Sequence Reducible Holdout Loss for Language Model Pretraining

Raghuveer Thirukovalluru[†], **Nicholas Monath**[‡], **Bhuwan Dhingra**[†], **Sam Wiseman**[†]
[†] Duke University
{raghuveer.thirukovalluru, bhuwan.dhingra, swiseman}@duke.edu
[‡] Google DeepMind
nmonath@google.com

Abstract

Data selection techniques, which adaptively select datapoints inside the training loop, have demonstrated empirical benefits in reducing the number of gradient steps to train neural models. However, these techniques have so far largely been applied to classification. In this work, we study their applicability to language model pretraining, a highly time-intensive task. We propose a simple modification to an existing data selection technique (*reducible hold-out loss* training) in order to adapt it to the sequence losses typical in language modeling. We experiment on both autoregressive and masked language modelling, and show that applying data selection to pretraining offers notable benefits including a 4.3% reduction in total number of steps, a 21.5% steps reduction in average, to an intermediate target perplexity, over the course of pretraining an autoregressive language model. Further, data selection trained language models demonstrate significantly better performance on out of domain datasets, including 7.9% reduction in total number of steps and 23.2% average steps reduction to an intermediate target perplexity.

1. Introduction

Data selection methods can often reduce the number of steps required to train neural models (Jiang et al., 2019; Kawaguchi and Lu, 2020; Mindermann et al., 2022). These methods typically evaluate the loss of a large number of examples under the model first and then selectively perform backward passes on the examples with either the highest loss, the largest gradient or the maximum expected improvement on a held out set (Loshchilov and Hutter, 2015; Jiang et al., 2019; Kawaguchi and Lu, 2020). Ideally, these data selection techniques could reduce training time for models that use a lengthy training process. Pretrained language models have seen tremendous success in many NLP tasks (Devlin et al., 2018; Chowdhery et al., 2022; OpenAI, 2023). Pretraining is an expensive process (Sharir et al., 2020) that typically happens on billions of tokens, for months at a time (Sevilla et al., 2022). In this paper, we explore whether data selection techniques, that have largely been studied in the context of classification, can also be applied to pretraining language models.

Typical training objectives for neural language models (LMs) include log likelihood of the next (or masked) token (Raffel et al., 2020). To implement these objectives efficiently, we compute these per-token objectives for all tokens in a given sequence. This presents a challenge for data selection techniques, which are less well suited for selecting entire sequences of examples. In order for data selection to be more efficient, we must throw out entire sequences rather than tokens. Token-level objectives must be aggregated into sequence-level measures of the utility of training examples when deciding which ones to throw out.

In this work, we adapt a recently proposed technique of Reducible Hold Out Loss (RHO-Loss) selection (Mindermann et al., 2022) to the sequence level by averaging over token-level losses. We show that data selection on pretraining results in (1) Notable reduction in number of training steps, and (2) LMs with significant better performance on out of domain datasets compared to standard pretraining.

2. Background

2.1. RHO-Loss

Consider training a neural model P_t on a dataset of labeled examples $\mathcal{D}$=$\{(x_1, y_1), (x_2, y_2), ...\}$. We refer to this model of interest, P_t, as the *target* model. While typical data selection methods pick the examples with high target loss (Kawaguchi and Lu, 2020), Mindermann et al. (2022) considers an additional auxiliary loss alongside the target loss. This auxiliary loss comes from

LREC-COLING 2024, pages 14705–14716

a different model P_{IL} pretrained on $\mathcal{D}_{ho}$, a small heldout portion of $\mathcal{D}$. P_t is exclusively trained on the remaining portion of $\mathcal{D}$ i.e., $\mathcal{D}_t$. Intuitively, a high loss under P_{IL} of an example in $\mathcal{D}_t$ suggests it is noise because it falls out of the distribution of $\mathcal{D}_{ho}$. Similarly, a low loss under P_t of an example suggests that it is redundant and not important. RHO-Loss uses a selection criterion following these principles.

Specifically, at each step of training P_t, Mindermann et al. (2022) first makes a forward pass on minibatch $\mathcal{B}_B$ with n_B number of examples i.e. $n_B = |\mathcal{B}_B|$. Then RHO-Loss is defined as the difference between the target loss and IL loss as shown in Eq. 1. Negative log likelihood is used for individual model losses. RHO-Loss is used to select the n_b (effective minibatch size) highest scoring examples among the n_B (n_B:n_b ratio is typically 10:1). A backward pass is then performed on the selected n_b examples to train P_t. P_{IL} is also called the 'Irreducible Loss' (IL) model.

$$\underset{(x,y)\in\mathcal{B}_B;\quad k=n_b}{\text{argtop-}k} \log P_{\text{IL}}(y|x) - \log P_t(y|x) \quad (1)$$

Evaluation: Mindermann et al. (2022) used steps to target accuracy as the main evaluation metric. Similarly, we use steps to target perplexity/loss as our main metric. Note that this metric ignores the cost of extra forward passes to get IL loss, the IL training cost. We, along with Mindermann et al. (2022) believe that costs are less important and can be effectively amortized. Elaborate details in §7.

2.2. Language Modelling

Let $\mathbf{s}^{(j)} = x_0^{(j)}, x_1^{(j)}, \ldots, x_m^{(j)}$ be the m tokens of j-th text sequence in dataset $\mathcal{D}$. P_t is the language model to be trained. $\mathcal{M}^{(j)}$ is a set of tokens of interest in $\mathbf{s}^{(j)}$ which contribute to the loss. $\mathbf{s}_i^{(j)}$ is a modified sequence that is used to predict token $x_i^{(j)}$. Language modelling loss for the minibatch $\mathcal{B}$ comprising n_b examples is

$$\mathcal{L} = -\frac{1}{Z}\sum_{j\in\mathcal{B}}\sum_{x_i^{(j)}\in\mathcal{M}^{(j)}} \log P_t(x_i^{(j)}|\mathbf{s}_i^{(j)}), \quad (2)$$

where $Z = \sum_{j\in\mathcal{B}}|\mathcal{M}^{(j)}|$.

2.2.1. Autoregressive Modelling (AM)

All tokens in each example contribute to the loss in this setting. Hence, $|\mathcal{M}^{(j)}| = m$ where m is the sequence length. We assume all sequences are tightly packed (Raffel et al., 2020). At each step, an autoregressive LM predicts token $x_i^{(j)}$ given past tokens $x_0^{(j)}, x_1^{(j)}, .., x_{i-1}^{(j)}$. Hence, $\mathbf{s}_i^{(j)} = x_0^{(j)}, x_1^{(j)}, .., x_{i-1}^{(j)}$ in this case.

2.2.2. Masked Language Modelling (MLM)

In MLM, multiple tokens in the input sequence are replaced by the [MASK] token. Let $\mathbf{s}_m^{(j)}$ be the masked sequence. An LM predicts masked token $x_i^{(j)}$ using this masked sequence $\mathbf{s}_m^{(j)}$. Hence $\mathbf{s}_i^{(j)} = \mathbf{s}_m^{(j)}$ in this case. $\mathcal{M}^{(j)}$ contains the mask tokens in sequence $\mathbf{s}_m^{(j)}$. In MLM, a random 15% tokens in every sequence are masked, few are replaced with original/random words (Devlin et al., 2018).

2.3. Computational Challenges of Data Selection for Language modelling

Eq. 2 shows that LM pretraining involves backpropagating loss from multiple *tokens* in a minibatch. A trivial way of applying data selection techniques would be to select tokens based on loss/RHO criterion and performing backward pass on them. Unfortunately, discarding individual tokens does not give any significant speed up because attention would still require activation values of the dropped tokens. Hence, sequences need to be dropped.

3. Methodology

In this section, we present some simple modifications to RHO-Loss that enables it to work for pretraining of language models. Following RHO-Loss, we split dataset $\mathcal{D}$ into two parts $\mathcal{D}_t$ and $\mathcal{D}_{ho}$. The auxiliary model P_{IL} is pretrained on the smaller $\mathcal{D}_{ho}$. Irreducible loss per token becomes $-\log P_{\text{IL}}(x_i^{(j)}|\mathbf{s}_i^{(j)})$. Using target model P_t, we define the reducible holdout loss per token $x_i^{(j)}$ as:

$$\mathcal{L}_{\text{RHO}}^{(j)}(i) = \log P_{\text{IL}}(x_i^{(j)}|\mathbf{s}_i^{(j)}) - \log P_t(x_i^{(j)}|\mathbf{s}_i^{(j)}). \quad (3)$$

Selecting high worth tokens and backpropagating on them is not computationally efficient. We hence perform a data selection run at the sequence level first and then apply a token level pretraining loss over the selected sequences. We present Sequence Reducible

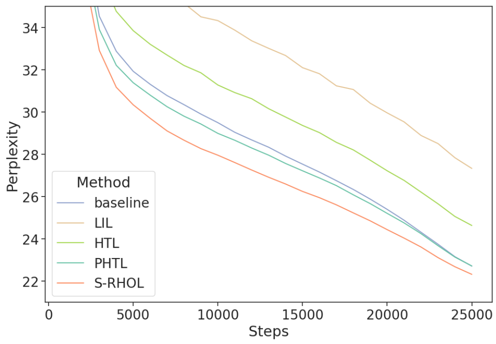

Figure 1: In domain validation ppl for AM. S-RHOL achieves -4.3% in %ΔSteps(FINAL) and -21.5% in %ΔSteps(MEAN) over baseline.

Holdout Loss (S-RHOL) for sequence $\mathbf{s}^{(j)}$ as

$$\mathcal{L}^{(j)}_{\text{S-RHO}} = F\left(\left\{\mathcal{L}^{(j)}_{\text{RHO}}(i) | x^{(j)}_i \in \mathcal{M}^{(j)}\right\}\right) \quad (4)$$

where $\mathcal{M}^{(j)}$ is the set of tokens of interest in sequence $\mathbf{s}^{(j)}_i$. All the tokens in the sequence $\mathbf{s}^{(j)}_i$ are in $\mathcal{M}^{(j)}$ for autoregressive modelling. All mask tokens in masked sequence $\mathbf{s}^{(j)}_m$ are in $\mathcal{M}^{(j)}$ for masked language modelling.

For F, we experimented with multiple choices - `mean`, `median`, `quartiles`. In both AM and MLM, we found `mean` performed better than others. Thus, S-RHOL of $\mathbf{s}^{(j)}$ is equal to the average value of reducible holdout loss of its tokens in AM, and of its masked tokens in MLM. Given S-RHO for $\mathbf{s}^{(j)}$, we can now select non-redundant, non-noisy sequences from $\mathcal{D}_t$.

For training P_t, the first forward pass is made on a minibatch $\mathcal{B}_B$ of n_B sequences and their S-RHO values are determined. We pick the top n_b examples by the highest value of $\mathcal{L}^{(j)}_{\text{S-RHO}}$:

$$\underset{\mathbf{s}^{(j)} \in \mathcal{B}_B; \quad k=n_b}{\text{argtop-}k} \quad \mathcal{L}^{(j)}_{\text{S-RHO}}. \quad (5)$$

These n_b examples together form the batch $\mathcal{B}$. The second forward pass is performed on $\mathcal{B}$ and is then followed by a backward run on the same. This part uses the exact token level language modelling loss as described in Eq. 2. This completes one single step of pretraining.

4. Experiments

We compare the following methods. **`baseline`** - The standard pretraining procedure from Eq. 2. **S-RHOL** - The method proposed in §3. **LIL** - This method is similar to S-RHOL but without the target loss component in Eq. 3. **HTL** - This again is similar to S-RHOL but without the IL loss component in Eq. 3. **PHTL** - This is an softer version of HTL. This method probabilistically picks sequences by softmax of their loss values like Jiang et al. (2019).

We use BERT-BASE($110M$ params) (Devlin et al., 2018) for MLM and GPT2-SMALL ($124M$ params) (Radford et al., 2019) for AM. We use the same architecture for P_{IL}(Mindermann et al., 2022). For AM, we follow (Gururangan et al., 2022) to combine 1B (Chelba et al., 2013), MED, CS (Lo et al., 2019) and REALNEWS (Zellers et al., 2019) resulting in $20.6B$ tokens. For MLM, we use wikipedia and bookcorpus (Devlin et al., 2018). To make the runs more tractable, we limit sequences to 128 length and training to $25K$ steps for all models (Izsak et al., 2021). Warmup is 8% of the total steps. Effective batch size was 4096 for both AM and MLM. . Following Izsak et al. (2021), we set learning rate schedule to go down to zero at pretraining end. The IL model has been trained for $75K$ steps on 30% data. While we agree that 75K is a large number of steps, we point out that the data used is limited. Mindermann et al. (2022) also showed that smaller sized IL models work reasonably well. Some of our later experiments also analyse the affect of strength of IL models.

4.1. Results: Autoregressive Modelling

Fig. 1, Table 1 show perplexity values on the validation set. Intermediate checkpoints during pretraining are essential in many applications. Hence, we also report MEAN intermediate numbers over the course of pretraining.

S-RHOL surpassed all other models. S-RHOL requires 4.3% fewer number of steps to achieve the final perplexity achieved by the

Models	**Perplexity**	ΔPerplexity (MEAN)	ΔPerplexity (FINAL)	%ΔSteps (MEAN)	%ΔSteps (FINAL)
baseline	22.7	0	0	0	0
LIL	27.3	4.9	4.6	85.9	-
HTL	24.6	1.9	1.9	33.9	-
PHTL	22.7	-0.4	0	-6.3	-0.1
S-RHOL	22.3	-1.3	-0.4	**-21.5**	**-4.3**

Table 1: In domain performance. Δ, %Δ are improvement, % improvement, over `baseline` resp. MEAN is the value averaged over the course of pretraining(every $1K$ steps of `baseline`), FINAL is value at end of pretraining.

Models	ΔPerplexity	ΔPerplexity		%ΔSteps	
	(GPT2)	(MEAN)	(FINAL)	(MEAN)	(FINAL)
S-RHOL	-12.43	-3.33	-1.44	**-23.2**	**-7.9**

Table 2: Average Out of Domain AM results. S-RHOL improvement better than In-Domain.

baseline. The gap between S-RHOL and baseline is much higher at an intermediate steps(<$25K$). Given a target perplexity, S-RHOL achieves it in 21.5% fewer steps on average as compared to the baseline. (for Ex. S-RHOL requires $\sim 6K$ steps to match the baseline ppl at $10K$ steps, a 40% improvement). Notably, LIL and HTL implementing subparts of S-RHOL performed much worse than baseline. We posit this is because of them not filtering out redundant and noisy examples respectively.

To further measure models' generalization to challenging data distributions, we tested the trained models on 6 out of domain datasets from Gururangan et al. (2022). S-RHOL strongly outperformed the baseline (Tab. 2). Pre-trained GPT2-SMALL perplexities are shown to justify the strength of the trained models.

4.2. Results: Masked Language Modelling

We perform S-RHOL pretraining over MLM models in Tab. 3. Tab. 4 shows the out of domain validation results. S-RHOL outperforms baseline and trends follow the AM case.

Finetuning: While S-RHOL significanty beats baseline on perplexity metrics over multiple in/out of domain datasets, Tab. 5 shows that S-RHOL only does comparably well with baseline on downstream datasets. We posit this is because finetuning metrics do not always correlate to strength of pretraining checkpoints. Research shows pretraining on just the finetuning datasets gets very strong finetuning numbers (Krishna et al., 2022).

Models	Final Loss	ΔFinal Loss		%ΔSteps	
		(MEAN)	(FINAL)	(MEAN)	(FINAL)
baseline	1.818	0	0	0	0
HTL	1.964	0.111	0.146	47.8	-
PHTL	1.832	-0.008	0.009	-2.5	-
S-RHOL	1.81	-0.043	-0.007	**-15.76**	**-4.77**

Table 3: In domain MLM performance. S-RHOL outperforms baseline.

Models	ΔFinal Loss		%ΔSteps	
	(MEAN)	(FINAL)	(MEAN)	(FINAL)
S-RHOL	-0.06	-0.02	**-16.46**	**-6.29**

Table 4: Average Out of Domain MLM performance. S-RHOL outperforms baseline.

Models	mnli	sst2	cola	mrpc	avg
baseline	82.0	91.5	56.9	88.7	79.8
S-RHOL	82.0	91.7	57.7	88.4	80.0

Table 5: Finetuning results with final MLM checkpoints. Both models are comparable.

4.3. Ablations

4.3.1. P_{IL} data size

Although setting aside 30% of $\mathcal{D}$ for $\mathcal{D}_{ho}$ is reasonable for large web scale datasets, this is not possible on smaller domains. Hence, we try with only 10% data for training our IL model. S-RHOL not only outperforms the baseline but also maintains the %Δ gains it achieved in the 30% IL split case. Table 6 shows results.

4.3.2. Hyperparameters

Peak learning rate, learning rate schedule are two important pretraining hyperparameters (Devlin et al., 2018). With multiple peak lr ($2\times$, $0.5\times$, ..), S-RHOL consistently outperformes baseline. We also experiment with $2\times$ schedule (lr goes down to zero at $50K$ steps, training still for $25K$ steps). Final gap with the baseline is much higher in this case suggesting baseline trains much quicker in the later stages when learning rate is closer to zero. Table 6 shows results. Details in §A.5, A.6.

4.3.3. Switchback to baseline

S-RHOL gap with baseline decreases quickly towards the end of pretraining in Fig. 1. To check if baseline can replace S-RHOL later stages in pretraining, we initialize an LM with S-RHOL checkpoint at $5K$ steps and continue standard pretraining for $20K$ more steps. FINAL gain vanishes (MEAN gain still high because of $5K$ S-RHOL steps). Hence, it is beneficial to continuously train with S-RHOL to effectively remove noise, redundancy. Table 6 shows results. More details in §A.7.

Models	%ΔSteps (MEAN)	(FINAL)
S-RHOL(10% data P_{IL})	-20.4	-4.1
S-RHOL(2x Wider LR schedule)	-33.8	-35.4
S-RHOL(Avg. Multiple Learning rates)	-16.7	-5.1
S-RHOL($5K$) + baseline($20K$)	-8.9	-0.5

Table 6: Ablations S-RHOL vs baseline. S-RHOL beats baseline in multiple settings.

Models	Perplexity	ΔPerplexity (MEAN)	(FINAL)	%ΔSteps (MEAN)	(FINAL)
S-RHOL	22.3	-1.3	-0.4	-19.9	-4.7
S-RHOL+	**22.2**	**-1.4**	**-0.5**	**-21.7**	**-5.2**

Table 7: S-RHOL+ uses weak IL models during the beginning of training and moves to stronger IL models towards the end.

4.3.4. Strength of IL model

The auxiliary IL model used in S-RHOL is trained on the holdout dataset $\mathcal{D}_{ho}$. One can train IL models of different strengths on this heldout dataset. While stronger IL models are expected to perform better, we empirically found that weaker IL models may be better suited in the initial steps of pretraining. Fig. 6 shows two pretraining runs of S-RHOL: one with weak IL model and another with strong IL model. Weaker IL models perform better than stronger IL models in the initial phase of training.

Factoring in this phenomenon, we try using IL models of different strength during different stages of training. While pretraining the IL model on $\mathcal{D}_{ho}$, we save multiple checkpoints of it throughout the training run. The weak early IL checkpoints are used in the initial stages of the target training and stronger checkpoints are used in the later stages. Using three IL models of increasing strength during the course of pretraining, we obtain improved results as shown in Table 7. S-RHOL+ the new method performs better than S-RHOL.

5. Related Work

Architectural changes for efficient pretraining: Multiple papers propose to reduce language model pretraining times by changing the model architecture. Liao et al. (2022) work on improving MLM runtime by dropping mask tokens from the initial LM layers and introducing them in the later layers. The squared dependency on inputs in a transformer helps them save pretraining runtime. Hou et al. (2022) is another similar work which drops unimportant tokens from intermediate layers to improve pretraining runtime. Geiping and Goldstein (2022) proposes a range of architectural and optimization tricks to train language models more efficiently. In contrast to these works, the presented S-RHOL functions at a data level to improve pretraninig runtime.

Data changes for efficient pretraining: Lee et al. (2021) is one such work which deduplicates the pretraining data and thereby improves pretraining runtime. While this could be a good preprocessing step, there is much scope left to explore the best strategies for removing examples during the actual pretraining procedure. This work makes considerable progress in this space.

Data Selection: On the other end of the spectrum, data selection is a widely studied class of methods. Jiang et al. (2019) evaluates the loss of an example over multiple epochs and selectively backpropagates on high loss examples. Kawaguchi and Lu (2020) also uses high loss examples while being less selective at the beginning and very selective towards the end of training. Loshchilov and Hutter (2015) is another work which focuses on high loss examples to train target models. Selecting high loss examples although filters redundant examples, is still prone to noisy examples. Some works Pleiss et al. (2020); Chen et al. (2019) can filter out noisy points.

6. Conclusion

In this paper, we introduce simple modifications to RHO-Loss, enabling it to work on sequences. S-RHOL demonstrates notable gains over the standard pretraining baseline on both autoregressive and masked language modelling. Also, S-RHOL consistently outperforms the baseline under multiple settings: weaker IL models, different learning rates and learning rate schedules. Further, S-RHOL trained LMs demonstrate significantly better generalization abilities compared to regular LMs.

7. Limitations

We followed (Mindermann et al., 2022) in evaluating the effectiveness of our dataselection algorithm based on the number of backward steps(and not wallclock time). Our main goal from this paper is to point out to the NLP community that data selection techniques are infact helpful for language model pretraining. We hope this work motivates more exploration of this underexplored topic on datasection for pretraining.

The main limitation of this work is the extra clocktime required by additional forward passes in Eq. 4. An implementation which can leverage data selection principles to improve pretraining clocktime is beyond the scope of this paper and a topic for future research. Our work was focused on the data selection algorithm's effectiveness rather than the wall clock efficiency. Nevertheless, we elaborately discuss ways to nullify this extra clocktime.

Multiple techniques can be used to reduce this time required by the selection forward.

Parallelized selection: Forward passes can be performed on multiple machines in parallel. In contrast, using multiple machines has diminishing returns for a backward pass (Anil et al., 2018; McCandlish et al., 2018). Further, very high batch sizes on backward hurts generalization performance (Shoeybi et al., 2019). Thus data selection can be thought of as one way to make use of additional computation (i.e., to make use of additional GPUs) without harming generalization.

Pipelining: (Jiang et al., 2019) suggest a possible method of using stale versions of the target model to select examples on a separate engine and updating the stale version every few thousand steps . Selection for batch 'N+1' can be done when target trains for batch 'N'. Thus, all delays from the extra forward step can be made zero.

Inference Accelerators: Using low precision/quantization can speed up inference by 10x (Jouppi et al., 2017). Note that activation values are not required for the selection forward pass as opposed to backward passes. Parallelized selection, Pipelining and Inference acceleration can nullify the extra time for selection forward, matching the S-RHOL cost to that of the baseline.

Regarding IL model, we followed Mindermann et al. (2022) in using the same sized IL model. While baseline and SRHOL were trained for 25k steps, IL model was trained for 75k steps(on a much smaller dataset - 30%, 10% of the pretraining data). Mindermann et al. (2022) also showed that smaller sized IL models work reasonably well. We can utilize this to further lower the cost of IL models. Further, this training can all be performed offline. Inference using IL can be performed offline or 'Pipelined' as mentioned before. IL training can thus be considered as preprocessing of the dataset. Further, each IL model can be used to perform multiple trainings amortizing it's cost. All runs in Tab. 6 (except the 10% data P_{IL}) use checkpoints of the same IL pretraining run.

Future research can consider using existing pretrained LM checkpoints (Ex: BERT, ROBERTA, GPT35, etc) as IL models. Data selection can thus be thought of as a reverse of distillation where information from a smaller model can train a larger model. Further, intermediate checkpoints of the same target model can possibly be used as IL models, to completely nullify the extra cost of training IL models and matching the SRHOL cost to that of the baseline.

Broader Impact and Discussion of Ethics: While our model is not tied to any specific applications, it could be used in sensitive contexts such as health-care, etc. Any work using our method is requested to undertake extensive quality-assurance and robustness testing before applying in their setting. To the best of our knowledge, the datasets used in our work do not contain any sensitive information.

Replicability:
Sourcecode: https://github.com/raghavlite/fast-pt

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. 2018. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. 2019. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070. PMLR.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

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*.

Jonas Geiping and Tom Goldstein. 2022. Cramming: Training a language model on a single gpu in one day. *arXiv preprint arXiv:2212.14034*.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. 2021. Demix layers: Disentangling domains for modular language modeling. *arXiv preprint arXiv:2108.05036*.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576, Seattle, United States. Association for Computational Linguistics.

Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. 2022. Token dropping for efficient bert pretraining. *arXiv preprint arXiv:2203.13240*.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*.

Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. 2019. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.

Kenji Kawaguchi and Haihao Lu. 2020. Ordered sgd: A new stochastic optimization framework for empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*, pages 669–679. PMLR.

Kundan Krishna, Saurabh Garg, Jeffrey P Bigham, and Zachary C Lipton. 2022. Downstream datasets make surprisingly good pretraining corpora. *arXiv preprint arXiv:2209.14389*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*.

Baohao Liao, David Thulke, Sanjika Hewavitharana, Hermann Ney, and Christof Monz. 2022. Mask more and mask later: Efficient pre-training of masked language models by disentangling the [mask] token. *arXiv preprint arXiv:2211.04898*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S Weld. 2019. S2orc: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782*.

Ilya Loshchilov and Frank Hutter. 2015. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. 2022. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR.

OpenAI. 2023. Gpt-4 technical report.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. 2020. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 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. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. 2022. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

A. Appendix

A.1. Details

Architecture

For our target model P_t, we use BERT-BASE($110M$ params) architecture (Devlin et al., 2018) for MLM and GPT2-SMALL ($124M$ params) (Radford et al., 2019) architecture for autoregressive language modelling. Given these are fairly small architectures, we use the same architetures for auxiliary model P_{IL}.

Data

Autoregressive Modelling: We combine data from 4 different domains listed in Gururangan et al. (2021). We use 1B (Chelba et al., 2013), MED (Lo et al., 2019), CS (Lo et al., 2019) and REALNEWS (Zellers et al., 2019) in the proportions as listed in Gururangan et al. (2021). The final training dataset comprises $20.6B$ tokens. All models are trained on this dataset. To create a holdout set for training IL models, 30% of this data (around $6.1B$ tokens) is set aside. A small amount of the remaining data is set aside as validation set.

Masked Language modelling: We follow Izsak et al. (2021) and Devlin et al. (2018) in combining Wikipedia and Bookcorpus. Similar to the previous case, we set 30% of this data aside as a heldout dataset.

Training

To make the pretraining more tractable for our budget, we limit sequences to 128 length for all the models. Izsak et al. (2021); Devlin et al. (2018) do this for 100% and 90% of their training respectively. Further, we restrict all method trainings to $25K$ steps similar to (Izsak et al., 2021). We fix the warmup at 8% of the total steps. Effective batch size was 4096 for both AR and MLM. All pretraining runs were performed on a node of 4 A6000 gpus. Following Izsak et al. (2021), we set the learning rate schedule to go down to zero at the end of training.

A.2. Results: Autoregressive Modelling

In Domain: Discussed in Fig. 1, Table 1.

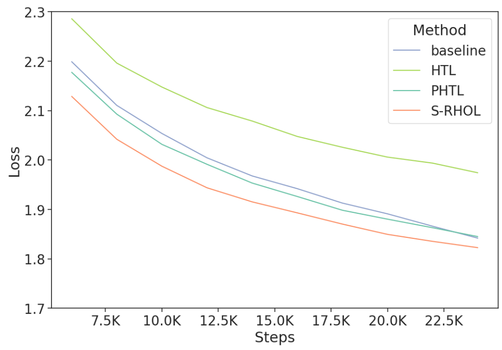

Figure 2: In domain validation loss for MLM. S-RHOL outperforms baseline.

Out of Domain: We also test the models on some out of domain datasets described in Gururangan et al. (2022). We further show the perplexity of a pretrained GPT2-SMALL in the plots for an enriched comparison. This serves multiple purposes. Firstly, it shows how S-RHOL and baseline generalise to out of distribution data. It also justifies that these trained models are enough strong (compared to GPT2).

Fig 3 shows the result. S-RHOL outperforms baseline in all the plots. Further, S-RHOL performs comparably to GPT2 justifying strength of the pretrained models. Note that we used a pretrained GPT2 trained on 1024 sequences and applied it to 128 length sequences.

A.3. Results: Masked language Modelling

In Domain: Table 3, Fig. 2 shows the comparison of S-RHOL with the other methods. Note that the figure depicts loss and not perplexity like in previous cases and hence is on a much more magnified scale. S-RHOL outperforms baseline in all of the experiments.

Out of Domain: Fig. 4 shows the out of domain validation losses similar to AR case. S-RHOL does better than the baseline on all the six datasets.

Finetuning: Discussed in Table 5.

A.4. Ablation: IL Model Strength

Fig 6 shows the pretraining process for different strength IL models.

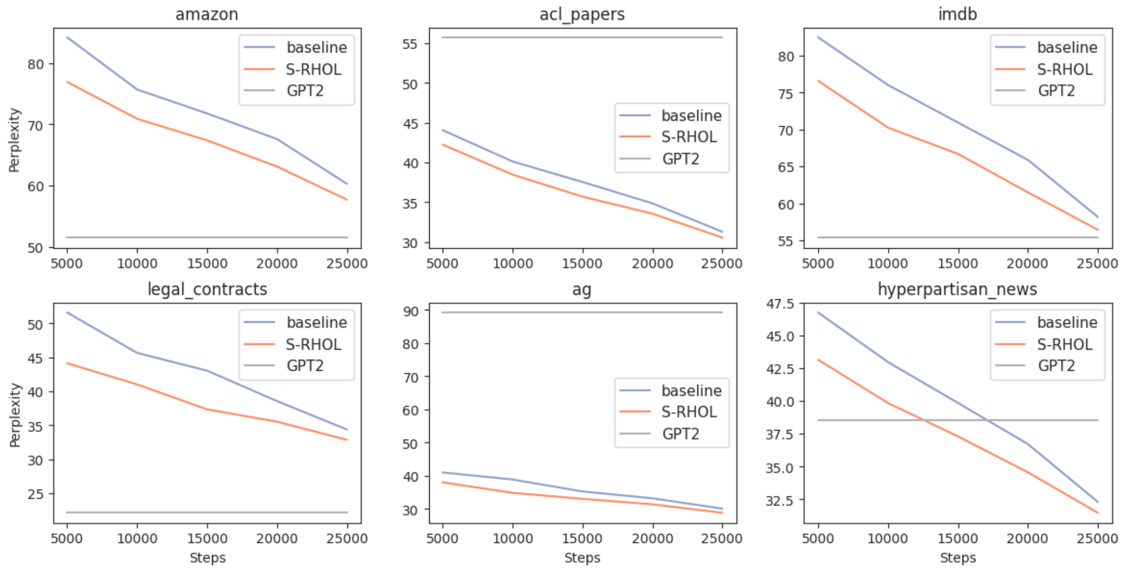

Figure 3: Out of Domain Perplexity values for six out of domain datasets. `S-RHOL` is better than `baseline` for all the six datasets. AM

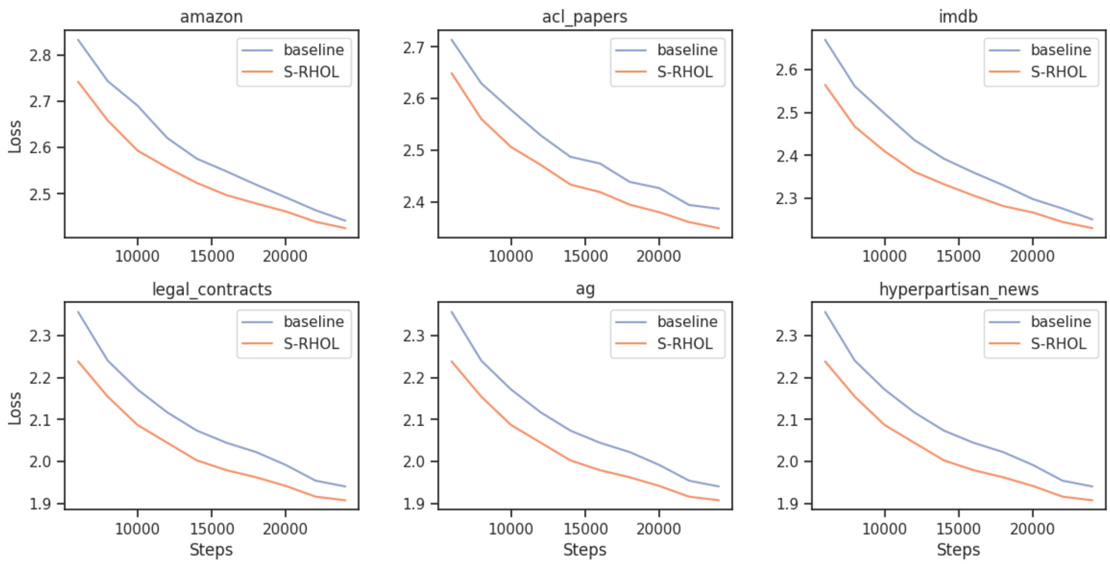

Figure 4: Out of Domain validation losses for six datasets. `S-RHOL` is better better than `baseline`.

A.5. Ablation: Longer Learning rate schedule

Language model pretraining typically happens for a large number of steps. While we could only train our models for 25K steps, to further understand how `S-RHOL` performs on a much large scale pretraining routine, we train models with much longer learning rate schedule in Fig. 7. The learning rate goes down to zero at $50K$ steps(instead of $25K$ as in Table 1) and warmup is at 8% of it. Note that the `%ΔSteps(FINAL)` in this case is much higher at -38.8%. This illustrates that the learning rate schedule contributes to the difference between `baseline` and `S-RHOL`. Further, this shows that one cannot directly interpolate `baseline` and `S-RHOL` curves from Fig. 1 because training for more steps would require a different learning rate schedule. Also note that the final perplexity of both models is much higher when compared to the smaller schedule case indicating that the training is still ongoing.

A.6. Ablation: Effect of Learning Rate

Note that `S-RHOL` is a strategy/methodology and not a single model. A good training strategy is expected to be robust across learning rates. To further establish the superiority of

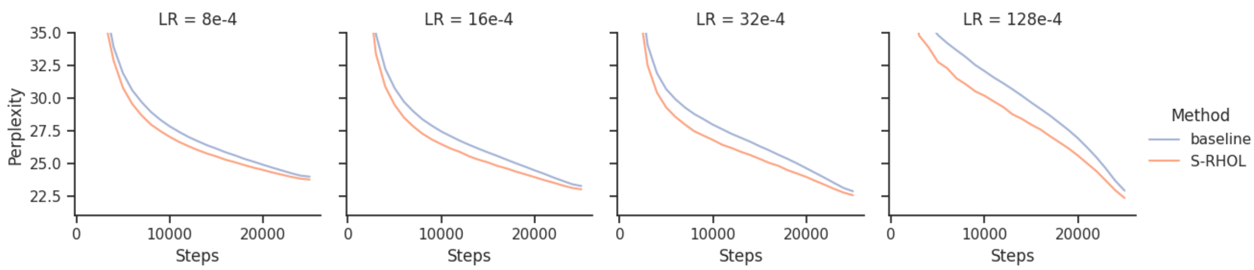

Figure 5: Affects of changing peak learning rate. In all the cases, `S-RHOL` is better than `baseline`.

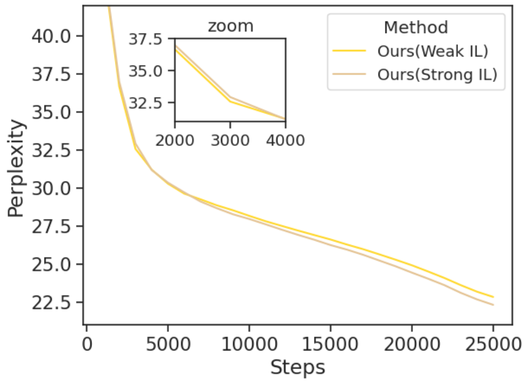

Figure 6: `S-RHOL`(Weak IL) does better in the initial stages until 5000 steps of training and worse in the later stages beyond 5000 steps.

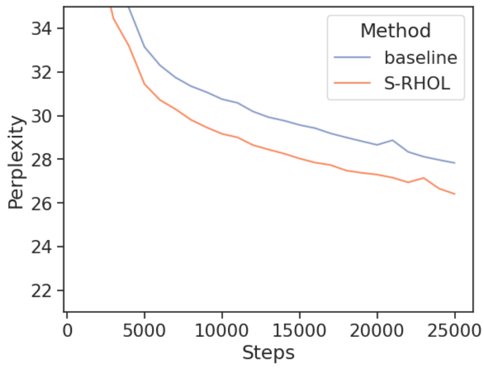

Figure 7: `baseline` and `S-RHOL` on a much wider lr schedule. `%ΔSteps(FINAL)` here is -38.8%

`S-RHOL` over the baseline, we perform rigorous testing of our methods across different learning rates. The results in Table 1 correspond to learning rate of $64e-4$. Fig 5 shows the comparison at multiple other learning rates. We use the same IL model for all of these runs. In all of the 4 different variants, `S-RHOL` performed better than baseline. In general, we observe that the model learns faster towards the end of pretraining cycle when the learning rate is going down towards zero. As seen in Fig. 1, the difference between `S-RHOL` and `baseline` decreases in this phase.

A.7. Ablation: Is `S-RHOL` only effective in Initial Phase?

It might seem from Fig. 1 that performance gap between `S-RHOL` and `baseline` decreases in the later stages of pretraining. When the learning rate gets low enough towards the end (inside the lr schedule), the `baseline` starts to train slightly better. We used a linear decay with lr going down to zero following Izsak et al. (2021). Note that the gap doesn't decrease as quickly in the case of a longer learning rate schedule (Fig. 7 in Appendix). The gap is high even at $25K$ steps. This suggests a decreasing learning rate towards the end of pretraining is working better for a baseline in removing the effect of noisy/redundant examples. That said, it still cannot match the case of continuously training with SRHOL. Following experiment justifies this.

We initialise an LM with `S-RHOL` checkpoint at $5K$ steps and continue training for $20K$ more steps using standard pretraining.

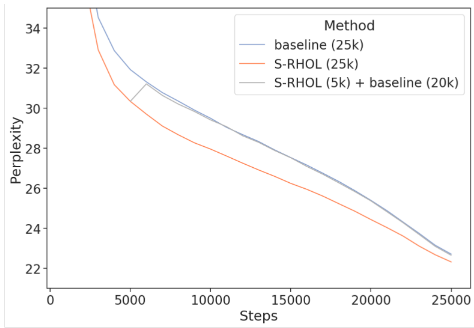

Figure 8: `S-RHOL` $(5K)$ + `baseline` $(20K)$ is only able to achieve `%ΔSteps`of -0.51%

The perplexity of `S-RHOL` $(5K)$ + `baseline` $(20K)$ suddenly increases within a few steps to match the perplexity of the baseline. It closely follows the validation curve of the baseline after that until training completes. This shows that if redundant/noisy examples are not continuously removed, model quickly becomes worse. It is beneficial to constantly keep removing redundant/noisy examples at every stage rather than just at first.