Insights into Pre-training via Simpler Synthetic Tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Pre-training produces representations that are effective for a wide range of downstream tasks, but it is still unclear what properties of pre-training are sufficient for effective gains. Notably, recent work shows that even pre-training on synthetic tasks can achieve significant gains in downstream tasks. In this work, we perform three experiments that iteratively simplify pre-training while still retaining much of its gains. First, building on prior work, we perform a more wide-ranging evaluation of three existing synthetic pre-training methods, evaluating them on six downstream tasks. Second, we find that pre-training on a simple and generic synthetic task, the set function, achieves almost the same benefits as the previous best synthetic task. Lastly, we find that more than 70% of the benefits of synthetic pre-training can be attained by using merely the statistics of pre-training. We believe our study provides useful and important insights and opens a new direction to understanding the effects of pre-training.

1 Introduction

Pre-training on a large amount of data from natural sources such as natural language text from the web—*natural pre-training*—is effective for a wide range of downstream tasks (Devlin et al., 2019; Brown et al., 2020; Chen et al., 2021; Lu et al., 2021). Recent works (Wu et al., 2021; Krishna et al., 2021; Ri & Tsuruoka, 2022; Anderson & Farrell, 2022) show that pre-training on data that is fully synthetically generated—*synthetic pre-training*—can also provide substantial gains, partially closing the gap between training from a randomly initialized model and a naturally pre-trained model.

What properties of pre-training are sufficient for effective gains? In this paper, we conduct a careful empirical study to help understand this question. We perform three experiments that iteratively simplify pre-training while still retaining much of its gains.

First, we perform a more wide-ranging evaluation of three previously proposed synthetic pre-training tasks, evaluating them on six downstream tasks. Prior works only evaluate their proposed method on a single downstream domain: Wu et al. (2021) on mathematical reasoning benchmarks, Chiang & Lee (2021) on GLUE, Krishna et al. (2021) on summarization, and Ri & Tsuruoka (2022) on language modeling and dependency parsing. Thus, it is unclear how general these methods are as well as how different synthetic pre-training tasks compare against each other. Our evaluation helps answer these questions. We find that the best performing synthetic task, LIME (Wu et al., 2021), closes a significant fraction of the gap between random initialization and natural language pre-training for summarization (85%), semantic parsing (83%), reading comprehension (55%), code translation (43%), and retrosynthesis (49%).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Second, we discover a surprisingly simple and generic synthetic task that captured 97% of the effectiveness provided by the existing best synthetic task. This simple task, which we call "`Set`", consists of outputting the input sequence without duplicate tokens. For example:

Input: a a c b a b

Output: a c b

Another surprisingly simple task, "`Identity`", which consists of copying the input to the output, alone provided large benefits for summarization and semantic parsing.

Third, we simplify even more and ask whether only the statistics of the pre-trained weights can explain the effectiveness of synthetic pre-training. We find that initializing parameters with statistics of a synthetic pre-trained model provides large benefits for some downstream tasks, recovering at least 70% of benefit from normal fine-tuning of a synthetic pre-trained model. Simplifying further allows us to see that by initializing all pre-attention layer norm parameters to a lower value (just one number!), we close more than 50% of the gap to natural language pre-training performance for a summarization task and a semantic parsing task.

In summary, our main contributions are:

1. We perform a more wide-ranging evaluation of three existing proposed synthetic tasks.
2. We discover two surprisingly simple tasks, `Set` and `Identity`, that could be useful for future work towards understanding transfer learning and how to design better synthetic tasks.
3. We find that initializing with the statistics of a synthetic pre-trained model can recover at least 70% of benefits of synthetic pre-training. Furthermore, smaller pre-attention layer norm weights play an important role for these initialization benefits.

2 Experimental Setup

In Sections 3 and 4 we evaluated how much different synthetic pre-training benefits six downstream tasks in the domains of natural language, semantic parsing, code, and chemistry. In this section, we provide details of the experimental protocol.

Architecture We trained with T5-Small, a 60 million parameter encoder-decoder Seq2Seq transformer model with 6 encoder layers, 6 decoder layers, 8 heads, head dimension 64, and MLP dimension 2048 (Raffel et al., 2020). As done in (Raffel et al., 2020), we framed all pre-training and downstream tasks into a text-to-text-format where the input and output are text strings.

Synthetic Pre-training Tasks In Section 3, we evaluated three previously proposed synthetic pre-training tasks: LIME (Wu et al., 2021), modeling the Nesting Dependency artificial language from (Ri & Tsuruoka, 2022), and the nonsense summarization task from (Krishna et al., 2021). In Section 4, we evaluated 18 simpler and more generic synthetic pre-training tasks on a summarization downstream task. We then evaluated `Set` and `Identity`, two of the simpler pre-training tasks, on all six downstream tasks. We provide explanations of the synthetic tasks in Sections 3.1 and 4.1.

Downstream Tasks We fine-tuned synthetically pre-trained models on Java to C# code translation (Lu et al., 2021), two semantic parsing benchmarks MTOP (Li et al., 2021) and WebQSP (Yih et al., 2016), USPTO-50K retrosynthesis (Liu et al., 2017), the reading comprehension benchmark SQuAD 1.1 (Rajpurkar et al., 2016), and 10K examples from the summarization task CNNDM-10K (Krishna et al., 2021). See a detailed description and input-output examples of each downstream task in Appendix D.

Pre-training and Fine-tuning Details See hyperparameters and other details of synthetic pre-training and downstream task fine-tuning in Appendix A and Appendix B.

Table 1: Evaluation of three previously proposed synthetic pre-training tasks and two simpler synthetic pre-training tasks. For tasks where we report both Exact Match (EM) and F1, the Average column is computed using EM.

	CNNDM-10K ROUGE1	MTOP EM/F1	WebQSP EM/F1	SQuAD EM/F1	Code Trans. EM	Retrosyn. EM	Average
Baselines							
Pre-trained T5	**35.8**	**81.0/95.2**	**83.1/91.9**	**77.5/86.0**	**61.6**	**43.1**	**63.7**
Wiki 10K	34.0	71.6/94.2	79.6/90.6	67.1/76.9	60.2	41.1	58.9
Random Init	18.9	38.6/83.0	28.2/72.9	17.2/25.2	57.0	39.2	33.2
Section 3: Benchmarking Existing Synthetic Pre-training							
LIME	**33.2**	**73.7/94.0**	**75.2/88.7**	**50.4/62.1**	**59.0**	**41.1**	**55.4**
Nested Language	27.1	65.9/91.9	58.5/83.5	50.3/62.4	58.8	40.4	50.2
Nons. Summary*	32.0	68.0/92.7	65.2/85.2	48.4/60.1	57.3	39.6	51.8
Section 4: Simpler Synthetic Pre-Training							
Set	**32.8**	**71.7/93.7**	**72.2/88.2**	**48.3/60.0**	**59.4**	**40.9**	**54.2**
Identity	30.2	68.6/93.0	69.8/86.8	26.1/35.4	57.8	40.5	48.8

Baselines We compared each synthetic task pre-trained T5-Small against randomly initialized (**Random Init**) and off-the-shelf natural language pre-trained T5-Small (**Pre-trained T5**). Pre-trained T5 was trained for 524K steps on the C4 dataset with the span-corruption objective (Raffel et al., 2020). We also compared against T5-Small trained on Wikipedia for 10K steps (**Wiki 10K**) with the same pre-training objective as off-the-shelf T5-Small. Just for the summarization downstream task CNNDM-10K, for fairer comparison as explained in Appendix C, we compared with off-the-shelf T5v1.1-Small instead of T5-Small.

3 Benchmarking Existing Synthetic Pre-training

We benchmark three previously proposed synthetic pre-training tasks across six downstream tasks. In Section 3.1 we describe the three previously proposed synthetic pre-training tasks we evaluate, and in Section 3.2 we explain the results in Table 1 and Appendix Table 4.

3.1 Synthetic Tasks

For each synthetic task, we generated one million data points with a vocabulary of 32K tokens, matching the vocabulary size of Pre-trained T5. We provide input-output examples of each generated synthetic task and the min/max input/output sequence lengths in Appendix E. Details about data generation can be found in the original papers.

LIME LIME is a set of three tasks—Deduct, Induct, and Abduct—inspired by Charles Peirce's three reasoning primitives (Wu et al., 2021). Each task consists of three elements: a rule string, a dictionary that represents substitutions, and a result string that is the result of applying those substitutions to the rule string. The three tasks are then constructed by using two of the three elements as inputs to predict the remaining element.

Nesting Dependency Artificial Language The Nesting Dependency Artificial Language is the Dyck language with multiple bracket types: sentences consist of nesting matching tokens, where half of the vocabulary tokens serve as left brackets and are bijectively matched with tokens in the other half of the vocabulary, which serve as right brackets (Ri & Tsuruoka, 2022). We turn generated text into a sequence-to-sequence task by using the T5 span-corruption pre-training objective (Raffel et al., 2020). The motivation behind the design of the Nesting Dependency Artificial Language is

*We obtained different results than Krishna et al. (2021) on CNNDM-10K because of our choice of optimizer. We were able to match their results when we used Adam like they did in their work instead of Adafactor. We chose to use Adafactor for all of our experiments due to much faster training time.

how "sentences of natural language often have dependency relations where the existence of a certain word can be predictive of another word" (Ri & Tsuruoka, 2022).

Nonsense Summary Krishna et al. (2021) developed the nonsense summarization task by conducting a qualitative analysis of summarization data and identifying 21 elementary operations used to create summaries such as "CopyFirstSentence" and "CopyQuoted." Each synthetic task data point consists of an input that is a nonsense document made of randomly sampled tokens and an output that is the corresponding nonsense summary consisting of applying a random subset of the 21 operations to the document.

3.2 Results

Table 1 shows results of evaluating the synthetic tasks across six downstream tasks, and Appendix Table 4 shows results of ablating the dataset size for retrosynthesis, CNNDM, and SQuAD.

Large gains beyond training from random initialization We observed large gains from all three synthetic tasks we evaluated. Out of these, LIME provided the most benefits across all six downstream tasks. We computed how much of the gap between Random Init and Pre-trained T5 is closed by LIME pre-training as $(\text{LIME} - \text{Random Init})/(\text{Pre-trained T5} - \text{Random Init})$. LIME closed 84.6%, 82.8%, 85.6%, 55.1%, 43.5%, and 48.7% of the gap for CNNDM-10K, MTOP, WebQSP, SQuAD, Code Translation, and Retrosynthesis respectively.

Task-related prior is not required for pre-training benefits For example, LIME pre-training benefits tasks including summarization (CNNDM-10K) and reading comprehension (SQuAD) despite LIME involving no natural language, representing no structure of natural language like the Nested Language does, and not being explicitly designed to reflect the operations used in summarization like the Nonsense Summary task.

Varying degrees of benefits Comparing Nonsense Summary and Nested Language, we see that the two synthetic tasks benefit MTOP, WebQSP, and SQuAD similarly, yet their benefits for the three other tasks are very different. The difference between Nonsense Summary and Nested Language as well as the superiority of LIME show that different synthetic pre-training can have significantly different effectiveness, supporting the possibility that there exists synthetic tasks that provide much greater downstream task benefits.

Short language pre-training One of the advantages argued for synthetic pre-training is its computational efficiency (Wu et al., 2021; Krishna et al., 2021). To explore this potential advantage, we pre-trained our model using the T5 span-corruption pre-training objective on Wikipedia (Guo et al., 2020) for 10K steps, and the result is shown in Table 1 labelled "Wiki-10K". Wikipedia pre-training for 10K steps provided more significant gains than LIME pre-training for 30K steps and Set pre-training for 10K steps. Based on this result, we see no evidence that the evaluated synthetic pre-training tasks provides computational efficiency benefits over natural language pre-training.

Natural language pre-training benefits non-natural language tasks Natural language pre-training, including for the low amount of 10K steps, outperforms synthetic pre-training on the non-natural language tasks of Code Translation and Retrosynthesis.

More gains with smaller datasets For CNNDM, SQuAD, and Retrosynthesis, we sampled training sets of varying sizes and then fine-tuned with randomly initialized, LIME pre-trained, and Pre-trained T5 models. In Appendix Table 4, the CNNDM-10K and CNNDM-100K results as well as the Retrosynthesis-10K and Reotrosynthesis-40K results show that benefits from synthetic (and T5 natural language) pre-training increase as the amount of training data for a task decrease.

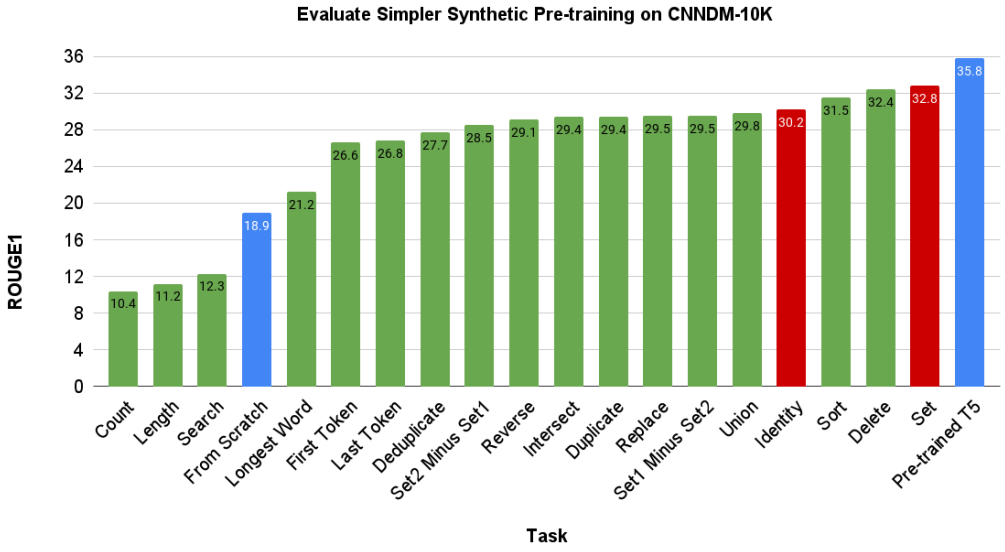

Figure 1: For each simple synthetic task, we pre-train on it and then fine-tune on CNNDM-10K. `Set` performed the best and `Identity`, one of the simplest tasks, provided significant benefit. The `Set` and `Identity` tasks are defined in 4.1, and the other tasks are defined in G.

4 Simpler Synthetic Pre-training

Given encouraging results in the previous section, we want to understand the cause of the benefits brought by synthetic pre-training. In this section, we explore if the complexities of the previously proposed synthetic tasks are necessary for the benefits we saw. We try pre-training with simpler and more generic synthetic tasks, similar to basic Python API functions. We find that two surprisingly simple and generic tasks, `Set` and `Identity`, provided significant benefits.

4.1 Simpler Synthetic Tasks

We constructed 18 simpler synthetic tasks similar to basic Python API functions. For each task, we generated one million data points with a vocabulary of 32K tokens, matching the vocabulary size of Pre-trained T5. In Appendix G.1, we define each task, explain how examples are randomly generated, and provide input-output examples. We first evaluated all tasks on CNNDM-10K. We then evaluated `Set`, one of the best performing tasks on CNNDM-10K, and `Identity`, one of the simplest tasks, on all six downstream tasks.

Set The `Set` task consists of outputting the input sequence in its original order except without tokens appearing more than once. An example input is [a b b a a a c d c], and the corresponding output is [a b c d]. The `Set` task data generation is explained in Appendix G.

Identity The `Identity` task consists of copying the input tokens to the output. An example input is [a d e a a], and the corresponding output is [a d e a a]. To generate one data point, we uniformly sample a sequence of tokens between length 10 and length 220 to be the input and output.

4.2 Results: simpler synthetic tasks provide similar benefits as existing synthetic tasks

Figure 4 shows results on CNNDM-10K for models pre-trained on each of 18 simpler synthetic tasks. Table 1 shows results on all six downstream tasks for `Set` and `Identity` pre-trained models. Averaged across the six downstream tasks, `Set` and `Identity` capture 97% and 70%[1] of the

[1]These percentages are computed by taking the average across six tasks of $\frac{\text{(Set/Identity)} - \text{(Random Init)}}{\text{(LIME)} - \text{(Random Init)}}$

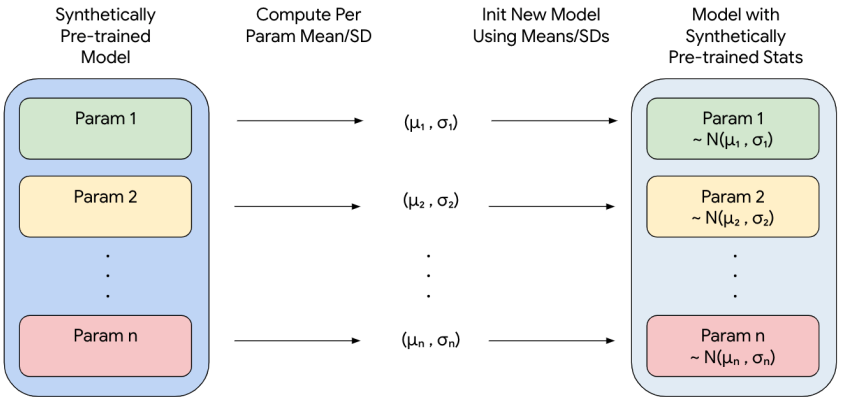

Figure 2: Illustration of Section 5.1 Per Param Mean/SD init scheme. We use this init scheme with LIME, `Set`, or `Identity` pre-trained models, and then we fine-tune on CNNDM-10K or MTOP. For example, using this init scheme with an `Identity` pre-trained model, we initialize the encoder layer 2 query matrix weights by sampling from a Gaussian with the the mean and SD of the encoder layer 2 query matrix from an `Identity` pre-trained model, and we initialize every other parameter in the same way. This init scheme recovered more than 70% of the benefit of normally fine-tuning a synthetic pre-trained model.

improvement over random initialization by LIME pre-training, the best previously proposed synthetic task evaluated in Section 3. `Identity` provides at least as much benefit as Nested Language and Nonsense Summary for all tasks besides Code Translation, for which Nested Language benefits more.

The significant benefits of `Set` and `Identity` pre-training suggest that the complexities of the three previously proposed synthetic tasks we evaluated—representing three fundamental logical operations, having a structural property that mimics natural language, and reflecting operations used for summarization—may not be the necessary factors for why pre-training on those tasks provides benefits. Also, due to their simplicity, `Set` and `Identity` may be useful for future work towards understanding pre-training and how to design better synthetic tasks.

5 Even Simpler Synthetic Pre-training with Coarse Statistics

Inspired by the previous section, we tried to find simpler explanations for the benefits of synthetic pre-training. We explored how benefits of synthetic pre-training can be attained by initializing based only on statistics. We found: (1) Initialization using the mean and standard deviation (SD) of synthetic pre-trained parameters can recover more than 70% of the benefit from synthetic pre-training. (2) Initialization using simpler schemes involving less statistics can still provide benefits. (3) Specifically the pre-attention layer norm plays an important role for initialization benefits.

5.1 How much of synthetic pre-training benefit can be attributed to better statistics for initialization?

In Table 2, we compare normal fine-tuning with fine-tuning a model that is initialized with the mean and standard deviation (SD) of each synthetic pre-trained model's parameter. We refer to this initialization scheme as Per Param Mean/SD and define it in Figure 2. Specifically, for each parameter p (e.g., a weight matrix) in the model, we compute the the mean μ_p and the standard deviation σ_p for all the entries in p. We then randomly reinitialized the model with the same μ_p and σ_p for the parameter p.

Using the Per Param Mean/SD initialization scheme achieves almost the same benefit as normally fine-tuning a synthetic pre-trained model. For example, for LIME, `Identity`, and `Set`, the Per Param Mean/SD initialization recovered 78%, 85%, and 97%[2] of MTOP normal fine-tuning performance.

[2]These percentages are computed by $\frac{\text{(Per Param Mean/SD init)} - \text{(Random Init)}}{\text{(Normal Fine-tuning)} - \text{(Random Init)}}$

Table 2: Evaluating different initialization schemes with statistics of LIME, `Set`, and `Identity` pre-trained models.

	CNNDM-10K			MTOP		
	LIME	Set	Identity	LIME	Set	Identity
Section 5.1						
Normal Fine-tune	33.2	32.8	30.2	73.7/94.0	71.7/93.7	68.6/93.0
Per Param Mean/SD	29.3	29.8	28.8	66.1/92.3	66.6/92.3	67.8/92.6
Section 5.2						
Per Param Scale	29.0	29.5	29.0	67.3/92.5	66.5/92.4	66.3/92.3
Whole Model Scale	15.5	26.1	20.5	34.3/81.4	38.7/83.1	59.8/91.0
Pre-attn LN Per Param Scale	27.8	25.1	27.3	59.9/90.3	58.9/89.0	46.7/86.3

Baseline	CNNDM-10K	MTOP
Random Init	18.9	38.6/83.0
Pre-trained T5	35.8	81.0/95.2

The remaining gap supports the possibility that other factors besides better initialization statistics may cause the benefits of synthetic pre-training. We note that it does not close the possibility that better initialization statistics is the sole cause of benefits because there exist other initialization schemes such that the statistics of the pre-trained model are a subset of the statistics from the initialization scheme.

5.2 Can simpler initialization schemes still produce benefits?

Next we explore if simpler initialization schemes that involve less statistics than Per Param Mean/SD can still produce benefits. First, we describe three initialization schemes we try: Per Param Scale, Whole Model Scale, and Subset Per Param Scale. Then, we explain the results shown in Table 2. We also tried two other initialization schemes, Across Layers Scale and Per Layer Scale, which we define in Appendix H and show results for in Appendix Table 5.

Per Param Scale We use the term "scale statistic" to denote the mean of layer norm parameters or the SD of non layer norm parameters. Unlike the Per Param Mean/SD init scheme in the previous subsection, where we initialize each parameter with two statistics (mean and SD), Per Param Scale initializes each parameter with only the scale statistic: initialize layer norm parameters by setting their value to the mean of the corresponding pre-trained parameter weights, and initialize non layer norm parameters by sampling from a Gaussian with mean 0 and the SD of the corresponding pre-trained parameter weights.

Whole Model Scale We initialize the whole model with only two statistics: the SD over all non layer norm weights and the mean over all layer norm weights.

Subset Per Param Scale The motivation for this init scheme is to find a smaller set of parameters whose initialization can provide significant benefit. We initialize a subset of the parameters with the Per Param Scale initialization and initialize the rest of the parameters with the default random initialization. With this initialization scheme, we tried four different subsets: attention parameters, MLP parameters, pre-attention layer norms, and pre-MLP layer norms.

Results

1. Per Param Scale initialization provided almost identical benefits as the Per Param Mean/SD initialization, suggesting that just the scale statistics, rather than both the mean and SD, are sufficient to provide benefits of initialization.

Table 3: Results from initializing pre-attention layer norm parameters to one value. They are initialized to the value of 1.0 in the "Random Init" baseline which is the default T5 initialization.

Pre-attn LN Init Value	CNNDM-10K	MTOP
0.05	28.2	25.7/73.3
0.1	28.3	32.5/77.5
0.2	28.2	56.4/89.3
0.4	24.4	56.9/89.9
0.6	20.9	50.0/88.2
0.8	18.6	50.7/88.5

Baseline	CNNDM-10K	MTOP
Random Init	18.9	38.6/83.0
Pre-trained T5	35.8	81.0/95.2

2. Whole Model Scale initialization provided few benefits relative to Per Param Mean/SD. With a `Set` pre-trained model, the initialization provided some benefits for CNNDM-10K, and with a LIME pre-trained model, the initialization provided some benefits for MTOP. For the other four pre-training and task combinations, there was almost no benefit over random initialization.

3. For the Subset Per Param Scale init scheme, pre-attention layer norms was the only subset we tried that provided significant initialization benefits. See results in Table 2 (labeled "Pre-attn LN Per Param Scale") and results for all four subsets tried in Appendix Table 5.

5.3 A Better Pre-attention Layer Norm Initialization

Synthetic pre-training results in lower pre-attention layer norm means Motivated by benefits from the Subset Per Param Scale initialization scheme—initializing pre-attention layer norms using their synthetic pre-trained means and all other parameters with the default initialization—we wanted to see if the pre-attention layer norm mean values in synthetic pre-trained models have a pattern. In Appendix Figure J, we plotted pre-attention layer norm parameter means of each layer for LIME, `Set`, and `Identity` pre-trained models. We noticed that the pre-attention layer norm means from the synthetic pre-trained models are much smaller values (below 0.4 for most layers) compared to the default initialization of 1.0.

Lower value pre-attention layer norm initialization provides benefits Inspired by the above observation, we tried an initialization scheme where all the pre-attention layer norm parameters are initialized to one value, and all other parameters are initialized with the default initialization. We tried six different values ranging from 0.05 to 0.8 and fine-tuned on CNNDM-10K and MTOP. Results in Table 3 show that initializing the pre-attention layer norm parameters to a lower value than the default initialization of 1.0 provided significant benefits: initializing them to 0.2 improved CNNDM-10K from 18.9 to 28.2 ROUGE1 and MTOP from 38.6% to 56.4% EM.

6 Related Work

Synthetic Pre-training Past work has shown that synthetic pre-training can benefit a wide range of downstream tasks. Pre-training on artificial languages that intuitively mimic properties of natural language benefit language modeling, dependency parsing, and GLUE (Ri & Tsuruoka, 2022; Papadimitriou & Jurafsky, 2020; Chiang & Lee, 2021). Pre-training on LIME, a simple synthetic Seq2Seq task, benefits mathematical reasoning benchmarks (Wu et al., 2021). Pre-training on a nonsense summarization task and also Zou et al. (2020)'s set of three denoising pre-training tasks applied to a corpus of random tokens benefits summarization benchmarks (Krishna et al., 2021).

Pre-training with synthetically generated fractal images nearly matches the benefits of pre-training with ImageNet (Anderson & Farrell, 2022).

Cross-domain Transfer Learning Synthetic pre-training benefiting downstream tasks across a wide range of domains is an instance of cross-domain transfer learning, which past work has demonstrated the efficacy of. Pre-training on music, code, and amino acid sequences benefits natural language downstream tasks (Papadimitriou & Jurafsky, 2020; Chiang & Lee, 2020). Pre-training on natural language benefits offline RL (Reid et al., 2022). Pre-training language models with many languages benefits downstream tasks in a new unseen language (Liu et al., 2020).

Understanding Pre-training Past work has explored what properties of the pre-training data are important for benefits. Ri & Tsuruoka (2022), Papadimitriou & Jurafsky (2020), and Chiang & Lee (2021) use synthetic pre-training to identify general properties of pre-training data that correlate with improved downstream language task performance by leveraging how synthetic data is easy to modify in a controlled manner. K et al. (2020), Conneau et al. (2020), and Dufter & Schütze (2020) explore data properties related to cross-language transfer learning. Sinha et al. (2021) shows that word order within sentences is not essential for natural langauge pre-training benefits.

Other past work has used probing methods to explore what potentially transferable knowledge is gained from pre-training. Ri & Tsuruoka (2022) explored how much knowledge of "position-aware context dependence" was gained from artificial language and natural language pre-training. Chi et al. (2020) and Papadimitriou et al. (2021) explored potentially transferable knowledge gained from multi-lingual language pre-training.

Initialization Our work tried to understand pre-training through the lens of initialization statistics that benefit optimization. There has been past work on improving transformer initialization to achieve more stable training and benefits with deeper models (Zhang et al., 2019; Huang et al., 2020; Zhang et al., 2020; Wang et al., 2022). These papers do not explore benefits from initializing with pre-trained model statistics.

7 Discussion

Limitations of current synthetic tasks In this work, we show that synthetic pre-training can provide significant benefits across many different tasks. However, synthetic pre-training still lags behind natural language pre-training. This is expected for natural language downstream tasks, but it is important to note that even on non-natural language tasks, natural language pre-training still outperforms synthetic pre-training. This result suggests that natural language pre-training produces effects that are missing from the evaluated synthetic pre-training. Exploring the differences between different pre-training may be an interesting direction for future work.

Synthetic pre-training requires deeper understanding Our results show that initialization using the statistics from synthetic pre-trained models can recover more than 70% of the benefit from synthetic pre-training. The remaining gap leaves open the possibility that the benefits of synthetic pre-training are due to other factors besides better statistics for initialization. Due to their simplicity, `Set` and `Identity` may be useful for future work towards deeper understanding of synthetic pre-training.

In the long term, as we learn more about synthetic pre-training and pre-training in general, we believe that for some downstream tasks it may be possible to develop synthetic pre-training that outperforms natural pre-training: the complexity of existing natural data is fixed, while in some sense the complexity of fully synthetically generated data is infinite.

Efficiency, Privacy, Ethics This work focuses on the understanding side of synthetic pre-training. But there are a lot more potential benefits to synthetic pre-training, including computational efficiency, being devoid of privacy, copyright, and ethics concerns, and more. We believe the general direction of synthetic pre-training has much more to offer than what has been explored in this paper.

References

Anderson, C. and Farrell, R. Improving fractal pre-training. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022*, pp. 2412–2421. IEEE, 2022. doi: 10.1109/WACV51458.2022.00247. URL https://doi.org/10.1109/WACV51458.2022.00247.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., 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. M., 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. Language models are few-shot learners. In *NeurIPS*, 2020.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Chi, E. A., Hewitt, J., and Manning, C. D. Finding universal grammatical relations in multilingual BERT. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 5564–5577. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.493. URL https://doi.org/10.18653/v1/2020.acl-main.493.

Chiang, D. C. and Lee, H. Pre-training a language model without human language. *CoRR*, abs/2012.11995, 2020. URL https://arxiv.org/abs/2012.11995.

Chiang, D. C. and Lee, H. On the transferability of pre-trained language models: A study from artificial datasets. *CoRR*, abs/2109.03537, 2021. URL https://arxiv.org/abs/2109.03537.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. Unsupervised cross-lingual representation learning at scale. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8440–8451. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.747. URL https://doi.org/10.18653/v1/2020.acl-main.747.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

Dufter, P. and Schütze, H. Identifying elements essential for bert's multilinguality. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 4423–4437. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.358. URL https://doi.org/10.18653/v1/2020.emnlp-main.358.

Guo, M., Dai, Z., Vrandečić, D., and Al-Rfou, R. Wiki-40B: Multilingual language model dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 2440–2452, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.297.

Huang, X. S., Pérez, F., Ba, J., and Volkovs, M. Improving transformer optimization through better initialization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020,*

13-18 July 2020, Virtual Event, volume 119 of *Proceedings of Machine Learning Research*, pp. 4475–4483. PMLR, 2020. URL http://proceedings.mlr.press/v119/huang20f.html.

K, K., Wang, Z., Mayhew, S., and Roth, D. Cross-lingual ability of multilingual BERT: an empirical study. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJeT3yrtDr.

Krishna, K., Bigham, J. P., and Lipton, Z. C. Does pretraining for summarization require knowledge transfer? In *EMNLP (Findings)*, pp. 3178–3189, 2021. URL https://aclanthology.org/2021.findings-emnlp.273.

Li, H., Arora, A., Chen, S., Gupta, A., Gupta, S., and Mehdad, Y. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 2950–2962, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.257. URL https://aclanthology.org/2021.eacl-main.257.

Liu, B., Ramsundar, B., Kawthekar, P., Shi, J., Gomes, J., Luu Nguyen, Q., Ho, S., Sloane, J., Wender, P., and Pande, V. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, 3(10):1103–1113, 2017. doi: 10.1021/acscentsci.7b00303. URL https://doi.org/10.1021/acscentsci.7b00303. PMID: 29104927.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. Multilingual denoising pre-training for neural machine translation. *Trans. Assoc. Comput. Linguistics*, 8: 726–742, 2020. URL https://transacl.org/ojs/index.php/tacl/article/view/2107.

Lu, J. and Zhang, Y. Unified deep learning model for multitask reaction predictions with explanation. *Journal of Chemical Information and Modeling*, 62(6):1376–1387, 2022. doi: 10.1021/acs.jcim.1c01467. URL https://doi.org/10.1021/acs.jcim.1c01467. PMID: 35266390.

Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C. B., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., Sundaresan, N., Deng, S. K., Fu, S., and Liu, S. Codexglue: A machine learning benchmark dataset for code understanding and generation. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c16a5320fa475530d9583c34fd356ef5-Abstract-round1.html.

Papadimitriou, I. and Jurafsky, D. Learning music helps you read: Using transfer to study linguistic structure in language models. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6829–6839. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.554. URL https://doi.org/10.18653/v1/2020.emnlp-main.554.

Papadimitriou, I., Chi, E. A., Futrell, R., and Mahowald, K. Deep subjecthood: Higher-order grammatical features in multilingual BERT. In Merlo, P., Tiedemann, J., and Tsarfaty, R. (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pp. 2522–2532. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.215. URL https://doi.org/10.18653/v1/2021.eacl-main.215.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL http://arxiv.org/abs/1606.05250.

Reid, M., Yamada, Y., and Gu, S. S. Can wikipedia help offline reinforcement learning? *CoRR*, abs/2201.12122, 2022. URL https://arxiv.org/abs/2201.12122.

Ri, R. and Tsuruoka, Y. Pretraining with artificial language: Studying transferable knowledge in language models. *CoRR*, abs/2203.10326, 2022. doi: 10.48550/arXiv.2203.10326. URL https://doi.org/10.48550/arXiv.2203.10326.

See, A., Liu, P. J., and Manning, C. D. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099.

Sinha, K., Jia, R., Hupkes, D., Pineau, J., Williams, A., and Kiela, D. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In Moens, M., Huang, X., Specia, L., and Yih, S. W. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 2888–2913. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.230. URL https://doi.org/10.18653/v1/2021.emnlp-main.230.

Wang, H., Ma, S., Dong, L., Huang, S., Zhang, D., and Wei, F. Deepnet: Scaling transformers to 1, 000 layers. *CoRR*, abs/2203.00555, 2022. doi: 10.48550/arXiv.2203.00555. URL https://doi.org/10.48550/arXiv.2203.00555.

Wu, Y., Rabe, M. N., Li, W., Ba, J., Grosse, R. B., and Szegedy, C. Lime: Learning inductive bias for primitives of mathematical reasoning. In *International Conference on Machine Learning*, pp. 11251–11262. PMLR, 2021.

Xie, T., Wu, C. H., Shi, P., Zhong, R., Scholak, T., Yasunaga, M., Wu, C., Zhong, M., Yin, P., Wang, S. I., Zhong, V., Wang, B., Li, C., Boyle, C., Ni, A., Yao, Z., Radev, D. R., Xiong, C., Kong, L., Zhang, R., Smith, N. A., Zettlemoyer, L., and Yu, T. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *CoRR*, abs/2201.05966, 2022. URL https://arxiv.org/abs/2201.05966.

Yih, W.-t., Richardson, M., Meek, C., Chang, M.-W., and Suh, J. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 201–206, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2033. URL https://aclanthology.org/P16-2033.

Zhang, B., Titov, I., and Sennrich, R. Improving deep transformer with depth-scaled initialization and merged attention. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 898–909. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1083. URL https://doi.org/10.18653/v1/D19-1083.

Zhang, B., Williams, P., Titov, I., and Sennrich, R. Improving massively multilingual neural machine translation and zero-shot translation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 1628–1639. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.148. URL https://doi.org/10.18653/v1/2020.acl-main.148.

Zou, Y., Zhang, X., Lu, W., Wei, F., and Zhou, M. STEP: sequence-to-sequence transformer pre-training for document summarization. *CoRR*, abs/2004.01853, 2020. URL https://arxiv.org/abs/2004.01853.

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]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Sections 2,3.1,4.1,G,D
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) 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] Appendix Section I
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]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]