector e_j uniformly distributed over the half-open interval $[-m_i, m_i]$ and d is the dimension of the embeddings.

Back-translation The *back-translation* operation first translates the sample s_i to an intermediate language before translating the intermediate translation back to English. To allow us to incorporate this transform into the search space, we relate the magnitude level with the quality of the translation: when back-translation is applied with a low magnitude, the intermediate language used is one that achieves a high BLEU3 score according to Aiken (2019). Similarly, high magnitude settings back-translate samples through a language with a poor BLEU3 score. Table 4 summarizes the languages that can be chosen for each level of magnitude. To generate translations, we use the python library *Googletrans* which uses the Google Translate Ajax API to make calls.

A.2 Search

At the beginning of training, all probability and magnitude hyperparameters are set to 0. This follows suggestions by Ho et al. (2019) who postulate that little to no augmentation is needed at the beginning of the training, since the model only starts to overfit later on, and the data should increasingly become more diverse throughout the training. Because the complexity of the BERT models lies in the contextual representation layers which are already pre-trained and the task-specific layer that needs to be fine-tuned is rather simple, we keep the architecture identical both for the child models and for the final model. A key difference between applying PBA to image classification and applying PBA to NLP tasks with pre-trained BERT model is that in the former settings models are commonly trained for hundreds of epochs, whereas only two to four epochs of fine-tuning are sufficient in the latter settings. Consequently, the original strategy of running a search for 160 or 200 epochs (depending on the model and dataset) while having exploit-and-explore procedures take place after every 3 epochs is not feasible for NLP tasks with a pretrained BERT model. Hence we modify the learning rate to slow down the fine-tuning process. More specifically, we look through a small grid search for a learning rate that can replicate the performance achieved when using the original parameters (Devlin et al., 2019) but on a larger number

m_i	INTERMEDIATE LANGUAGES
1	Portuguese, Italian, French, Czech, Swedish
2	Dutch, Maltese, Polish, Romanian, Russian
3	Afrikaans, Belarusian, Slovak, Danish, Indonesian
4	German, Albanian, Bulgarian, Japanese, Spanish
5	Chinese, Croatian, Finnish, Latvian, Arabic, Malaysian
6	Greek, Korean, Norwegian, Serbian, Turkish, Welsh
7	Galician, Icelandic, Slovenian, Vietnamese
8	Catalan, Estonian, Filipino, Hungarian, Swahili
9	Irish, Thai, Hebrew, Ukrainian, Persian
10	Lithuanian, Macedonian, Yiddish, Hindi

Table 4: The intermediate translation languages for each magnitude level m_i. They are separated according to inverse BLEU3 scores.

of epochs. Thus, by reducing the learning rate, we find a way to carry out the search over a total of 48 epochs. The search is conducted on 16 child models that are trained in parallel. At the beginning of the search, approximately 80% of the training data are set aside to form the validation set, which will be used to periodically assess the performance of the child models. The remaining training data are used to optimize the networks. After each epoch (instead of 3 originally), the exploit-and-explore procedure takes place where the 4 worst performing (on the validation set) child models copy the weights and the parameters of the 4 best performing child models. At the end of the 48 epochs, the augmentation schedule of the model with the highest performance (on the validation set) is extracted.

A.3 Train

To train the final model, the train and validation data are grouped to form the final training set. Training is conducted using the same learning rate and the same number of epochs as during the search and uses the discovered schedule for augmentation. At the end of the training, the performance of the trained model is evaluated on an independent test set.

A.4 Implementation details

The implementation parameters can be found in Table 5.

A.5 Results and Discussion

Additional results, tables and figures can be found in this section.

A.5.1 Schedule Robustness Experiment

In the main text, we touched upon the fact that the schedules yielded by the same search can vary significantly from one run to another. For illustration, the schedules yielded by 10 independent searches on the SST-2 dataset with N=1500 samples are displayed in Figure 1. To reduce the number of figures, the product of the magnitude and the probability hyperparameters through the epochs is shown for each schedule. This figure shows that the optimal set of hyperparameters varies significantly from one search to another. In Figure 2 the average magnitude and probability parameters over the 10 schedules at each epoch is displayed. These plots allow us to realise that, while the parameters generally increase throughout the epochs, the magnitudes and probabilities of each transform have a similar value. Although some operations (e.g. Random Swap) have slightly higher average parameters than others, we can see that no augmentation transform clearly dominates the others. This could indicate that the role of the augmentation operations in this setting is not the one that is expected. Indeed, it seems that the data augmentation merely acts as a regulariser and do not help the network learn any kind of invariance.

	PARAMETERS [%]		
	EPOCHS	LEARNING RATE	BATCH SIZE
SST-2	48	1E-5	32
MNLI	48	5E-5	32

Table 5: Implementation details of both the child models and the final model (both with and without augmentation). The model used is the uncased BERT base model with a classification layer.

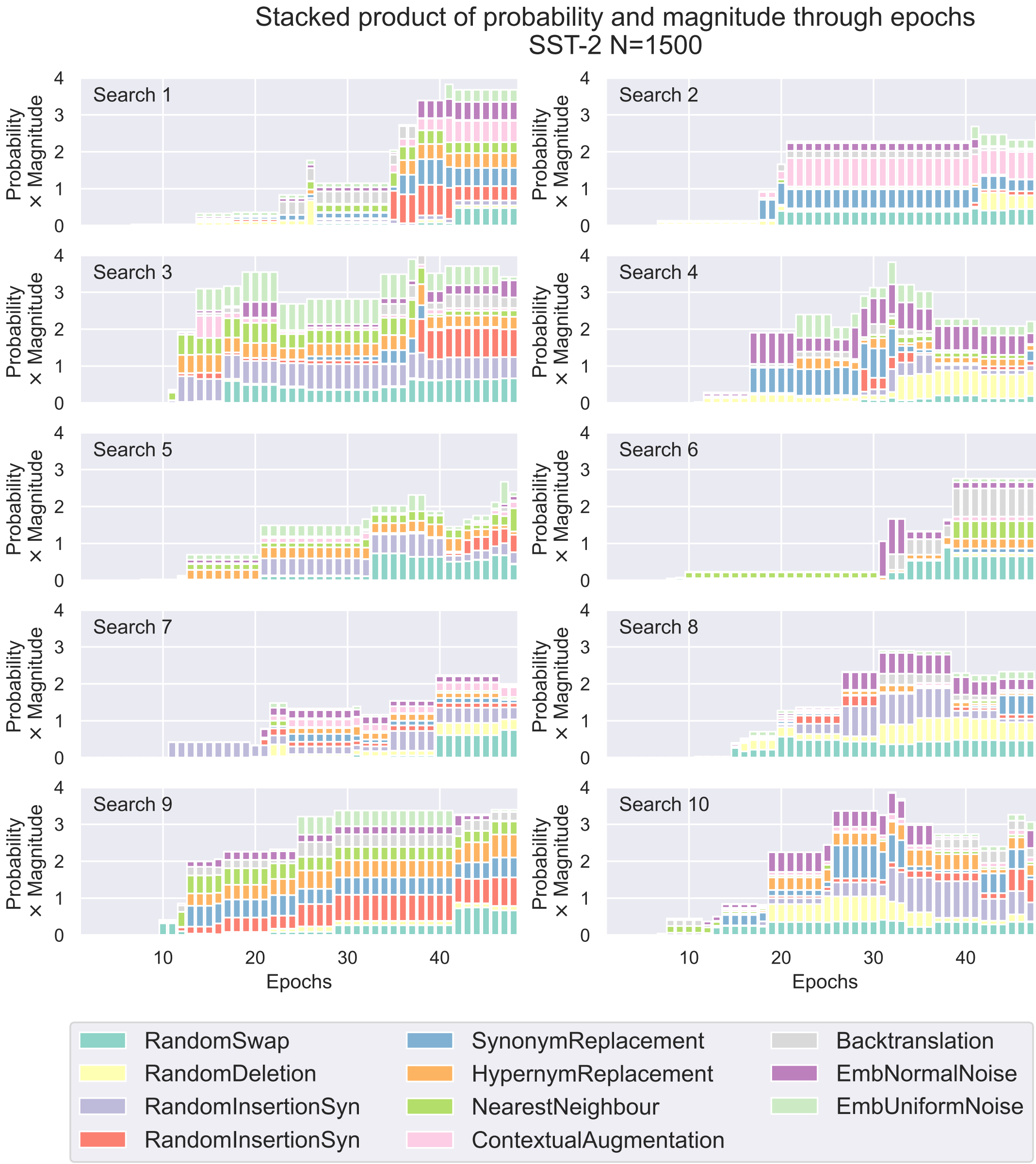
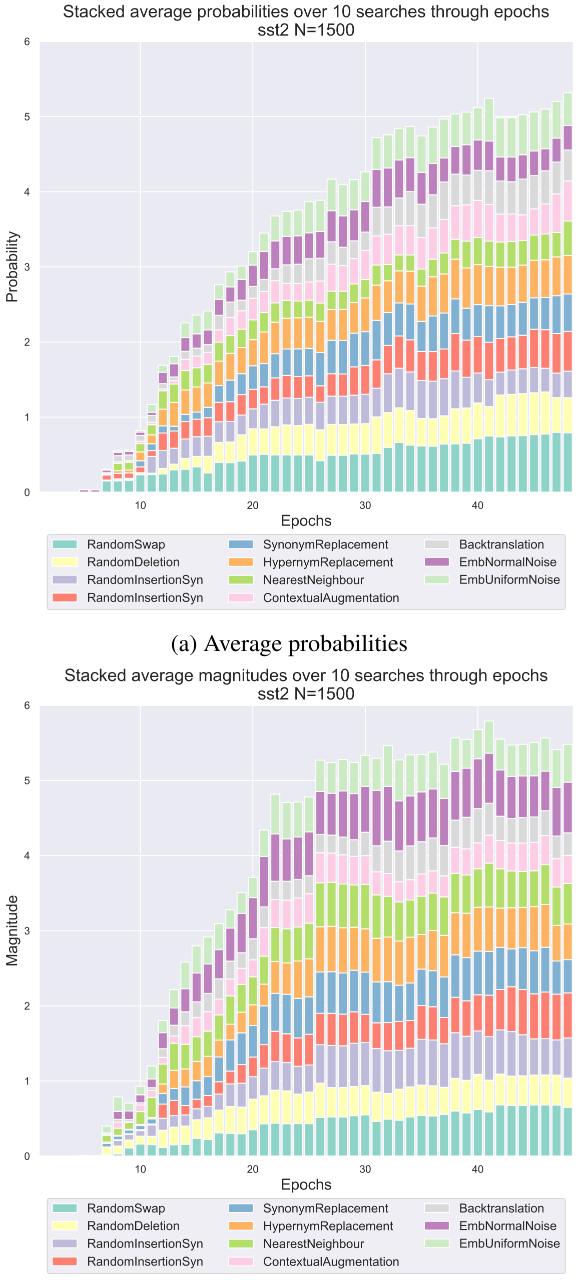


Figure 1: The schedules yielded by 10 independent searches on SST-2 with $N = 1500$ samples (using 250 for training and 1250 for validation during the search). The height of each bar corresponds to the product of the probability and the magnitude parameters at each epoch.



(a) Average probabilities

(b) Average magnitudes

Figure 2: The average probability and magnitude values for the schedules yielded by 10 independent searches on SST-2 with N=1500 samples. The height of each bar corresponds to the average probability and magnitude parameters at each epoch over the 10 schedules.

A.5.2 Validation Size Experiment

In this section, we discuss the split ratio experiment in more detail. Overall, the main idea behind using a large validation set is to choose augmentation hyperparameters that do not overfit the validation samples and thus generalize well to unseen data. However, in our case, since the total number of available samples N is small in all experiments, this implies that the size of the training set will be extremely limited. This might hinder the learning process (with too few training examples it can be difficult to learn the optimal network weights) or make the choice of augmentation hyperparameters irrelevant for larger training sets (there is no guarantee that the augmentation chosen for the small train set with will also help when ultimately training with both the training and the validation set). Thus, there exists a clear trade-off between the size of the two sets: while a large validation set can allow for better optimization of the augmentation hyperparameters, a larger training set allows for better optimization of the network weights which, in turn, has an impact on the quality of the augmentation hyperparameters evaluation.