Simple and Effective Gradient-Based Tuning of Sequence-to-Sequence Models

Jared Lichtarge[1] **Chris Alberti**[1] **Shankar Kumar**[1]

[1]Google Research

Abstract Recent trends towards training ever-larger language models have substantially improved machine learning performance across linguistic tasks. However, the huge cost of training larger models can make tuning them prohibitively expensive, motivating the study of more efficient methods. Gradient-based hyper-parameter optimization offers the capacity to tune hyper-parameters during training, yet has not previously been studied in a sequence-to-sequence setting. We apply a simple and general gradient-based hyperparameter optimization method to sequence-to-sequence tasks for the first time, demonstrating both efficiency and performance gains over strong baselines for both Neural Machine Translation and Natural Language Understanding (NLU) tasks (via T5 pretraining). For translation, we show the method generalizes across language pairs, is more efficient than Bayesian hyper-parameter optimization, and that learned schedules for some hyper-parameters can out-perform even optimal constant-valued tuning. For T5, we show that learning hyper-parameters during pre-training can improve performance across downstream NLU tasks. When learning multiple hyper-parameters concurrently, we show that the global learning rate can follow a schedule over training that improves performance and is not explainable by the 'short-horizon bias' of greedy methods (Wu et al., 2018). We release the code used to facilitate further research.

1 Introduction

Finding good hyper-parameter values is critical to achieving good performance across machine learning domains; this has inspired much work into hyper-parameter optimization (HPO) (see Feurer and Hutter (2019)). Traditionally popular HPO methods require running many trials of hyperparameter sets in parallel or sequential training runs (Bengio, 2012; Snoek et al., 2012; Li et al., 2016). These methods become infeasible as the cost of individual runs increases. This difficulty is exacerbated by recent trends towards larger models (Devlin et al., 2019; Brown et al., 2020; Adiwardana et al., 2020; Chowdhery et al., 2022), which have come to dominate progress on linguistic tasks, yet are only sparsely or indirectly tuned.

The growing field of gradient-based HPO methods offers an alternative to conventional HPO by allowing hyper-parameters to be learned based on a loss function, which can greatly improve over the efficiency of comparing constant values tuned across multiple runs (Maclaurin et al., 2015; Pedregosa, 2016; Franceschi et al., 2018)[1]. Many gradient-based methods additionally allow hyper-parameters to dynamically vary in value over a training run as opposed to only taking static values[2]. However, most prior work on gradient-based HPO methods has not focused on text-processing, with notable exceptions in Hu et al. (2019) and Lorraine et al. (2020). This domain mismatch makes it unclear how well these methods may work for the large language model setting.

We present the first study of gradient-based hyper-parameter learning on sequence-to-sequence tasks (Sutskever et al., 2014). We extend a greedy gradient-based approach that has been applied previously to image classification tasks (Luketina et al., 2016; Wu et al., 2018; Baydin et al., 2017),

[1]See Appendix A for a full description of related works.

[2]We refer to hyper-parameters which vary over a training run as *dynamic*, and those which are constant as *static*.

as it is simple, generalizable, and easily extensible. This allows us to apply greedy hyper-parameter learning to a) multiple hyper-parameters simultaneously and b) experiment across models and tasks. We learn hyper-parameters for momentum and learning rate scaling for Transformer (Vaswani et al., 2017) sequence-to-sequence models for neural machine translation (NMT) and T5 model pretraining (Raffel et al., 2019).

For NMT, we show that hyper-parameter schedules can be learned greedily with minimal tuning across language pairs, and that those learned schedules can be more efficient than Bayesian-optimized tuning and more performant than optimal constant-valued tuning. We demonstrate the absence of 'short-horizon bias' while learning momentum, and the benefit of treating momentum as a *dynamic* hyper-parameter. For T5, we show that learning a learning rate scalar alongside momentum changes the behavior of that scalar, improving both the convergence speed and performance of T5 pretraining, gains which are reflected in performance on downstream NLU tasks.

2 Method

We use a method that allows hyper-parameters to be learned greedily by gradient descent over the course of training. Per training step, we perform a bi-level optimization to learn both the model parameters via the training loss, and learned hyperparameters via the *guidance loss*. The *guidance set* is held-out from the training data to provide the loss by which the hyperparameters are learned.

Let X denote a training dataset and Ω be a general optimizer function for training a model θ on X, with hyperparameters λ and loss function $\mathcal{L}_X$. Our training method can be summarized as:

$$g_t = \nabla_{\theta_t} \mathcal{L}_X(\theta_t) \qquad \theta_{t+1} = \Omega(\theta_t, g_t, \lambda_t)$$
$$\hat{g}_t = \nabla_{\lambda_t} \mathcal{L}_H(\theta_{t+1}) \qquad \lambda_{t+1} = \hat{\Omega}(\lambda_t, \hat{g}_t, \hat{\lambda}),$$

where at each time step t, the updated model parameters θ_{t+1} are first computed based on the gradient (g_t) of the training loss $\mathcal{L}_X$. To compute the guidance loss gradients ($\hat{g}_t$) for the hyper-parameters, we calculate the loss $\mathcal{L}_H$ of the new model θ_{t+1} on the guidance set. Finally, the updated hyperparameter values λ_{t+1} are obtained based on a meta-optimizer $\hat{\Omega}$ with corresponding meta-hyperparameters $\hat{\lambda}$. Thus in every training step, we update both the model parameters and the hyperparameters. The process is formalized in Algorithm 1 in Appendix B.

This method is greedy; the horizon of the guidance objective is limited to a single step. Wu et al. (2018) showed that greedy methods applied to learning the learning rate can have a bias towards tiny learning rates, which prevent them from learning and achieving good performance over longer horizons (*short-horizon bias*). We will explore the practical consequences of this phenomenon by using this method to learn a learning rate scalar α and momentum β_1.

3 Experiments

For NMT, we use Transformer models with 121M parameters and the LAMB optimizer (You et al., 2019)[3]. We train on NMT datasets from the WMT19 machine translation task (Barrault et al., 2019). For evaluation, we decode using beam search and report BLEU (Papineni et al., 2002) scores. For T5, we use the *small* configuration (60M parameters) and the Adafactor optimizer (Shazeer and Stern, 2018). We use the C4 dataset (Raffel et al., 2019). We report loss on the C4 development set and the same evaluation criteria as the original T5 paper for downstream tasks. For all hyper-parameter learning experiments, we use the Adam optimizer (Kingma and Ba, 2014) with default settings as meta-optimizer, tuning only the meta-learning-rate $\hat{\eta}$. For the guidance set, we use a single held-out training batch[4]. As the hyper-parameters must vary within constrained ranges, α is kept positive by an exponential activation function, and β_1 is constrained between 0 and 1 by a sigmoid.

[3]For complete experiment setup details, see Appendix C.

[4]In preliminary experiments, we found no benefit to a larger guidance set.

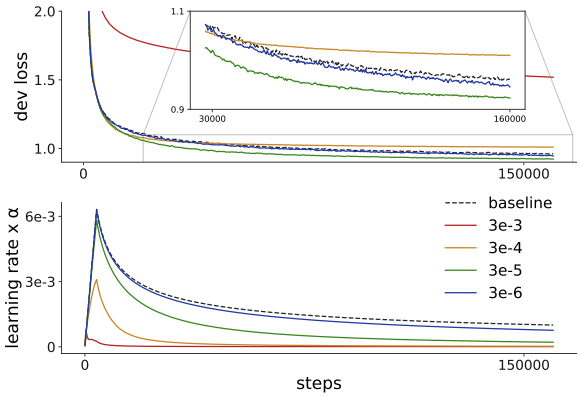

Figure 1: Learning α, varying $\hat{\eta}$ values. German-English NMT

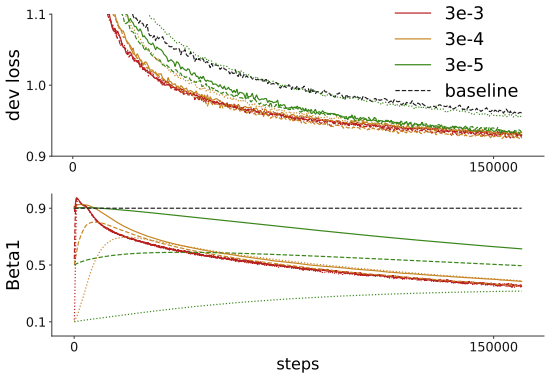

Figure 2: Learning β_1, varying $\hat{\eta}$ and init. values. German-English NMT

3.1 Neural Machine Translation

In Figure 1, we compare the evolution of learning the learning rate scalar (α) over training for runs with differing meta-learning rates ($\hat{\eta}$) for the German-English language pair. In Figure 2 we do the same for learning β_1, varying the initialization values in addition to $\hat{\eta}$. For learning α, the guidance optimization drives all runs to as low a learning rate as is allowed by the meta-learning rate, demonstrating the 'short horizon bias'. Note that some guided α runs do outperform the baseline, but require tuning of $\hat{\eta}$ to prevent convergence on the guidance objective. In contrast, the learned β_1 (Figure 2) runs converge to a similar schedule given a sufficiently high $\hat{\eta}$, decaying from high to low momentum over the course of training, regardless of the initialization value. All runs with guided β_1 outperform the baseline. To evaluate how well these gains generalize, we guide α and β_1 alone and together for 6 language pairs, setting $\hat{\eta}$ to 3e-5 for all runs (Table 1).

In order to evaluate the practical applicability of guiding these hyperparameters, we compare the guided runs to a typical hyperparameter optimization scheme, against which we can evaluate both performance and efficiency. We tune both the baseline runs (via the hyper-parameters directly) and the guided runs (via $\hat{\eta}$) with Bayesian optimization (BO)[5] for 100 trials. In Table 2, we find that across guided-parameter settings, the non-BO-optimized guided run outperforms the best BO-tuned baseline model, with some slight gains for the guided α run with further BO-tuning[6]. Note the guided β_1 runs do not require $\hat{\eta}$ tuning to reach best performance.

For all setups, the learned hyper-parameters achieve better performance than Bayesian optimization in fewer training runs and less time. Though the 'short-horizon bias' requires tuning $\hat{\eta}$ while learning α, doing so still yields performance and efficiency gains over BO-tuning. For β_1 alone, there seems to be no equivalent bias, as any sufficiently

Table 1: BLEU scores of baseline vs guided runs across language pairs. $\hat{\eta}$ is set to 3e-5.

	de-en	en-de	fi-en	en-fi	lt-en	en-lt
base	38.6	37.4	27.2	18.4	27.3	11.3
α	39.6	**39.4**	27.6	**19.7**	27.7	11.7
β_1	39.8	**39.4**	**28.4**	19.4	**28.0**	12.1
α+β_1	**39.9**	38.8	27.5	19.6	27.8	**12.2**

Table 2: BLEU scores of 100 BO-tuned runs vs untuned for baseline and guided runs, on *de-en*. Time is summed runtime in hours.

	time	# runs	α	β_1	α+β_1
base	7.1	1	38.6	38.6	38.6
+ BO	708	100	39.4	39.4	39.5
guided	43.2	4	39.6	39.8	**39.9**
+ BO	1.1k	100	39.7	39.8	**39.9**

[5]The specific algorithm we use is Gaussian Process Bandits (Frazier, 2018; Golovin et al., 2017).

[6]We count the 4 different values of $\hat{\eta}$ we tried in Figure 1 as tuning runs for the non-BO-tuned guided setup.

high $\hat{\eta}$ converges to roughly the same useful schedule. The BO-tuned optimal static β_1 value (0.73) approximates the average β_1 of the converged runs in Figure 2, suggesting that the remaining 0.4 BLEU points are only attainable with a β_1 value that changes over the course of training. Learning both hyper-parameters together does not change their evolution but yields a small additional boost.

3.2 T5 pretraining

We run similar experiments for T5 models, learning α, β_1, and both. For α, we see the learning rate scalar decrease prematurely similarly to the NMT setting, demonstrating again the 'short-horizon bias' (Appendix, Figure 2), but no guided run outperforms the baseline, even with low $\hat{\eta}$ values[7]. For β_1 alone, we replicate a similar converged schedule as in the NMT setting, but see only minor changes in development set loss across all models, including those varying β_1 without learning during training (Appendix, Figure 2). This suggests that tuning β_1 in general is less useful in this setting. Interestingly, when we tune both hyper-parameters together, the evolution of the α parameter changes character (Figure 3), and we find a 3X improvement in speed of convergence relative to baseline and increases in final performance for multiple different settings of $\hat{\eta}$.

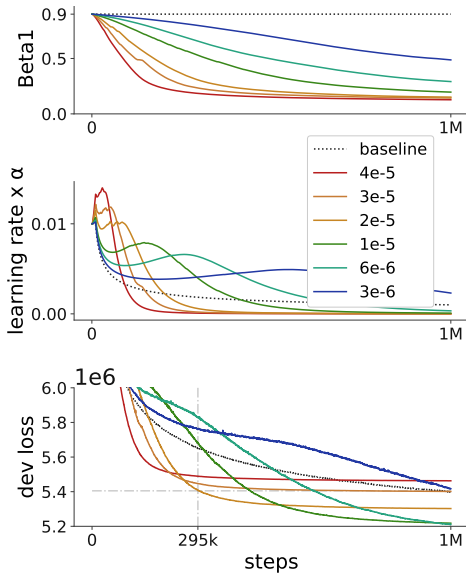

Figure 3: Learning α+β_1 for T5, varying $\hat{\eta}$.

We finetune the baseline and $\hat{\eta}$=1e-5 models on each of the downstream NLU tasks drawn from the GLUE (Wang et al., 2018) and superGLUE (Wang et al., 2019) benchmarks, as well as SQuAD (Rajpurkar et al., 2016), using the same finetuning settings as the original T5 paper[8] (Table 3). We find improvements across 15 of 18 downstream NLU tasks, with average improvements of 0.4 points on GLUE and 1.4 points on superGLUE.

[7]See Appendix F for full details on isolated α and β_1 experiments.

[8]For details on the setup of finetuning, see Appendix C.4. For full results, including on SQuAD, see Appendix F.1.

GLUE	CoLA *φcorr*	SST *acc*	MRPC $\overline{met}$	STS $\overline{met}$	QQP $\overline{met}$	MNLI $\overline{met}$	QNLI *acc*	RTE *acc*	WNLI *acc*	avg
base	**47.9**	92.3	88.5	84.1	87.4	84.1	90.2	67.9	57.7	77.8
α+β_1	44.5	**92.6**	**90.0**	**85.1**	**87.7**	**84.9**	**90.5**	**69.0**	**59.2**	**78.2**

sGLUE	BoolQ *acc*	CB $\overline{met}$	COPA *acc*	MultiRC $\overline{met}$	ReCoRD $\overline{met}$	RTE *acc*	WiC *acc*	WSC *acc*	avg
base	73.6	**98.4**	58.0	43.7	63.8	64.6	67.2	67.3	67.1
α+β_1	73.6	95.3	**61.0**	**46.0**	**64.0**	**66.4**	**67.6**	**74.0**	**68.5**

Table 3: Fine-tuning baseline and α+β_1 models on the GLUE and superGLUE (sGLUE) NLU tasks. $\overline{met}$ denotes the mean of the two metrics typically reported for that task, and *avg* takes the average across tasks. Max value of 5 runs shown, see Appendix F.1 for average results.

4 Discussion and Limitations

Our results shed light into multiple facets of Hyper-parameter optimization (HPO). For Neural Machine Translation, we show that although learning the learning rate scalar decays the learning rate prematurely when allowed to converge to the guidance objective (exhibiting the 'short-horizon bias' (Wu et al., 2018)), tuning the meta-learning-rate produces better results with less tuning than Bayesian optimized static tuning. In learning momentum, we demonstrate the absence of short-horizon bias; for momentum, and potentially other hyper-parameters, greedy gradient-based HPO can learn over a single run a schedule which out-performs optimal static tuning. For hyper-parameters such as momentum whose optimal values change over training, methods which allow for dynamic hyper-parameters will always have an edge over static tuning methods.

In our T5 experiments, we show that the 'recipe' which yielded good results in NMT produced, with minimal tuning, a pretrained model which outperforms the baseline after finetuning on downstream NLU tasks. We discovered that learning hyper-parameters in conjunction can alter their evolution over training. When learned alongside momentum, the initial growth of the learning-rate scalar followed by gradual decay is a result that is not explicable by the short-horizon bias, which would predict monotonic and premature decay to zero. This raises the possibility that learning certain hyper-parameters dynamically may be constrained by the static values of non-learned hyper-parameters, and that learning multiple hyper-parameters together may be necessary in some settings to make learning any of them useful. Characterizing the phenomenon of interaction between hyper-parameters is a direction for future work.

Our experiments are limited to two global hyper-parameters which are typically tuned. Future work should explore a wider set of hyper-parameters and at a varying granularity (e.g. a distinct hyper-parameter value per parameter (Lorraine et al., 2020)). We show that learning hyper-parameters together can alter their dynamics but leave to future work the characterization of the mechanism and mapping of interactions between learned hyper-parameters. We have shown that greedily learning the learning rate scalar can produce behavior unexplained by the short-horizon bias, but have left to future work the characterization of this phenomenon. The method we explore is limited to differentiable hyper-parameters, and is greedy, so may be improved upon by more complex methods which can take into account either non-differentiable hyperparameters (MacKay et al., 2019) and/or longer horizons (Micaelli and Storkey, 2021).

5 Broader Impact

Since Wu et al. (2018) described short-horizon bias for greedy methods, work in the gradient-based HPO community has progressed towards more complex methods which seek to address short-horizon bias with longer horizons (Micaelli and Storkey, 2021) or by other means (Donini et al., 2019). Our result showing the absence of bias for learning momentum, and easy performance gains for NMT when doing so, should encourage further evaluation of the behavior of diverse learnable hyper-parameters under greedy meta-optimization. Additionally, we have shown that intuitions about the short-horizon bias do not fully explain the behavior of the learning-rate scalar, which increases at the start of training when learned alongside momentum. These observations, taken together, should encourage further exploration of greedy gradient-based methods. We do not anticipate this work having potential negative societal impacts beyond those posed by automated methods in machine learning in general. Rather we hope that it may contribute towards the realization of efficient and general gradient-based HPO, which will help improve the efficiency of training models, reduce energy consumption, and democratize access to machine learning. We hope that our encouraging results and release of the code we used to produce them[9] will facilitate future work within the research community and give practitioners the tools to apply gradient-based HPO in diverse settings.

[9] `https://www.github.com/google-research/google-research/tree/master/gradient_based_tuning`

6 Reproducibility Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Sections 3 and 4.,

   (b) Did you describe the limitations of your work? [Yes] See latter portion of Section 4.

   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We anticipate no specific potential negative impacts beyond those of improving automated machine learning methods in general. We state this in Section 5.

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes] We do not violate the guidelines.

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We present no theoretical results.

   (b) Did you include complete proofs of all theoretical results? [N/A] We present no theoretical results.

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [N/A] We will release the code prior to the publication of the work. While this is clearly not the same as releasing it now (at submission time), we intend to do so as open-sourcing the code is a main aspect of the intended impact of the work.

   (b) Did you include the raw results of running the given instructions on the given code and data? [N/A] See above.

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [N/A] Close analogues of the figures in this paper will be automatically generated by the training code.

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes] The code, which will be released prior to publication, will be well documented.

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] See Appendix C.

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] We took care to ensure our experiments comparing methods were fair, including in these mentioned categories.

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] We vary $\hat{\eta}$, combination of hyper-parameters, and in comparing NMT to T5 pretraining, we vary optimizer, model, and task.

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes] See section 3 and Appendix C.

(i) Did you compare performance over time? [Yes] See Figures in Section 3.

(j) Did you perform multiple runs of your experiments and report random seeds? [No] We did perform multiple runs of the experiments but do not report random seeds.

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We do not report error bars, we report the max and average metric values over repeated runs.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A] We do not employ NAS approaches.

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [Yes] See Section 3.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 3.

(b) Did you mention the license of the assets? [Yes] See Appendix G.

(c) Did you include any new assets either in the supplemental material or as a URL? [No] We will include a link to the code at publication time.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Our experiments were performed on publicly available datasets.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] None of our datasets contains personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Acknowledgements. The authors would like to thank Andrew Chou, Felix Stahlberg, Ji Ma, and the anonymous reviewers, for their helpful comments.

References

Adiwardana, D., Luong, M., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y., and Le, Q. V. (2020). Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977.

Almeida, L. B., Langlois, T., Amaral, J. D., and Plakhov, A. (1998). Parameter adaptation in stochastic optimization. *On-Line Learning in Neural Networks, Publications of the Newton Institute*, pages 111–134.

Antoniou, A., Edwards, H., and Storkey, A. (2018). How to train your maml. *arXiv preprint arXiv:1810.09502*.

Bansal, T., Jha, R., Munkhdalai, T., and McCallum, A. (2020). Self-supervised meta-learning for few-shot natural language classification tasks. *arXiv preprint arXiv:2009.08445*.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M., and Zampieri, M. (2019). Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. (2017). Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*.

Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural Comput.*, 12(8):1889–1900.

Bengio, Y. (2012). *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways. *CoRR*.

Clarke, R. M., Oldewage, E. T., and Hernández-Lobato, J. M. (2021). Scalable one-pass optimisation of high-dimensional weight-update hyperparameters by implicit differentiation. *CoRR*, abs/2110.10461.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (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.

Domke, J. (2012). Generic methods for optimization-based modeling. In Lawrence, N. D. and Girolami, M., editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 318–326, La Palma, Canary Islands. PMLR.

Donini, M., Franceschi, L., Pontil, M., Majumder, O., and Frasconi, P. (2019). Scheduling the learning rate via hypergradients: New insights and a new algorithm. *CoRR*, abs/1910.08525.

Feurer, M. and Hutter, F. (2019). *Hyperparameter Optimization*, pages 3–33. Springer International Publishing, Cham.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. (2017). Forward and reverse gradient-based hyperparameter optimization. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1165–1173. PMLR.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. (2018). Bilevel programming for hyperparameter optimization and meta-learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR.

Frazier, P. I. (2018). A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Fu, J., Luo, H., Feng, J., Low, K. H., and Chua, T. (2016). Drmad: Distilling reverse-mode automatic differentiation for optimizing hyperparameters of deep neural networks. *CoRR*, abs/1601.00917.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. (2017). Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM.

Grazzi, R., Pontil, M., and Salzo, S. (2021). Convergence properties of stochastic hypergradients. In Banerjee, A. and Fukumizu, K., editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 3826–3834. PMLR.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. (2020). Flax: A neural network library and ecosystem for JAX.

Hu, Z., Tan, B., Salakhutdinov, R. R., Mitchell, T. M., and Xing, E. P. (2019). Learning data manipulation for augmentation and weighting. *Advances in Neural Information Processing Systems*, 32.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Blanco, E. and Lu, W., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics.

Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2016). Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560.

Lorraine, J., Vicol, P., and Duvenaud, D. (2020). Optimizing millions of hyperparameters by implicit differentiation. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1540–1552. PMLR.

Luketina, J., Berglund, M., Greff, K., and Raiko, T. (2016). Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pages 2952–2960. PMLR.

MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. B. (2019). Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *CoRR*, abs/1903.03088.

Maclaurin, D., Duvenaud, D., and Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2113–2122, Lille, France. PMLR.

Majumder, O., Donini, M., and Chaudhari, P. (2019). Learning the learning rate for gradient descent by gradient descent. In *Proceedings of the AutoML workshop*.

Micaelli, P. and Storkey, A. J. (2021). Gradient-based hyperparameter optimization over long horizons. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737 – 746.

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Raghu, A., Lorraine, J., Kornblith, S., McDermott, M., and Duvenaud, D. K. (2021). Meta-learning to improve pre-training. *Advances in Neural Information Processing Systems*, 34:23231–23244.

Raghu, A., Raghu, M., Kornblith, S., Duvenaud, D., and Hinton, G. (2020). Teaching with commentaries. *arXiv preprint arXiv:2011.03037*.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. (2019). Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1723–1732. PMLR.

Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wu, Y., Ren, M., Liao, R., and Grosse, R. B. (2018). Understanding short-horizon bias in stochastic meta-optimization. *CoRR*, abs/1803.02021.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. (2019). Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

A Related Work

The field of hyperparameter optimization (HPO) is well summarized in Feurer and Hutter (2019). Here we review related work in gradient-based HPO, of which our method is one approach. Online gradient-based HPO was proposed by Almeida et al. (1998). Bengio (2000) formulated hyperparameter search in terms of optimization. Domke (2012) described a strategy to compute the gradient of loss with respect to hyperparameters in a CRF model. The use of validation-loss gradients to update continuous hyperparameters by backpropagating through the entire training procedure was demonstrated by Maclaurin et al. (2015). To reduce time-complexity of tracing back through the entire training procedure, subsequent work explored approaches where the parameter and hyperparameter updates are performed in an alternating fashion (Luketina et al., 2016; Franceschi et al., 2017, 2018; Baydin et al., 2017; Majumder et al., 2019). Luketina et al. (2016) proposed greedy per-step validation loss gradient updates, applied to regularization hyperparameters that are trained alongside the elementary parameters of the model. Baydin et al. (2017) described an application of the greedy approach to optimize learning rates using the training set loss. Wu et al. (2018) highlighted the short-horizon biases arising from the greedy strategy. Fu et al. (2016); Donini et al. (2019); Micaelli and Storkey (2021) presented approaches that overcome some of the limitations of the greedy strategy while being more efficient than the full trajectory approach of Maclaurin et al. (2015). The above methods considered either forward- or reverse-mode differentiation to compute the hyper-gradients. Alternative approaches, using the Implicit Function Theorem to approximate the gradients, were explored in Pedregosa (2016); Lorraine et al. (2020); Clarke et al. (2021); Grazzi et al. (2021). Shaban et al. (2019) proposed an approach using truncated backpropagation to approximate the hypergradient. MacKay et al. (2019) presented a method for learning a hyperparameter schedule that works for non-differentiable hyperparameters. Some works focus on using gradients to learn data weighting or augmentation schemes, such as Hu et al. (2019). Raghu et al. (2020) leverages gradient methods to learn various 'commentaries' that are example-level parameters that can improve performance via example weighting and data manipulation and also provide insights into model training. MAML (Finn et al., 2017) and subsequent works (Antoniou et al., 2018; Bansal et al., 2020) employ a bi-level, gradient based training procedure using a distribution over tasks that improves the generalization performance and can be utilized to learn hyperparameters. Raghu et al. (2021) apply a gradient-based method to meta-learn hyperparameters for multi-task pretraining on protein-protein interaction networks.

B Algorithm

Algorithm 1 Guided Learning

Require: θ_0: initial parameter vector
Require: λ_0: initial hyperparameter vector
Require: $\hat{\lambda}$: hyperparameter vector of meta-optimizer
  $t \leftarrow 0$ ▷ Initialization
  while θ_t not converged **do**
    $(X_t, H_t) \leftarrow$ GetNewMiniBatch() ▷ New training/guidance mini-batch
    $g_t^X \leftarrow \nabla_{\theta_t}$ComputeLoss$(X_t, \theta_t)$ ▷ Gradient of train loss wrt θ_t
    $\theta_{t+1} \leftarrow$ Optimizer$(g_t^X, \theta_t, \lambda_t)$ ▷ Parameter update
    $\hat{g}_t^H \leftarrow \nabla_{\lambda_t}$ComputeLoss$(H_t, \theta_{t+1})$ ▷ Gradient of guidance loss wrt λ_t
    $\lambda_{t+1} \leftarrow$ MetaOptimizer$(\hat{g}_t^H, \lambda_t, \hat{\lambda})$ ▷ Hyperparameter update
    $t \leftarrow t + 1$
  end while

C Setup

C.1 NMT Experiments

C.1.1 Model. For all experiments, we use the JAX framework (Bradbury et al., 2018), building off of models from the flax library (Heek et al., 2020). We use Transformer models (Vaswani et al., 2017) and the LAMB optimizer (You et al., 2019), with a 32k sentence-piece vocabulary (Kudo and Richardson, 2018) for each language pair. Our Transformers have 8 heads and 6 layers with a total of 121M parameters, and for the LAMB optimizer we use the default values of β_1, β_2, and ϵ as 0.9, 0.999, and 1e-6 respectively.

C.1.2 Data. We train on 6 different language pairs, with training, development, and test sets drawn from the WMT19 machine translation task (Barrault et al., 2019). We tokenize the language pairs into joint 32K subword vocabularies with SentencePiece models (Kudo and Richardson, 2018). After filtering the datasets slightly by language ID and with length-based heuristics, we remove a single batch of the remaining data to set aside as a guidance set for each language pair. This is based on our preliminary experiments where we found no change in performance between holding out 1% of the training data for guidance (iterated through repeatedly over training) or holding out a single batch (applied at every step), so throughout this work we hold out only a single batch for the guidance set[10]. The resulting dataset sizes are shown in Table 4.

Table 4: Comparing dataset sentence count across language pairs. The acronyms de, en, fi and lt refer to German, English, Finnish and Lithuanian, respectively.

	de-en	en-de	fi-en	en-fi	lt-en	en-lt
train	32M	32M	5.5M	5.5M	1.9M	1.9M
guide	2165	2227	2363	2337	2339	2305

C.2 Training

We train with dropout and attention dropout both set to 0.1, and without label smoothing or weight decay regularization. The default learning rate is set to 0.4, which follows a square-root decay schedule following a a linear warmup of 4000 steps. We use a training batch size of ∼2, 300 examples on average. In the experiments where we compare learned hyperparameters to Bayesian HPO (Snoek et al., 2012), the objective for the BO is to minimize the loss on the development set, and we select the best of 100 trials for each BO run.

C.2.1 Evaluation. We decode with beam search decoding with a beam size of 4, and report BLEU (Papineni et al., 2002) scores calculated using the sacreBLEU tool (Post, 2018).

C.3 T5 Experiments

C.3.1 Model. For the T5 experiments, we pretrain *T5* models (Raffel et al., 2019) using the *Adafactor* optimizer (Shazeer and Stern, 2018). We train a T5 model in the *small* configuration, with 8 layers and 6 attention heads per layer and a total of 60M parameters. For Adafactor we use a default learning rate of 1e-3 and a *decay_rate* of 0.8.

[10]We likely see no difference because at most we learn two hyperparameters. With higher-dimensional learned hyperparameterizations, overfitting on the guidance set may become a concern that can be addressed by iterating through a larger guidance dataset.

C.3.2 Data and Training. For pretraining, we train for 1M steps on the C4 dataset (Raffel et al., 2019), using a 32k sentence-piece vocabulary, the same as in the original T5 paper. We use a batch size of 256 packed examples and a maximum input length of 512 sentence-pieces, with dropout set to 0.0. We use a learning rate of 0.01 with 10000 steps of constant value followed by reciprocal square root decay. The unsupervised objective is the same masked language modeling objective that was proposed in the original T5 paper. 15% of tokens are masked in the input sequence, replacing each masked span with a sentinel token. The model is then trained to predict the missing text for each sentinel token.

C.3.3 Evaluation. In pretraining, we report the loss on the C4 development set. For finetuning, we evaluate the appropriate metrics for each of the GLUE and superGLUE tasks. To arrive at the final average for each set of tasks, we follow the T5 paper in averaging the metrics within each tasks (to get the $\overline{met}$ values shown in Table 3) and then simply averaging those scores across the tasks of the super-task.

C.4 Finetuning on Downstream NLU Tasks

We finetune on downstream NLU tasks from the GLUE and superGLUE meta-tasks. We initialize from the 1M step pretraining checkpoints and train for an additional 250,000 steps with a batch size of 8, mirroring the T5 paper finetuning scheme (Raffel et al., 2019).

C.5 Meta-Optimization

In our hyperparameter learning experiments, we meta-optimize with Adam and its default hyperparameters. β_1, β_2, and ϵ are set to 0.9, 0.999, and 1e-8 respectively. For both NMT and T5 experiments, we use a guidance batch size mirroring the size of the training batch in each setting.

While model parameters may be allowed to take positive or negative values, the hyperparameters we study must be bound to a range of appropriate values; the learning rate must be positive and momentum must be between 0 and 1. To achieve this, we pass the learned hyperparameters through an activation function; exponential for learning rate and sigmoid for momentum (Table 5). Unlike other hyperparameters, the learning rate is frequently set on a pre-determined schedule. In order to not override the pre-existing schedule, we learn a scalar α on the schedule which is initialized at 1.

Table 5: Learned hyperparameters and their activation functions.

hparam	activation fn	domain	init
α	e^x	$(0, \infty)$	1
β_1	$(1 + e^{-x})^{-1}$	$(0, 1)$	0.9

D Hardware

For all experiments, we use TPUv3 with 16 cores. NMT training runs took ~7 hours to train. T5 training took ~48 hours for pretraining and ~3-6 hours for finetuning depending on the task. In both setups, training runs that guided hyper-parameters took approximately 1.5X as long in terms of wall-clock time than baseline runs. The memory requirements of the guided and unguided runs were similar.

E Tuning β_1 via Bayesian optimization

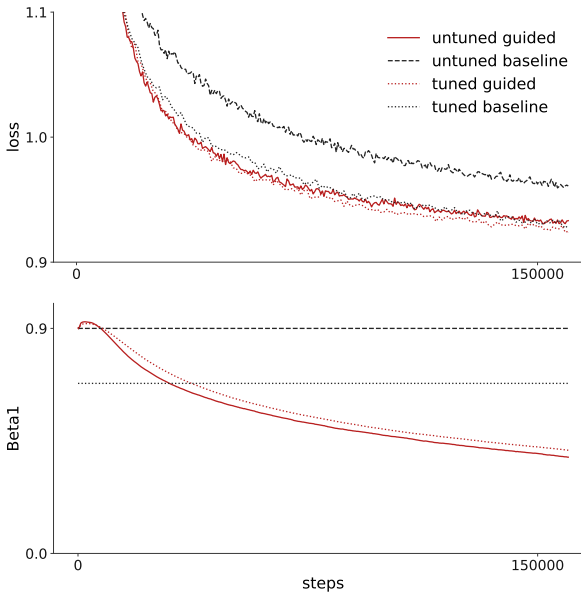

Figure 1: Learning β_1 for NMT models, comparison of untuned learned and baseline runs (solid lines) to BO-tuned learned and baseline runs (dotted lines). These runs correspond to those reported in the β_1 column in Table 2.

F T5 experiments

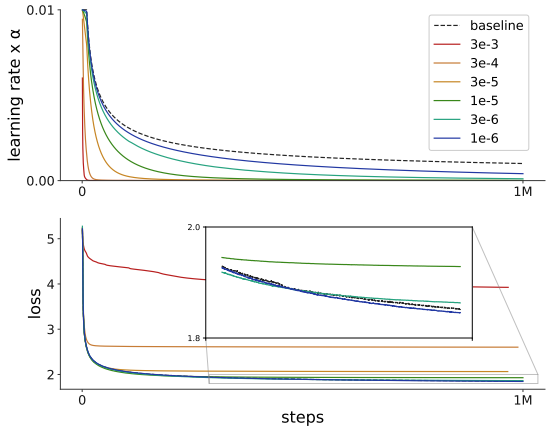

Figure 2: Learning α alone for T5 pretraining, comparison of a sweep over color-coded meta learning rates to baseline (black). The lowest meta-learning rate setting (1e-6, in blue) does outperform the baseline, but only very slightly. The short-horizon bias is evident; note that all learning rate scalars only decrease relative to the baseline learning rate schedule.

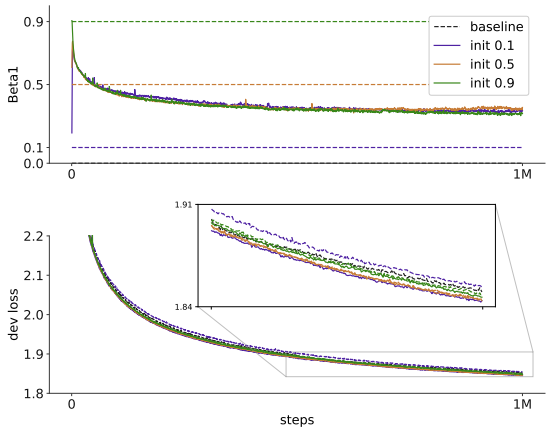

Figure 3: Learning β_1 alone for T5 pretraining, comparison of learned (meta learning rate 3e-3) runs vs baseline for initializations [0.1, 0.5, 0.9], and default baseline 0.0. Note that learned β_1 values converge to the same gradually decaying schedule, similar to that of the NMT models in Figure 2. The runs on that schedule do very slightly out-perform the non-learned hyperparameter runs. However, unlike in the NMT case, none of the changes in β_1, dynamic or static, have a significant impact upon the accuracy of the model at any point in training. This suggests that this setup is simply insensitive to the value of β_1.

F.1 Downstream NLU Task Full Results

	GLUE avg	CoLA $\phi corr$	SST *acc*	MRPC *F1*	MRPC *acc*	STS *Pearson*	STS *Spearman*	QQP *F1*
base	77.8	47.9	92.3	90.3	86.7	84.0	84.3	85.6
α+β_1	78.2	44.5	92.5	91.7	88.2	84.9	85.3	86.0
	QQP *acc*	MNLI-m *acc*	MNLI-mm *acc*	QNLI *acc*	RTE *acc*	WNLI *acc*	SQuAD *EM*	SQuAD *F1*
base	89.2	83.8	84.5	90.2	67.9	57.7	88.2	80.1
α+β_1	89.5	84.6	85.2	90.5	69.0	59.2	88.3	80.6
	sGLUE avg	BoolQ *acc*	CB *F1*	CB *acc*	COPA *acc*	MultiRC *F1a*	MultiRC *EM*	ReCoRD *F1*
base	67.1	73.6	98.1	98.7	58.0	18.9	68.5	63.3
α+β_1	68.5	73.6	94.6	96.0	61.0	22.6	69.7	63.6
	ReCoRD *acc*	RTE *acc*	WiC *acc*	WSC *acc*				
base	64.3	64.6	67.1	67.3				
α+β_1	64.5	66.4	67.6	74.0				

Table 6: Fine-tuning baseline and learned α+β_1 models on SQuAD, plus the 9 GLUE and 8 superGLUE (sGLUE) downstream NLU tasks. All values are the *max* of 5 separate runs.

	GLUE avg	CoLA $\phi corr$	SST *acc*	MRPC *F1*	MRPC *acc*	STS *Pearson*	STS *Spearman*	QQP *F1*
base	79.8	47.3	92.2	89.9	86.0	83.6	84.0	87.2
α+β_1	80.1	43.6	92.5	90.8	87.0	84.6	85.0	87.2
	QQP *acc*	MNLI-m *acc*	MNLI-mm *acc*	QNLI *acc*	RTE *acc*	WNLI *acc*	SQuAD *EM*	SQuAD *F1*
base	90.5	83.8	84.4	89.1	64.8	57.5	88.1	80.0
α+β_1	90.5	84.5	85.1	90.3	66.0	57.2	88.2	80.5
	sGLUE avg	BoolQ *acc*	CB *F1*	CB *acc*	COPA *acc*	MultiRC *F1a*	MultiRC *EM*	ReCoRD *F1*
base	64.8	73.1	97.9	98.4	56.4	6.4	62.1	63.1
α+β_1	67.3	72.7	93.2	93.1	57.2	21.0	69.4	63.3
	ReCoRD *acc*	RTE *acc*	WiC *acc*	WSC *acc*				
base	64.1	63.8	66.1	63.5				
α+β_1	64.3	65.9	66.7	73.5				

Table 7: Fine-tuning baseline and learned α+β_1 models on SQuAD, plus the 9 GLUE and 8 superGLUE (sGLUE) downstream NLU tasks. All values are the *average* of 5 separate runs.

G Licensing of Data

WMT (Workshop on Machine Translation) 2019: *http://www.statmt.org/wmt19/translation-task.html* and downloaded from *https://www.tensorflow.org/datasets/catalog/wmt19_translate*

> LICENSING OF DATA (from the statmt.org website) states that it can be used for research purposes:
>
> *The data released for the WMT19 news translation task can be freely used for research purposes, we just ask that you cite the WMT19 shared task overview paper, and respect any additional citation requirements on the individual data sets. For other uses of the data, you should consult with original owners of the data sets.*

The C4 dataset is released by Google, available at *https://www.tensorflow.org/datasets/catalog/c4*, licensed by *Creative Commons Attribution 4.0 License*