A Thorough Evaluation of Task-Specific Pretraining for Summarization

Sascha Rothe*
Google
rothe@google.com

Joshua Maynez*
Google
joshuahm@google.com

Shashi Narayan*
Google
shashinarayan@google.com

Abstract

Task-agnostic pretraining objectives like masked language models or corrupted span prediction are applicable to a wide range of NLP downstream tasks (Raffel et al., 2019), but are outperformed by task-specific pretraining objectives like predicting extracted gap sentences on summarization (Zhang et al., 2020). We compare three summarization specific pretraining objectives with the task agnostic corrupted span prediction pretraining in a controlled study. We also extend our study to a low resource and zero shot setup, to understand how many training examples are needed in order to ablate the task-specific pretraining without quality loss. Our results show that task-agnostic pretraining is sufficient for most cases which hopefully reduces the need for costly task-specific pretraining. We also report new state-of-the-art number for two summarization tasks using a T5 model with 11 billion parameters and an optimal beam search length penalty.

1 Introduction

Previous work mostly used task-agnostic pretraining methods like corrupted span prediction (T5; Raffel et al., 2019), masked language model (BERT; Devlin et al., 2018), denoising objective (BART; Lewis et al., 2019 or a vanilla language model (GPT; Radford et al., 2019). Intuitively it makes sense to refine the pretraining to a setup that closer resembles the downstream task. Wang et al. (2020) demonstrate that task-specific priors into BERT language model pretraining improves on low-resource finetuning tasks. This is also done by Zhang et al. (2020) with PEGASUS, where important sentences are removed/masked from an input document and are generated together as one output sequence from the remaining sentences, to teach summarization models to do better content selection and Narayan et al. (2021) proposed a content planning pretraining objective with PEGASUS, by pre-pending the output sequence with the entity plans observed in it.

XSum	SOTA (Narayan et al., 2021)	47.80 / 25.06 / 39.76
	PEGASUS (Zhang et al., 2020)	47.21 / 24.56 / 39.25
	T5 base (ours; $\alpha = 0.9$)	42.96 / 20.38 / 35.10
	T5 xxl (ours; $\alpha = 0.8$)	**48.83 / 25.96 / 40.70**
CNN/DMail	SOTA (Dou et al., 2020)	**45.94** / 22.32 / **42.48**
	PEGASUS (Zhang et al., 2020)	44.17 / 21.47 / 41.11
	T5 xxl (Raffel et al., 2019)	43.52 / 21.55 / 40.69
	T5 base (ours; $\alpha = 0.9$)	43.09 / 20.67 / 39.96
	T5 xxl (ours; $\alpha = 0.8$)	45.32 / **22.60** / 42.17
SAMSum	SOTA (Rohde et al., 2021)	53.01 / 28.27 / 48.84
	T5 base (ours; $\alpha = 1.0$)	49.38 / 24.16 / 44.88
	T5 xxl (ours; $\alpha = 0.9$)	**53.10 /28.73 / 48.94**

Table 1: Current state-of-the-art ROUGE-1 / -2 / -L scores for summarization datasets and our results in the T5 framework with optimal beam alpha. Raffel et al. (2019) only report numbers for CNN/DailyMail.

PEGASUS achieved state of the art ROUGE-1/-2/-L scores (Lin, 2004) on BBC XSum (Narayan et al., 2018) with 47.21 / 24.56 / 39.25 and CNN/DailyMail with 44.17 / 21.47 / 41.11. These numbers could not be matched by Raffel et al. (2019) even when using a much larger model with up to 11 billion parameters. This seems to support the intuition that task specific pretraining is important for the best performance. However, Raffel et al. (2019) used a beam search length penalty (beam alpha) of 0.6. We set the beam alpha parameter to the optimal value and report new state of the art results on XSum and SAMSum (Table 1).

Given these new results we want to answer the questions if task-specific pretraining objectives are still at an advantage. To avoid any influence of hyperparameters, pretraining datasets, tokenization or evaluation scripts we reimplement all experiments in the same framework, namely the PEGASUS framework.[1] To our surprise, we found that in a controlled comparison the task-agnostic pretraining methods perform as good as task specific

*Equal contribution.

[1]https://github.com/google-research/pegasus

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 140–145
November 7–11, 2021.

pretraining methods for large finetuning setups. We further extend our study to a low resource and zero shot setup, to understand how many training examples are needed in order to ablate the task specific pretraining without quality loss. And finally we want to see if our findings also translate to other text generation tasks. We therefore pretrain a model with corrupted text and evaluate it on grammatical error correction.

2 Pretraining Models

We use a transformer architecture (Vaswani et al., 2017) with 12 hidden layers, a hidden size of 768, filter size 3072 and 12 attention heads, with a total of 223M parameters. All models are pretrained for 1.5 million steps on the C4 corpus (Raffel et al., 2019) with a batch size of 16, Adafactor (Shazeer and Stern, 2018), a learning rate of 0.01, and maximum input-output lengths of 512 and 256, in the PEGASUS framework. If not mentioned otherwise we do not perform any hyperparameter tuning but use the best performing hyperparameters founded by Zhang et al. (2020). We do not explore a pretraining plus prefinetuning setup in this paper (Aghajanyan et al., 2021).

We now briefly explain the task-agnostic objective of corrupted span prediction and two task-specific objectives, salient sentence selection for summarization and text corruption for grammar error correction. Additionally, we also experimented with the objectives of masking and predicting random and lead sentences.

Corrupted Span Prediction (T5) This pretraining objective is based on a span-prediction task, an adaptation of masked-language objective for autoregressive seq2seq models. As in BERT, we mask out 15% of the input text. We allow masking on continuous spans of lengths 1, 2, 3, 4 and 5 with probabilities 0.1, 0.2, 0.4, 0.2, 0.1, respectively. An example of span prediction:

Input: `This is an` [x] `sentence` [y] `words.`

Target: [x] `example` [y] `with eight`

Mask Salient Sentence (PEGASUS) We follow Zhang et al. (2020) to select and mask whole sentences from documents. The concatenated gap-sentences can be seen as a pseudo-summary and will serve as targets. To more closely approximate a summary, sentences that appear to be important/principal to the document are selected. As a proxy for importance ROUGE-1 F1 score between the sentence and the rest of the document is used.

Mask Random Sentence (MRNDS) We also pretrain a model with randomly select sentences as gap-sentences. This can be seen as a sentence level version of the masked language model (Devlin et al., 2018), a version of T5 that generates whole sentences or as a simplification of PEGASUS where the content selection aspect is missing.

Mask Lead Sentence (MLEADS) In this setup we pretrain a model with the first m sentences of a document as gap-sentences. This is motivated by the fact that for some text snippets, for example news, the most important information comes at the beginning of a paragraph. This is a natural setup for summarization since it is known that lead-sentences are a good baseline to compare summarization models against.

Text Corruption (TEXTCOR) Analogous to PEGASUS for summarization we pretrain a task specific model for grammatical error correction. To create pairs of broken and correct text snippets we corrupt each sentence using a combination of the following operations: a) drop tokens b) swap tokens c) insert tokens d) replace tokens e) drop characters f) swap characters g) insert characters h) lower-case a word i) upper-case the first character of a word. We limited our self to the fore mentioned purely unsupervised corruption techniques and do not use more sophisticated methods like replacing words with common misspellings as done by Náplava and Straka (2019).

3 Finetuning Experiments

All our experiments are done in the PEGASUS framework. We validate that the numbers are roughly identical with a comparable setup in T5.[2] For this, numbers in Table 1 labeled *T5 base ours* should match numbers in Table 2 labeled *T5 Full*. Both experiments correspond to the same model size conducted in different frameworks.

Datasets and Eval Metrics We measure the performance on three commonly used summarization benchmarks, namely CNN/DailyMail (Hermann et al., 2015), BBC XSum, (Narayan et al., 2018) and SAMSum (Gliwa et al., 2019) using ROUGE-1,

[2]https://github.com/google-research/text-to-text-transfer-transformer

	0	10	100	1000	10000	Full
	XSum (Lead-1: 16.30 / 01.61 / 11.95)					
T5	13.57 / 03.50 / 11.10 *	**26.49 / 08.15 / 20.92**	**34.87 / 13.08 / 27.50**	37.07 / 15.22 / 29.87	39.91 / 17.61 / 32.46	**44.34 / 22.01 / 36.65**
MRNDS	19.01 / 02.81 / 14.53	19.80 / 03.37 / 15.12	31.11 / 10.88 / 24.92	38.13 / 16.19 / 30.93	40.59 / 18.33 / 33.15	43.85 / 21.65 / 36.29
MLEADS	**19.35 / 02.60 / 14.76**	24.14 / 06.32 / 18.99	34.38 / 13.05 / 27.85	38.00 / 16.09 / 30.89	39.80 / 17.63 / 32.30	43.38 / 21.26 / 35.81
PEGASUS	19.04 / 03.05 / 13.76	23.45 / 06.05 / 17.90	33.77 / 12.95 / 27.16	**38.26 / 16.46 / 31.02**	**41.06 / 18.65 / 33.54**	44.15 / 21.87 / 36.58
	CNN/DailyMail (Lead-3: 39.60 / 17.70 / 36.20)					
T5	13.84 / 04.37 / 12.53 *	24.00 / 09.13 / 22.08	31.88 / 13.99 / 29.55	35.90 / 16.58 / 33.31	39.84 / 18.75 / 36.99	**43.59 / 21.42 / 40.56**
MRNDS	32.86 / 12.40 / 29.57	35.30 / 14.35 / 31.81	35.53 / 15.65 / 32.53	38.06 / 17.44 / 34.94	**41.16 / 19.12 / 38.09**	**43.59 / 21.18 / 40.40**
MLEADS	**39.68 / 17.82 / 36.07**	**39.68 / 17.82 / 36.07** ^{o}	**39.68 / 17.82 / 36.07** ^{o}	**39.68 / 17.82 / 36.07** ^{o}	40.45 / 18.79 / 37.48	43.18 / 20.83 / 40.07
PEGASUS	34.11 / 13.35 / 29.86	33.75 / 13.76 / 29.70	34.37 / 15.05 / 30.97	37.09 / 17.39 / 34.09	40.41 / 19.12 / 37.43	43.53 / 21.15 / 40.35
	SAMSum (Lead-5: 31.94 / 09.91 / 27.03)					
T5	04.00 / 00.65 / 03.75 *	**33.76 / 11.30 / 29.43**	41.28 / 16.51 / 37.04	46.18 / 21.44 / 41.59	50.16 / 26.36 / 45.83	**50.96 / 27.02 / 46.56**
MRNDS	26.90 / 07.63 / 24.79	31.03 / 10.25 / 27.94	**42.12 / 18.06 / 38.22**	**47.25 / 21.39 / 42.20**	**50.19 / 25.74 / 45.96**	50.54 / 25.98 / 46.45
MLEADS	**29.86 / 08.59 / 26.43**	33.75 / 10.97 / 29.83	39.02 / 15.10 / 34.89	45.18 / 20.11 / 40.10	49.15 / 24.65 / 44.58	49.84 / 25.72 / 45.56
PEGASUS	22.78 / 05.80 / 20.60	29.76 / 09.83 / 26.61	38.94 / 15.62 / 34.77	46.45 / 21.15 / 41.29	50.15 / 25.98 / 45.80	50.21 / 26.34 / 46.18

Table 2: ROUGE-1 / -2 / -L scores in summarization datasets. Results are shown on their full test sets using only 10, 100, 1000 and 10000 training examples, and the whole training set (all). We also report on zero-shot results. We report Lead-1 baseline for BBC from (Narayan et al., 2018) and Lead-3 baseline for CNN/DailyMail from (Rothe et al., 2020). For SAMSum, we achieve the best lead scores when we select top 5 sentences for each input. Result in gray are worse than the lead sentence baseline. Best results in each block are bolded. Results marked with * are not comparable, see text. For results marked with ^{o}, the untrained checkpoint at step 0 was performing best on the development set.

-2 and -L as metric. The datasets differ in the degree of abstraction and summarization length. The summaries of CNN/DailyMail are more of extractive nature and have an average length of 3 sentences. The summaries of BBC XSum are single-sentences and more abstractive. The SAMSum summaries consist of 2-3 meeting minutes. Finally, the CNN/DailyMail, BBC XSum and SAMSum datasets have 287k/13.4k/11.5k, 204k/11.3k/11.3k and 14.7k/818/819 training/development/test examples, respectively. We finetune our pretrained models on the full datasets and subsampled versions with 10, 100, 1,000 and 10,000 examples.

During finetuning, we use maximum input/output lengths of 1024/128 for CNN/DailyMail, 1024/64 for XSum and 512/128 for SAMSum. All models were finetuned with a batch size of 256. The best model was selected based on the ROUGE-L performance on the full development set. During inference, all models were decoded with a beam alpha of 0.8 and a beam size of 5. Results shown in Table 2 are the average performance of 5 models trained with different samples, as low resource setups are known to have high-variance.

Results We found that the performance of the span prediction objective is always better or on par with the performance of the salient sentence prediction objective for all three datasets when using the whole training set. Linguistically, it might be more interesting to generate full sentences than spans, but empirically, we found no evidence to support that the mask salient sentence pretraining is better at content selection than the corrupted span pretraining for summarization. In fact, we found that constraining pretraining to task-specific information such as the most important information at the beginning of a paragraph (MLEADS; CNN/DailyMail), makes it hard to generalize across datasets and leads to inferior performance compared to pretraining by generating random sentences (MRNDS).

For low-resource setups results varied a bit depending on the task. For abstractive datasets such as XSum and SAMSum, T5 achieved better performance than PEGASUS with as little as 10 or 100 examples. With 1000 and 10000 examples, results from both models were on par for SAMSum, but PEGASUS reported better than T5 for XSum. For CNN/DailyMail, PEGASUS continuously outperformerd T5 for all low-resource setups. On the other side CNN/DailyMail is not ideal for evaluating low-resource models due to the extractive nature of summaries; one can simply perform well by selecting the first few sentences. The Lead baseline and MLEADS are on par and outperform the other methods, while MLEADS does not use any training data when 1000 examples or less are provided.

Zero-Shot We also assess how well pretrained models perform out-of-the-box on different generation tasks (zero-shot). For this we simply infer

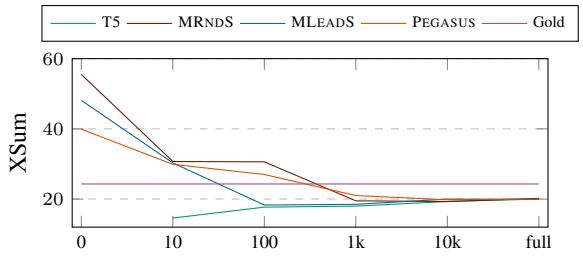

Figure 1: Comparison of how models adapt to target lengths from zero-shot to low-resource cases. We plot the average summary lengths for different models. We report results on XSum, similar patterns were found on CNN/DailyMail and SAMSum.

on the test sets using different pretrained checkpoints without any finetuning. Results in Table 2 are not surprising that the sentence-level pretraining in MRNDS, MLEADS and PEGASUS are better than T5 in producing well-formed summaries; also it is probably not fair to evaluate T5 for zero-shot summarization as T5 models are pretrained to generate masked spans and not full sentences.

Length comparisons It has been argued that ROUGE tends to prefer longer summaries, so we wanted to investigate if (a) model leverages this phenomenon and (b) if it is unfair to compare different pretraining methods trained on self-supervised targets with different length distributions. As depicted in Figure 1, for the zero-shot case we observe very different average lengths in predicted summaries for different models, with PEGASUS being closest to the target lengths. However, by 1000 training examples, all models start generating summaries of comparable lengths.

4 Grammatical Error Correction

We further investigate if our findings translate to other generation tasks. Here, we focus on the task of grammatical error correction, but also other important aspects of text generation show benefit from task specific pretraining and are still underexplored; e.g., improving evaluations (Sellam et al., 2020), factuality (Chen et al., 2020) or planning for grounded generation (Narayan et al., 2021).

Datasets and Eval Metrics For Grammatical Error Correction we fine-tune our pre-trained models on the FCE (Yannakoudakis et al., 2011) and W&I (Bryant et al., 2019) corpora. We evaluate on the standard benchmark of CoNLL-14, using CoNLL-13 as the development set. Reported numbers in

	10	100	1000	10000	all
T5	19.54	28.43	39.36	51.08	55.07
TEXTCOR	**21.71**	**30.86**	**49.40**	**55.94**	**59.67**
MRNDS	03.78	03.78	03.78	31.24	39.63

Table 3: $F_{0.5}$ scores on CoNLL-14 for the grammatical error correction task.

Table 3 are $F_{0.5}$ scores (Dahlmeier and Ng, 2012) computed by the M^2 scorer.[3]

Results As shown in Table 3 TEXTCOR outperforms T5 on all dataset sizes. The results also show that an unrelated task-specific pretraining objective hurts performance even when training on the full dataset. This is notable as for example the MRNDS pretraining is not that far of from a normal language model pretraining and should learn a reasonable amount about language and well formed sentences.

Zero Shot In contrast to summarization, no easy baseline exists for grammatical error correction. A simple copy baseline would give us a high word overlap like BLEU or ROUGE, but on our main metric $F_{0.5}$ this only gets a score of 4.24. Our pretrained TEXTCOR model achieves an $F_{0.5}$ score of 18.64, precision 40.94 and recall 5.87. The T5 model needs only 10 training examples to achieve the same $F_{0.5}$ score (Table 3). We hypothesize that zero shot performance of the TEXTCOR pretraining could be greatly improved by tuning the hyperparameters of the text corruption to better match distribution the CoNLL dev and test sets. However, this would limit the scope the pretrained model even further as this distribution would not translate to other datasets or related tasks, like correcting OCR (optical character recognition) or ASR (automatic speech recognition) errors.

5 Conclusion

We evaluated several pretraining techniques on two different text generation tasks, summarization and grammatical error correction. Our findings are that, while pretraining for summarization is very important, we found no evidence that task specific pretraining improved on common benchmarks for abstractive datasets, even in a low resource setting. On extractive datasets, task specific pretraining showed benefits but the results are below a sentence selection baseline, questioning the practical usefulness. Given the trend to larger neural network

[3] https://github.com/nusnlp/m2scorer

models with significant costs to train them, we recommend to use a task agnostic pretraining regime. Corrupted span prediction is currently our most successful candidate, with state-of-the-art results on two investigated summarization benchmarks. But we are curious if even more flexible pretraining technique will emerge. For grammar error correction, task specific pretraining was showing superior performance, especially in a low resource setting. We therefore believe that, task-specific pretraining or prefinetuning can still be useful for important aspects of text generation.

Acknowledgments

We thank the reviewers and the area chair for their feedback. We would like to thank Yao Zhao, Dipanjan Das, Mohammad Saleh, Samer Hassan, and Slav Petrov for useful discussions.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *CoRR*, abs/2101.11038.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020. KGPT: Knowledge-grounded pretraining for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8635–8648, Online. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2020. Gsum: A general framework for guided neural abstractive summarization. *arXiv preprint arXiv:2010.08014*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jakub Náplava and Milan Straka. 2019. Grammatical error correction in low-resource scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 346–356, Hong Kong, China. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simões, Vitaly Nikolaev, and Ryan T. McDonald. 2021. Planning with learned entity prompts for abstractive summarization. *CoRR*, abs/2104.07606, To appear in TACL.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Tobias Rohde, Xiaoxia Wu, and Yinhan Liu. 2021. Hierarchical learning for generation with long source sequences. *arXiv preprint arXiv:2104.07545*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Rui Wang, Shijing Si, Guoyin Wang, Lei Zhang, Lawrence Carin, and Ricardo Henao. 2020. Integrating task specific information into pretrained language models for low resource fine tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3181–3186, Online. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.