Amuro & Char: Analyzing the Relationship between Pre-Training and Fine-Tuning of Large Language Models

Anonymous ACL submission

Abstract

Large language model development relies on the pre-train-then-align paradigm, in which the model is typically pre-trained on a large text corpus and undergoes a tuning stage to align the model with human preference or downstream tasks. We investigate the relationship between pre-training and supervised fine-tuning by considering multiple tasks as well as different pre-trained model checkpoints. Our results on 18 datasets and two models suggest that i) although the model benefits significantly through supervised fine-tuning, it may forget previously known domain knowledge and tasks that are not seen during fine-tuning; ii) the model exhibits high sensitivity to evaluation prompts after supervised fine-tuning, but this sensitivity can be alleviated through further pre-training; iii) continual pre-training improves the model in a latent way that manifests after fine-tuning; iv) The model can already solve some tasks after pre-training, while fine-tuning most benefits datasets where the model does not show capability during pre-training. [1]

1 Introduction

The rise of large language models (LLMs) as a general-purpose tool for a diverse range of natural language processing tasks has dramatically transformed the field, introducing new paradigms for data collection and model training (Brown et al., 2020, Biderman et al., 2023, Touvron et al., 2023, Jiang et al., 2023, Chowdhery et al., 2023, Groeneveld et al., 2024, Wang et al., 2024, *inter alia*). Numerous models, training methods, datasets, and evaluation methods continue to be developed on an ongoing basis. Nevertheless, a unified paradigm has emerged for training LLMs: pre-train on an enormous corpus of diverse documents, ranging from 250B (Biderman et al., 2023) to 15T (AI@Meta, 2024) tokens, followed by an alignment stage to make the model more useful and performative for various tasks.

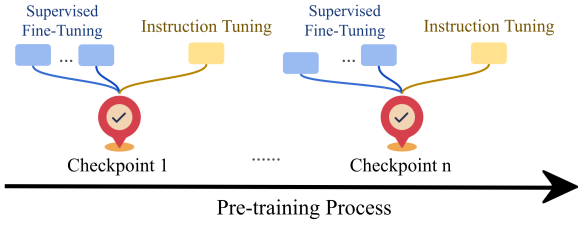


Figure 1: Illustration of the experimental scheme. Intermediate pre-training checkpoints are fine-tuned on different datasets.

Based on this paradigm, work has focused on improving these two stages. Work to improve pre-trained models includes larger training sets (Hoffmann et al., 2022; AI@Meta, 2024; Touvron et al., 2023), different data selection mechanisms (Xia et al., 2024), higher quality data (Zhou et al., 2024), and various model architectures (Su et al., 2024; Touvron et al., 2023). Meanwhile, research on model alignment includes different training objectives (Rafailov et al., 2024; Schulman et al., 2017), new datasets (Narayanan and Aepli, 2024), more efficient training (Hu et al., 2021; Dettmers et al., 2024) and safety tuning (Bianchi et al., 2023). The alignment stage usually involves either supervised fine-tuning for specific tasks or instruction fine-tuning for general-purpose usage. Regardless, fine-tuning (almost always) comes at the end of pre-training and yields remarkable improvements on downstream tasks (Touvron et al., 2023; Groeneveld et al., 2024). Consequently, the benefits of each stage are largely explored independently, with improvements to pretraining being orthogonal to benefits from model alignment.

Rather than exploring these two training regimes independently, we ask: What does the model learn

[1] Code, results, and data to reproduce the experiments are available at https://anonymous.4open.science/r/AmuroCharRelease-DEC5. All the model checkpoints resulting from this work are available at [redacted for anonymity]

and forget during pre-training and fine-tuning? Specifically, **how do pretraining and fine-tuning interact to produce the resulting model?** Does more pre-training hinder better fine-tuning results? Answering these questions requires us to examine how models learn during pre-training and how this affects fine-tuning. Therefore, we begin by fine-tuning two language models under a variety of conditions to determine how fine-tuning affects model behavior. We explore both supervised and instruction fine-tuning, testing the models' memorization and forgetting when learning specific tasks and serving as general-purpose language-AI tools. We then explore the affect of pre-training on these behaviors by fine-tuning **multiple pre-training checkpoints** of a large language model (Figure 1), evaluating each checkpoint and its fine-tuned variant on downstream evaluation sets. We track model abilities during pre-training and compare them to improvements achieved after fine-tuning at the corresponding pre-training step.[2]

Our experiments yield the following insights into LLM training: (1) although supervised fine-tuning can improve performance on in-distribution tasks, it can also cause the model to forget domain knowledge or tasks that it was previously capable of solving (§4); (2) fine-tuned models show high sensitivity to evaluation prompts, but this sensitivity can be alleviated by more pre-training (§4); (3) continued pre-training can improve a model in ways that are only revealed after fine-tuning (§6); (4) tasks for which the model already performs well during pre-training benefit much less from fine-tuning than those where the model does not demonstrate capabilities (§5, §6);

Our findings provide insights into model training and can inform methods for both pre-training and fine-tuning. Furthermore, our work shows the value of analyzing the training dynamics, in addition to analyzing the final checkpoint of an LLM, as an aspect of interpretability, and we encourage model developers to release these checkpoints to aid future studies.

2 Background: Model Training

We use "model alignment" as a general term for techniques that align a model with a desired behavior, often accomplished by fine-tuning models after pretraining. The term is also associated with other definitions (Shen et al., 2024).

We begin with a brief survey of the core components of LLM training: pre-training, fine-tuning, and instruction fine-tuning. The first step of training an LLM is pre-training on a massive text corpus (Achiam et al., 2023; Touvron et al., 2023; Groeneveld et al., 2024). Initial work increased model size to hundreds of billions of parameters (Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2023), along with explorations in model size, training corpus size, and training data characteristics (Radford et al., 2019; Hoffmann et al., 2022; Gururangan et al., 2020). Other work increased the amount of pre-training data (Computer, 2023; Soldaini et al., 2024), with new models now reaching 15 trillion tokens (AI@Meta, 2024).

After the pre-training stage, when a specific task of interest has been identified, supervised fine-tuning can improve a pre-trained model. Task-agnostic tuning became popularized with the advent of T5 models (Raffel et al., 2020), where a pre-trained LLM is tuned using a general text-to-text solution. Instruction fine-tuning is preferred when more general model behaviors are desired. When multiple tasks are given to the model, the model is commonly given a task-specific prefix or an instruction along with the task input, leading to the development of various methods of prefix tuning (Li and Liang, 2021) and instruction tuning (Wei et al., 2021; Mishra et al., 2022; Victor et al., 2022).

Other works explore human preference tuning with or without a reward model (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Rafailov et al., 2024; Song et al., 2024; Xu et al., 2024). In-context learning utilizes a small amount of supervised data to improve model performance without updating the parameters. In this work, we focus specifically on single-task supervised fine-tuning and multi-task instruction tuning.

3 Experimental Setup

In this section, we describe the models and datasets used. The hyperparameter tuning procedure and setup for each fine-tuning setting can be found in Appendix B.

[2]While we believe that we were the first to explore these issues through intermediate model checkpoints, recently released work has also utilized pre-training checkpoints and are highlighted in Section 8.

[3]https://huggingface.co/datasets/pietrolesci/gpt3_nli

Supervised Fine-Tuning			
Task	**Training**	**ID Test**	**OOD Test**
Summary Generation	XSum	XSum, XLSum	CNN
Question Generation	SocialIQa	SocialIQA	SciQ, TweetQA
Natural Language Inference	MNLI	MNLI1, MNLI2	RTE, GPT3NLI[3]
Paraphrase Detection	Paws	Paws	QQP, STS-B
Instruction Tuning			
Dataset	**Description**		
TÜLU-v2	A mixture of instruction datasets.		
ARC	Grade-school multiple-choice QA.		
OpenbookQA	Open book exam QA.		
Hellaswag	Commonsense inference.		
BoolQ	Reading comprehension.		
SciQ	Science exam multiple choice QA.		

Table 1: Dataset information. For Generation tasks, ROUGE-L is used as evaluation metric, and accuracy is used for classification tasks. ID = In-domain, OOD = Out-of-domain.

3.1 Model Choice

We consider two open models of different architectures and scales: Llama3-8B (AI@Meta, 2024) and OLMo-1B (Groeneveld et al., 2024). To minimize potential confounding factors such as multilingual ability and double descent (Belkin et al., 2019; Caballero et al., 2022; Schaeffer et al., 2023), we exclusively select models predominantly pre-trained in English and incorporate significantly more pre-trained tokens than the number of parameters. We do not include models trained in a multi-stage manner to ensure uniformity of the tokens seen by the model during pre-training. Some of our experiments consider intermediate pre-training checkpoints. We select checkpoints uniformly by the number of tokens seen from the pre-training history along with the first and the final checkpoints. Unfortunately, very few large language models release intermediate pre-training checkpoints (summarized in Table 2). Further consideration and reasoning of model selection are included in Appendix A.

3.2 Training Procedure

We fully fine-tune each of the selected model checkpoints using two different procedures to create fine-tuned models: supervised fine-tuning and instruction tuning. The supervised fine-tuning is conducted separately for each model checkpoint and dataset, while the instruction fine-tuning is done once using the instruction dataset. The instruction-tuned model is evaluated on a suite of LLM benchmarks. All experiments are conducted on two Nvidia 80GB A100, with a total cost of approximately 1100 GPU hours. The detailed number of GPU hours consumed for each experiment is included in Appendix E.

Supervised Fine-tuning. We adapt the datasets from Yang et al. (2024) for supervised fine-tuning. For each in-domain dataset, one to two cross-domain evaluation datasets are supplied. OLMo-1B is fully fine-tuned for 3 epochs with a batch size of 8, while Llama3-8B is fine-tuned with a batch size of 16 and 2 training epochs. Both models are trained with learning rates resulting from minimal hyperparameter tuning (Appendix B). Each task is formatted using a default prompt-completion format (Table 5).

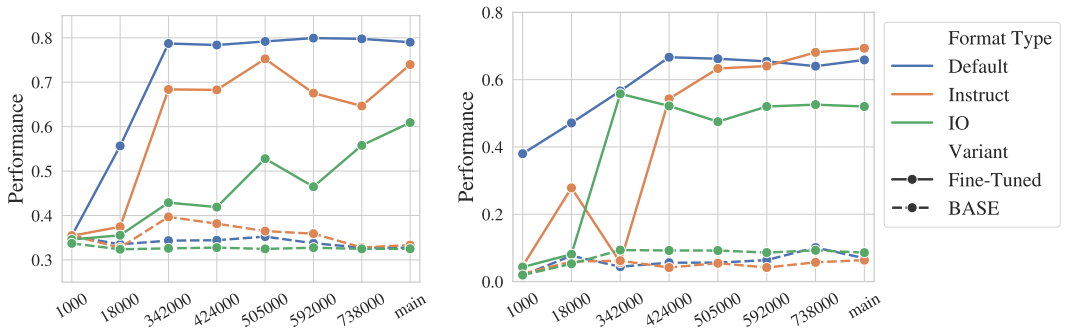
Instruction Fine-Tuning. We instruction-tune the model on TÜLU (Ivison et al., 2023), following the decision of Groeneveld et al., 2024. Each model checkpoint is fully fine-tuned for 5 epochs with a batch size of 8 and a learning rate of 2×10^{-6}.

3.3 Evaluation

For each model, we conduct a few-shot evaluation with a shot size of 4, after examining with shot size in $\{0, 2, 4, 6\}$.

Datasets. The datasets are summarized in Table 1 and data licenses are in Table 7. We evaluate the model with an in-domain test set and one or two out-of-domain test sets for each of the supervised fine-tuning tasks. We conduct experiments on the tasks of *summary generation* (Narayan et al., 2018; Hasan et al., 2021; Hermann et al., 2015), *question generation* (Sap et al., 2019; Xiong et al., 2019; Welbl et al., 2017), *natural language inference* (Williams et al., 2018; Wang et al., 2018; Dagan et al., 2006; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), and *paraphrase detection* (Zhang et al., 2019; Wang et al., 2018; Agirre et al., 2007). We subsample 6,000 training instances for each set to ensure a fair comparison.

In instruction fine-tuning, we base our downstream evaluation settings on Groeneveld et al. (2024), as OLMo is found to have stable performance on these datasets. The instruction-tuned models are evaluated on ARC (both `arc easy` and `arc challenge`) (Clark et al., 2018), OpenbookQA (Mihaylov et al., 2018), Hellaswag (Zellers et al., 2019), BoolQ (Clark et al., 2019),



(a) MNLI matched (b) SocialIQa

Figure 2: Example of model performance with different task formats. The figure of all datasets can be found in Figure 14.

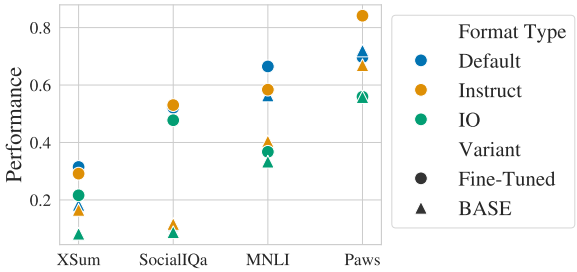


Figure 3: LLAMA3-8B performance with different task format. `Instruct` and `Default` always lead to highest evaluation results.

and SciQ (Welbl et al., 2017).

Metrics. We use accuracy (Pedregosa et al., 2011) for classification tasks and ROUGE-L (Lin, 2004) for generation tasks. The maximum amount of newly generated tokens is set to 5 for classification tasks and 60 for generation tasks. Outputs are generated with greedy decoding. For classification tasks, we experiment with both constrained decoding and logit-based predictions. We find the best performance by selecting the label with the highest logit of its first subtoken (Appendix C).

4 Supervised Fine-Tuning: What does the model learn and forget?

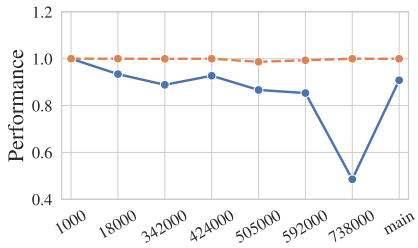
We begin our analysis with the supervised fine-tuning process to understand the downstream results of the training process. Specifically, we explore three dimensions: **task format, task transfer, and domain knowledge**. In each case, we fine-tune both final checkpoints and intermediate pre-training checkpoints to understand the relationship between pre-training and fine-tuning.

4.1 Task Format

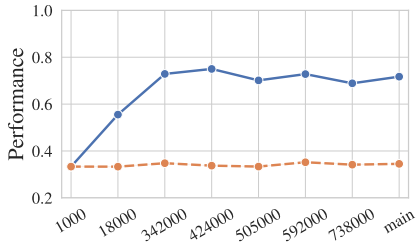
LLMs can be extremely sensitive to prompt perturbation in few-shot settings (Sclar et al., 2023; Leidinger et al., 2023; Salinas and Morstatter, 2024; Wahle et al., 2024). We hypothesize that fine-tuning fits the model to a specific task format, resulting in higher performance when the evaluation set matches this format. To test this hypothesis, we vary the task format to either match the training format, use a different format, or rely on instructions.

We carefully construct three different prompt formats for the following settings. 1) `Default` is the same format used for training, where we expect the model to benefit from learning the task format; 2) `IO` format, by contrast, reflects a common way of performing supervised fine-tuning by incorporating only unprocessed input and output; 3) `Instruct` uses a human-readable instruction template to format the input. Table 5 shows an example of each format. The performance of Llama3-8B with different task formats is shown in Figure 3. Checkpoint performance on OLMo before and after fine-tuning is shown in Figure 2.

Across both models, `IO` format leads to the least favorable performance, as the only task-specific information in this format is included in the evaluation shots. Model reports similar performance when evaluated with the `default` and `instruct` format, aligning with the findings in Hewitt et al. (2024) that the models retain their instruction-following ability after fine-tuning without instructions. However, in the early pre-training steps, aligning the task format with fine-tuning data plays a crucial role (Figure 2), suggesting that the instruction-following ability has not yet been developed. In this view, **fine-tuning teaches the model how to format a response for the task, while further pretraining enhances the instruction-following ability**. In other words, the instruction provides a directed prior for the model to behave in a certain way.



(a) Paws → QQP



(b) MNLI → GPT3NLI

Figure 4: Example of out-of-domain performance for fine-tuned models. The **solid blue** line represents the fine-tuned checkpoint evaluated on an out-of-domain dataset, and the **dashed orange** line represents the base checkpoint where the model is not fine-tuned. Figure 4a shows an example of fine-tuning hurting OOD performance, while Figure 4b shows an example of fine-tuning boosting OOD performance as pre-traininng proceeds.

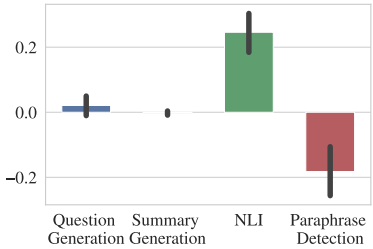


Figure 5: Ratio of out-of-domain performance change for each task, averaged across checkpoints.

4.2 Domain Knowledge

We next explore how the domain-generalization ability is affected by fine-tuning by inspecting whether the model forgets the domain knowledge after fine-tuning on a different domain. An example of OOD model performance is shown in Figure 4, and the mean ratio of change by datasets is presented in Figure 5 and Figure 15.

The models do not benefit equally from the in-domain fine-tuning: Llama shows subtle benefits on question generation tasks, while not benefiting at all on the other tasks (Figure 15). Across OLMo training history (Figure 5), NLI datasets experience a boost when fine-tuning on MNLI, while fine-tuning on Paws is detrimental to other paraphrase detection datasets. This suggests that both forgetting and learning are happening in fine-tuning: the model learns to perform the task with in-domain knowledge, but it may, in turn, forget information more distant from what is learned in fine-tuning. Furthermore, **under the same task, the amount of general-purpose pre-training may not affect the model's reaction to out-of-domain knowledge**. Questions remain, however, about whether domain-specific continual pre-training or continual pretraining on similarly distributed data would bring different conclusions, which requires further study of pre-training dynamics.

4.3 Task Transfer

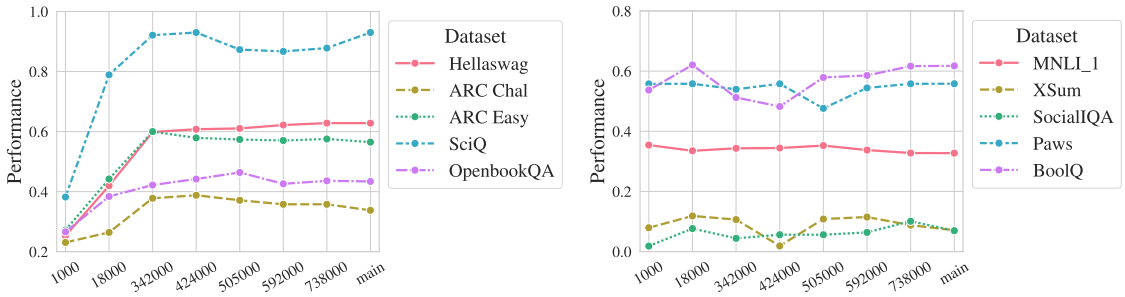
Model forgetting occurs when model training on new tasks improves those tasks at the expense of previously trained tasks (Luo et al., 2023; Mehta et al., 2023; Li and Lee, 2024). To understand whether the model will forget a previously known task solution when fine-tuned on a different one, we evaluate model forgetfulness by examining whether the model does worse on some tasks after fine-tuning for other tasks. Specifically, we divide our tasks into two types: classification and generation.

We notate the training datasets as D_T and the evaluation datasets as D_E. We represent the performance of a pre-trained model (BASE) on checkpoint i as $\text{Perf}^i_{BASE}(d)$ for an evaluation dataset $d \in D_E$, and the performance of the i-th checkpoint fine-tuned on dataset $t \in D_T$ be $\text{Perf}^i_t(d)$. To normalize the effect caused by uneven performance across different datasets, we compute the mean ratio of change (MRC) in performance for each checkpoint as follows.

$$\text{MRC} = \frac{1}{|D_E \setminus \{t\}|} \sum_{\forall d \in D_E, d \neq t} \frac{\text{Perf}^i_t(d) - \text{Perf}^i_{BASE}(d)}{\text{Perf}^i_{BASE}(d)} \quad (1)$$

Models fine-tuned on classification tasks and evaluated on generation tasks decrease on average 61.4% compared to models that are never fine-tuned. In contrast, models fine-tuned on generation tasks can still perform the same as the BASE model on classification tasks, with a 0.3% MRC, which is not statistically significantly different from a 0% change. Our findings on all pre-training checkpoints align with the findings of Yang et al. (2024) on the final checkpoint of LLAMA-7B and our experiments on the final checkpoint of Llama3-8B (Appendix G).

Regardless of the pre-training stage, **a model maintains classification abilities when trained**



(a) Learned during Pre-training.

(b) Never learned during pre-training.

Figure 6: Few-shot performance on different pre-training steps.

for generation but loses generation abilities when trained for classification. This is not surprising given that classification tasks can be seen as a subset of generation, while the reverse is not true. The model follows a simplicity bias (Shah et al., 2020) and thus is more likely to memorize simple classification tasks than generation tasks with an exponentially larger search space. Additionally, since we evaluate the classification tasks based on the output logits and the base model performs randomly on the classification tasks, it is much easier for the models to maintain the same performance as the BASE models. Regardless of the stage of pre-training, fine-tuning can cause a model to lose abilities when the desired fine-tuning behavior does not support those abilities.

Across these three experimental settings, we find that fine-tuning teaches a model how to perform a task without hurting the model's instruction-following ability, but can sacrifice generalization across domains and tasks.

5 How does the model change across pre-training?

Section 4.1 reveals that the effect brought by fine-tuning could be different depending on the amount of pre-training, but how exactly does pre-training affect downstream fine-tuning results? We begin by considering how additional pre-training changes the **BASE** model. Typically, researchers track the value of the training or held-out loss during training. However, performance improvements on downstream tasks do not always follow the same trend with the loss curves (Groeneveld et al., 2024).

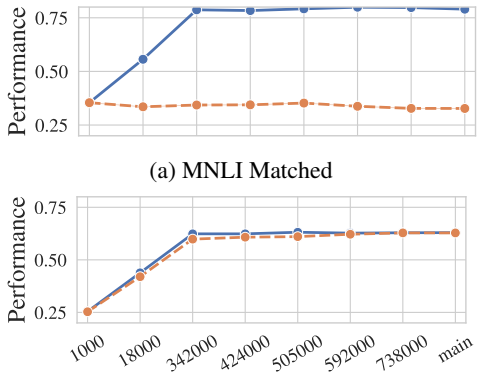
Instead, we evaluate the pre-trained checkpoints with few-shot examples, as models without alignment tend to do poorly in a zero-shot context. Four shots are randomly sampled from the datasets, which are selected based on the highest performance shot amount reported in Yang et al. (2024). The model's performance at each pre-training step is reported in Figure 6.

Broadly speaking, our results suggest that all datasets fall into one of two groups. For the first group of datasets (Figure 6a), although the model shows clear improvement during the early stages of pre-training, performance levels off fairly early on and remains consistent. The dramatic improvements in the early stages of pre-training may result from larger steps in early optimization. We find improvements stop increasing past step 342,000. The second group (Figure 6a) shows tasks that are never learned during pre-training. Performance remains constant throughout the whole pre-training process even when we vary shot sizes. These datasets include MNLI, XSum, and BoolQ. A natural hypothesis for this finding is potential data contamination in the pre-training data. However, the evaluation datasets are selected based on the popularity of the task and the content of pre-training data. All datasets that experience improvement do not exist in the model's pre-training data (Soldaini et al., 2024), while the more likely leaked datasets (MNLI, XSUM) never gain an improvement during the pre-training process.

Overall, these results reveal an interesting dichotomy. **Some tasks can be learned during pre-training, while others cannot**. Next, we explore what exactly the model is learning regarding this second group of datasets during pre-training by exploring the fine-tuned models.

6 Does more pre-training yield better fine-tuning results?

Groeneveld et al. (2024) compares OLMo's performance on several tasks before and after fine-tuning the final checkpoint and finds that fine-tuning enables the model to do well on tasks for which the



(a) MNLI Matched

(b) Hellaswag

Figure 7: Example of few-shot performance on different pre-training steps of the models that benefited (7a) and did not benefit from fine-tuning (7b). The **solid blue** line represents the fine-tuned checkpoint, and the **dashed orange** line represents the base checkpoint. The results of all datasets can be found in Figure 10 and Figure 9.

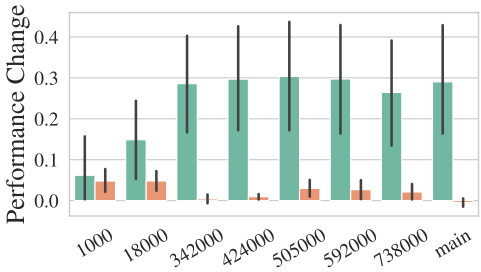


Figure 8: Amount of performance increase brought by fine-tuning between tasks that model can solve in pre-training (**mandarin orange**) and tasks that the model could not solve until fine-tuning (**sage green**). The exact number of mean increase is shown in Appendix J.

unaligned model does poorly. We observe (§5) that while some datasets improved during pre-training, there is a group of datasets for which a pre-trained model does poorly. Does the model learn useful information for these tasks but cannot express it without fine-tuning? In this section, we further explore this dataset dichotomy by examining fine-tuned checkpoints for each of the datasets.

Our results appear in Figure 7 and Figure 8. First, we consider those datasets where the pre-trained models do well (Figure 6a). These datasets do not improve with fine-tuning, suggesting that whatever is learned during fine-tuning, which we discuss below, the model already gains the knowledge during pre-training. This effect is observed at all checkpoints; fine-tuning simply does not help.

However, a different story is observed for datasets that are not learned during pre-training. For these, fine-tuning yields significant improvements at every model checkpoint, with Figure 8 showing the magnitude of improvement on these datasets compared to no improvement to the datasets already learned during pre-training. Moreover, earlier checkpoints obtain more substantial gains from fine-tuning than later checkpoints. The benefit of fine-tuning continues to increase until a certain threshold in pre-training steps is reached (approximately 424,000). Figure 7 shows representative plots comparing the performance of a pre-trained versus fine-tuned model at different checkpoints for two datasets (full list in Appendix F). For Hellaswag (learned during pre-training), fine-tuning does not benefit the model, even during early checkpoints when the model performs poorly on the task. Nevertheless, for MNLI (not learned during pre-training), fine-tuning dramatically improves the model. Interestingly, later checkpoints achieve better results after fine-tuning, even when the performance of the pre-trained model is unchanged. This suggests that **the model is, in fact, improving during pre-training, but it cannot express that improvement without fine-tuning**.

Our findings suggest that early stopping in pre-training will not be detrimental to downstream fine-tuning performance. When the budget is limited, the benefits of fine-tuning an LLM could exceed the benefits of continued pretraining, which sheds light on the potential of a cost-effective training paradigm with less pre-training. However, directly identifying such stopping criteria without fine-tuning intermediate checkpoints is challenging. We only empirically observed that the point where more pre-training lead to diminishing return on downstream fine-tuning results approximately align with the turning point of few-shot performance in Section 5. Without such a hypothesis, the improvement trend is invisible before fine-tuning the checkpoints. Overall, when resource-intensive pre-trained LLMs are not available, fine-tuning models on checkpoints with less pre-training may be a reasonable practical choice for obtaining a high-quality model.

7 Discussion

Our study fine-tunes model pre-training checkpoints to understand the dynamics of pre-training and fine-tuning on model performance.

Fine-tuning teaches additional task format but leads to forgetting unused abilities. Our results show that fine-tuning guides the model to under-

stand the format and complete a given task. As this information diminishes, the model's overall ability improves. Additionally, more pre-training will lead to a model that reacts better to instruction-style prompts, and the ability to interpret such instruction will not be lost when the model is fine-tuned in a different format. However, fine-tuning comes at the expense of other model abilities, such as the capability of solving tasks or domains that are unrelated or weakly related to the fine-tuning task. This insight can be helpful in our understanding of the multitask abilities of LLMs, where certain tasks can introduce conflicts during multi-task training (Mueller et al., 2022).

Some datasets can be learned without fine-tuning. We discover a dichotomy between datasets. Some are learned during model pre-training, while others show no improvements during pre-training. Furthermore, the datasets learned during pre-training do not benefit from fine-tuning. This observation, combined with our study about what is learned during fine-tuning (§4) suggests that some tasks are presented in a manner that aligns with what the model sees during pre-training, and thus fine-tuning provides no additional information. It may be possible to modify tasks to better align with pre-training and thus make them learnable.

Pre-training can improve models in unseen ways. Some datasets are not learned during pre-training but benefit significantly from fine-tuning (§5). However, these datasets still benefit from additional pre-training, even though those benefits are not revealed without fine-tuning (§6). The model learns important information to solve the task, even though it cannot express that information without fine-tuning. We empirically observe that the point where more pre-training lead to diminishing return on downstream fine-tuning results approximately align with the turning point of few-shot performance in Section 5. Future work may identify ways to verify the turning point and detect these improvements during pre-training, which can better guide pre-training choices to produce models that perform better post-fine-tuning. Perhaps there is a way in which information about these tasks can be included in pre-training, allowing the model to better utilize the massive amount of pre-training data. For example, early stopping during pre-training could lead to better utilization of limited training resources if we know when to stop.

8 Related Work

Recent studies identify *phase transition* of model training (Olsson et al., 2022; Wei et al., 2022), where new capabilities or behaviors suddenly emerge when certain thresholds of model complexity are reached. The aspects of complexity often include model size, amount of training by FLOPs or tokens, and model architecture. Several prior works studied the training dynamics of language models by analyzing the internals of train-from-scratch models (Tirumala et al., 2022; Chen et al., 2023; Tian et al., 2023; Chen et al., 2024; Chang et al., 2024). The results of these works suggest that the behaviors that are often overlooked after training could be valuable signals for model analysis. In addition to train-from-scratch models, Ren and Sutherland (2024) studied the fine-tuning dynamics of language models. This work focuses on *the effect of pre-training dynamics on downstream fine-tuning results* by fine-tuning intermediate pre-training checkpoints on various tasks. Due to the scarcity of publically accessible intermediate pre-training checkpoints, the effect of fine-tuning at different pre-training stages is largely unexplored. Concurrent work (Snell et al., 2024) also fine-tunes intermediate pre-training checkpoints and finds that supervised fine-tuning results can be used as a signal to predict when emergence occurs, while our findings point out a dichotomy of model behavior on different datasets, with the potential for data-efficient and budget-friendly training by understanding the stages of model training.

9 Conclusion

We explore the relationship between fine-tuning and pre-training LLMs through fine-tuning multiple pre-training checkpoints of large language models. Our results on 18 datasets and two models provide insights into LLM training. We identify the aspects that LLM learns and forgets during supervised fine-tuning; By analyzing pre-training history, we find that pre-training improves the model in a latent way that is only observable after fine-tuning. The model may excel at some tasks without fine-tuning. However, the model can rapidly learn datasets that it does not demonstrate capabilities during pre-training with a small amount of supervision. Overall, our study highlights the value of analyzing language model training dynamics. We encourage model developers to release pre-training checkpoints to facilitate research on LLM training.

Limitations

While our insights suggest directions for future work, we note important limitations inherent in our experiments. We discuss the weaknesses and limitations in the following section.

Computing Resource. Due to computational constraints, we can only conduct checkpointing experiments on a 1B model. We supply the final checkpoint of an 8B model to verify the findings that are shared across checkpoints. The amount of GPU hours spent for each experiment in this study is listed in Table 4.

Model Size and Variant. For the analysis with intermediate checkpoints, this study considered a single, relatively small LLM, which may, therefore, conceal the emergent capability brought by larger models (Wei et al., 2022). To combat this, we include the final checkpoint of an 8B model from a different model family. Future work needs to confront these issues on larger models and more datasets.

Availbility of Pre-training Checkpoints. Although Choshen et al. (2024) points out that the behavior of a model can often be predicted with a model with the same architecture but a different family. This study would benefit significantly from including a broader spectrum of models, but the public pre-training checkpoint releases are limited. We list open-source LLMs with intermediate checkpoint release in Appendix A. After a series of preliminary experiments, we select available models' best-performing and robust families.

Analysis Protocol. Wu et al. (2023) show that the evaluation result may be affected by samples that have been memorized by the model during training instead of revealing the reasoning capability. The only analysis protocol used in this work is the downstream performance of a trained model. More investigation should be done into model internals during pre-training dynamics and how they relate to the effects of fine-tuning.

Training Paradigm. Although multiple tuning strategies exist, to create a fair comparison environment where checkpoints receive the same amount of training, models are fine-tuned with a fixed amount of epochs in this work. On different pre-training stages, the model may converge at a different speed. Further study can be done to study the effect of pre-training on different fine-tuning methods or fine-tuning dynamics in different pre-training stages. We only explored the scenario of full-parameter fine-tuning. Whether parameter-efficient fine-tuning or human preference tuning will lead to a different conclusion also remains an open question.

Randomness. In this study, we only assess uncertainty with Bootstrap during evaluation. However, uncertainty may emerge during training, which poses optimizer initialization and data ordering, the study of which requires an extensive amount of computing resources.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Eneko Agirre, Llu'is M'arquez, and Richard Wicentowski, editors. 2007. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Association for Computational Linguistics, Prague, Czech Republic.

Ai2. 2024. Olmo2 blog.

AI@Meta. 2024. Llama 3 model card.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Khuyagbaatar Batsuren, Ekaterina Vylomova, Verna Dankers, Tsetsuukhei Delgerbaatar, Omri Uzan, Yuval Pinter, and Gábor Bella. 2024. Evaluating subword tokenization: Alien subword composition and oov generalization challenge. *arXiv preprint arXiv:2404.13292*.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. 2023. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *arXiv preprint arXiv:2309.07875*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. 2022. Broken neural scaling laws. *arXiv preprint arXiv:2210.14891*.

Tyler A Chang, Zhuowen Tu, and Benjamin K Bergen. 2024. Characterizing learning curves during language model pre-training: Learning, forgetting, and stability. *Transactions of the Association for Computational Linguistics*, 12:1346–1362.

Angelica Chen, Ravid Schwartz-Ziv, Kyunghyun Cho, Matthew L Leavitt, and Naomi Saphra. 2023. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms. *arXiv preprint arXiv:2309.07311*.

Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. 2024. Training dynamics of multi-head softmax attention for in-context learning: Emergence, convergence, and optimality. *arXiv preprint arXiv:2402.19442*.

Leshem Choshen, Yang Zhang, and Jacob Andreas. 2024. A hitchhiker's guide to scaling law estimation. *arXiv preprint arXiv:2410.11840*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

John Hewitt, Nelson F Liu, Percy Liang, and Christopher D Manning. 2024. Instruction following without instruction tuning. *arXiv preprint arXiv:2409.14254*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks,

Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, et al. 2023. A taxonomy and review of generalization research in nlp. *Nature Machine Intelligence*, 5(10):1161–1174.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Alina Leidinger, Robert Van Rooij, and Ekaterina Shutova. 2023. The language of prompting: What linguistic properties make a prompt successful? *arXiv preprint arXiv:2311.01967*.

Chen-An Li and Hung-Yi Lee. 2024. Examining forgetting in continual pre-training of aligned large language models. *arXiv preprint arXiv:2401.03129*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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.

LLM360. 2024. K2 blog release.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. 2023. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

David Mueller, Nicholas Andrews, and Mark Dredze. 2022. Do text-to-text multi-task learners suffer from task conflict? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2843–2858, Abu Dhabi, United Arab Emirates. 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.

Manu Narayanan and Noëmi Aepli. 2024. A Tulu resource for machine translation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1756–1767, Torino, Italia. ELRA and ICCL.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

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. *Journal of machine learning research*, 21(140):1–67.

Yi Ren and Danica J Sutherland. 2024. Learning dynamics of llm finetuning. *arXiv preprint arXiv:2407.10490*.

Abel Salinas and Fred Morstatter. 2024. The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. *arXiv preprint arXiv:2401.03729*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo. 2023. Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle. *arXiv preprint arXiv:2303.14151*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585.

Hua Shen, Tiffany Knearem, Reshmi Ghosh, Kenan Alkiek, Kundan Krishna, Yachuan Liu, Ziqiao Ma, Savvas Petridis, Yi-Hao Peng, Li Qiwei, Sushrita Rakshit, Chenglei Si, Yutong Xie, Jeffrey P. Bigham, Frank Bentley, Joyce Chai, Zachary Lipton, Qiaozhu Mei, Rada Mihalcea, Michael Terry, Diyi Yang, Meredith Ringel Morris, Paul Resnick, and David Jurgens. 2024. Towards bidirectional human-ai alignment: A systematic review for clarifications, framework, and future directions. *arXiv preprint arXiv:2406.09264*.

Aaditya K Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *arXiv preprint arXiv:2402.14903*.

Charlie Snell, Eric Wallace, Dan Klein, and Sergey Levine. 2024. Predicting emergent capabilities by finetuning. *arXiv preprint arXiv:2411.16035*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Kaiser Sun, Peng Qi, Yuhao Zhang, Lan Liu, William Wang, and Zhiheng Huang. 2023. Tokenization consistency matters for generative models on extractive NLP tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13300–13310, Singapore. Association for Computational Linguistics.

Tianhua Tao, Junbo Li, Bowen Tan, Hongyi Wang, William Marshall, Bhargav M Kanakiya, Joel Hestness, Natalia Vassilieva, Zhiqiang Shen, Eric P Xing, et al. 2024. Crystal: Illuminating llm abilities on language and code. *arXiv preprint arXiv:2411.04156*.

Yuandong Tian, Yiping Wang, Beidi Chen, and Simon S Du. 2023. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. *Advances in Neural Information Processing Systems*, 36:71911–71947.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Sanh Victor, Webson Albert, Raffel Colin, Bach Stephen, Sutawika Lintang, Alyafeai Zaid, Chaffin Antoine, Stiegler Arnaud, Raja Arun, Dey Manan, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Jan Philip Wahle, Terry Ruas, Yang Xu, and Bela Gipp. 2024. Paraphrase types for generation and detection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024. Helpsteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106.

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

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*.

Wenhan Xiong, Jiawei Wu, Hong Wang, Vivek Kulkarni, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. TWEETQA: A social media focused question answering dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5020–5031, Florence, Italy. Association for Computational Linguistics.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng Ann Heng, and Wai Lam. 2024. Unveiling the generalization power of fine-tuned large language models. *arXiv preprint arXiv:2403.09162*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Model and Data Selection

Only a small subset of large language models publicly release their intermediate training checkpoints. We list these models in Table 2 and would like to call for model developers to release intermediate checkpoints in the future to aid the research of training dynamics. To reduce the confounding factor of language and stages of training, we select the models that are dominantly trained in English and followed a single-staged training strategy. Only the models pre-trained with significantly more tokens than the model parameters are considered to avoid the occurrence of double descent (Belkin et al., 2019; Schaeffer et al., 2023) in the middle of pre-trianing, which could lead to a broken scaling law (Caballero et al., 2022) that complicates the analysis. Additionally, we restrict our selection to models pre-trained on over one trillion tokens, thereby ensuring a sufficiently extended training trajectory is represented. We conduct initial experiments with OLMo and RedPajama-INCITE. We observe that the RedPajama-INCITE shows subtle improvement following instruction-tuning or fine-tuning, and its 7B variant shows lower performance compared to OLMo 1B. Therefore, we select OLMo 1.0 1B as the backbone for analysis.

During this study, several recent initiatives released the intermediate checkpoints. We also list these works in Table 2.

B Hyperparameter Tuning

For both supervised fine-tuning and instruction tuning, we pre-set the effective batch size to 8, and tune the learning rate within $\{2 \times 10^{-5}, 2 \times 10^{-6}, 2 \times 10^{-7}\}$. OLMo-1B is fine-tuned for 3 epochs on the supervised fine-tuning tasks and 5 epochs on Tulu for instruction tuning. Llama3-8B is fine-tuned for 2 epochs with a learning rate of 5×10^{-6}, with learning rate selected from $\{5 \times 10^{-5}, 5 \times 10^{-6}, 5 \times 10^{-7}\}$. In both settings, we adopt an AdamW optimizer with a linear learning rate scheduler. The optimizer is warmed up for the first 3% of the training time.

C Prediction Generation Method

For classification tasks, we examine three different prediction generation methods: Free Generation (`Free`), Constrained Generation (`Constrained`), and Token Probability (`TokenProb`), the results are shown in Table 3. In `Constrained`, we force the output to include at least one label in the acceptable label set. In `TokenProb`, we compare the logits of acceptable labels and select the label with the highest score as the final output. This ablation study is conducted only on the BASE and fine-tuned versions of the final checkpoint of the pre-trained model. We find that, although prediction generation methods have less effect on the evaluation result of a fine-tuned model, BASE variants suffer much more from not knowing the desired output. Therefore, we proceed with all the classification experiments with `TokenProb`.

C.1 Label and Tokenizations

Depending on the tokenizer variant, the label text may be tokenized differently, leading to evaluation unreliability. For example, in paraphrase detection, the model could assign probability on both "`yes`" and " `yes`" (the same label with a prefix space). This behavior is reported and explored in various related work (Sun et al., 2023; Batsuren et al., 2024; Singh and Strouse, 2024). In this study, we leniently regard all individual tokens that contain the whole label or part of the label along with some special characters that do not affect the semantics as an acceptable target label.

D Task Format

We adopt the task format from (Yang et al., 2024), with an additional task format of input-output. How each dataset is formated can be found in Table 5.

E GPU Hours per-Experiment

We show a table of GPU hours spent for each experiment in Table 4. The total number of GPU hours spent on this project is approximately 1067 A100 hours. We lose track of the GPU hours spent on preliminary experiments, so a lower-bound estimation is reported.

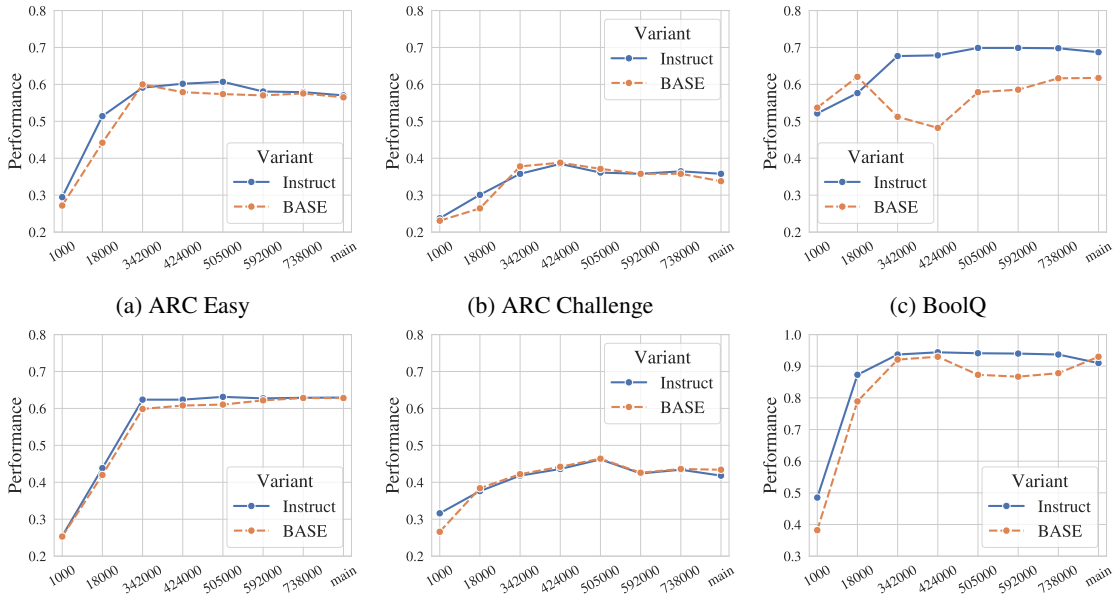
F Per-dataset Figures

We show the model performance on each dataset after supervised fine-tuning and instruction tuning correspondingly in Figure 10 and Figure 9. The

	Pythia	OpenLLAMA	K2 (LLM360)	Crystal (LLM360)	Baichuan2
Citation	Biderman et al., 2023	Geng and Liu, 2023	LLM360, 2024	Tao et al., 2024	Yang et al., 2023
Size (Param)	70M, 160M, 410M, 1B, 1.4B, 2.8B, 6.9B, 12B	3B, 7B	65B	7B	7B, 13B
Languages	English	English	English	English	English & Chinese
Pre-trained Tokens	300B	1T	1.4T	1300B	2.6T
Note	-	-	Multi-phase pre-training	Multi-phase pre-training	-

	OLMO-2	OLMO	TinyLLaMA	RedPajama-INCITE	Bloom
Citation	Ai2, 2024	Groeneveld et al., 2024	Zhang et al., 2024	Computer, 2023	Le Scao et al., 2023
Size (Param)	4T, 5T	1B, 7B	1B	7B	176B
Languages	English	English	English	English	Multilingual
Pre-trained Tokens	7B, 13B	3T, 2.5T	3T	1.2T	366B
Note	Multi-phase pre-training	-	BOS Token leads to training history inconsistency.	Poor fine-tunablity	-

Table 2: Large language models with public release of intermediate pre-training checkpoints. All models are under Apache 2.0 license.



(a) ARC Easy (b) ARC Challenge (c) BoolQ

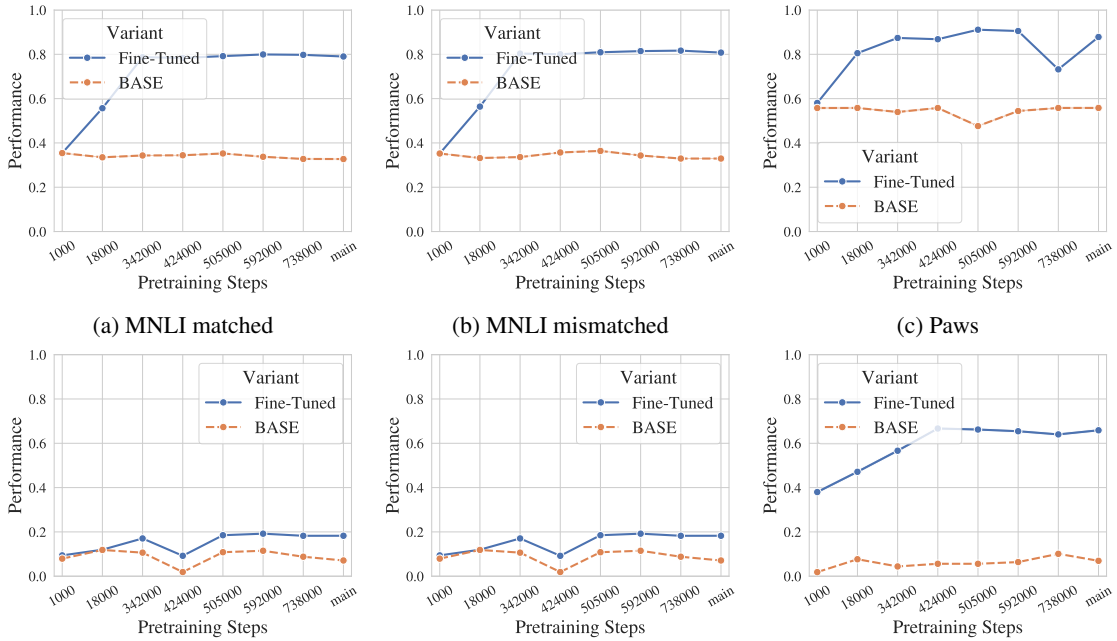
(d) Hellaswag (e) Openbook QA (f) SciQ

Figure 9: Model performance after instruction tuning on each pre-training step.

datasets that already show improvement during pre-training do not benefit from fine-tuning, while performance improves drastically on the datasets that the model has never learned during pre-training.

Out-of-domain Generalization The out-of-domain performance for each dataset with respect to pre-training steps is shown in Figure 11. Overall, the model generalizes well after fine-tuning on NLI tasks, while its performance deteriorates when evaluated on out-of-domain paraphrase detection tasks.

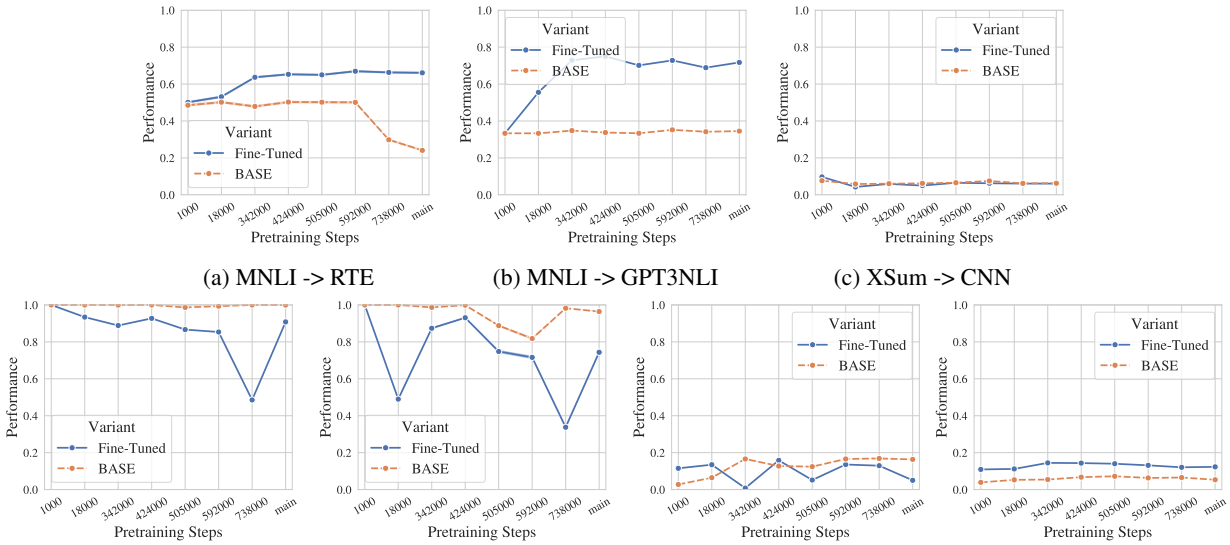
Cross-task Generalization The cross-task per-



(a) MNLI matched (b) MNLI mismatched (c) Paws

(d) XSum (e) XLSum (f) SocialIQA

Figure 10: Model performance after supervised fine-tuning on each pre-training step.



(a) MNLI -> RTE (b) MNLI -> GPT3NLI (c) XSum -> CNN

(d) Paws -> QQP (e) Paws -> STS-B (f) SocialIQA -> SciQ (g) SocialIQA -> TweetQA

Figure 11: Out-of-domain performance after supervised fine-tuning on each pre-training step.

formance for each dataset with respect to pre-training steps is shown in Figure 12 and Figure 13.

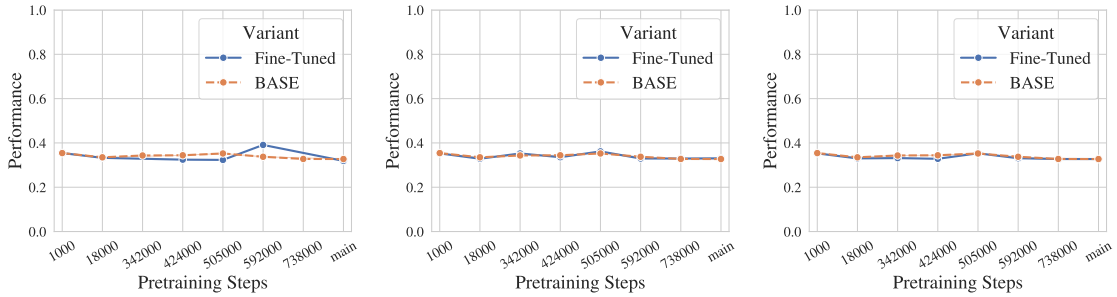
Task-Format The performance of models on evaluation sets formatted with different prompt formatting methods is shown in Figure 14.

G Llama3-8B Results

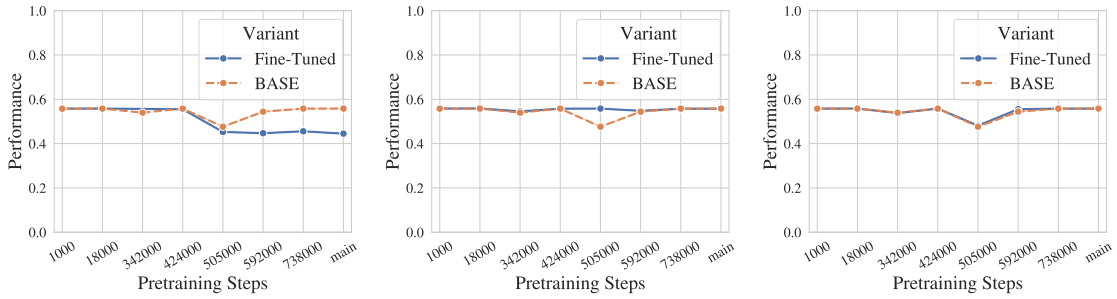
To provide more evidence of the findings on a different model architecture and size, we conduct some experiments on the final checkpoint of Llama3-8B.

Task Transfer Similar to our findings with OLMo, Llama3-8B fine-tuned on classification tasks and evaluated on generation tasks decreases on average 61.0% compared to models that are never fine-tuned. In contrast, models fine-tuned on generation tasks perform similarly to the BASE model on classification tasks, with a 10.6% MRC.

Domain Knowledge The ratio of out-of-domain performance change by task is shown in Figure 15. Overall, we observe that Llama and OLMo experi-

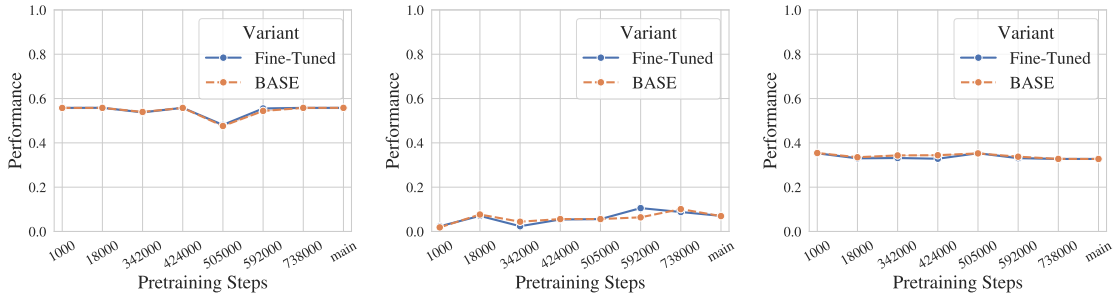


(a) Paws -> MNLI (b) SocialIQA -> MNLI (c) XSum -> MNLI

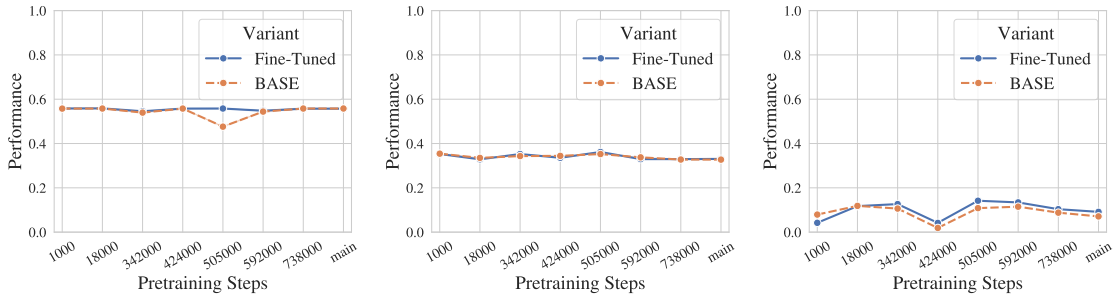


(d) MNLI -> Paws (e) SocialIQA -> Paws (f) XSum -> Paws

Figure 12: Cross-task performance after supervised fine-tuning on each pre-training step. The model is fine-tuned on a classification task and evaluated on a generation task or a classification task with a different label set.



(a) Paws -> XSum (b) SocialIQA -> XSum (c) MNLI -> XSum



(d) Paws -> SocialIQA (e) MNLI -> SocialIQA (f) XSum -> SocialIQA

Figure 13: Cross-task performance after supervised fine-tuning on each pre-training step. The model is fine-tuned on a generation task and evaluated on a classification task.

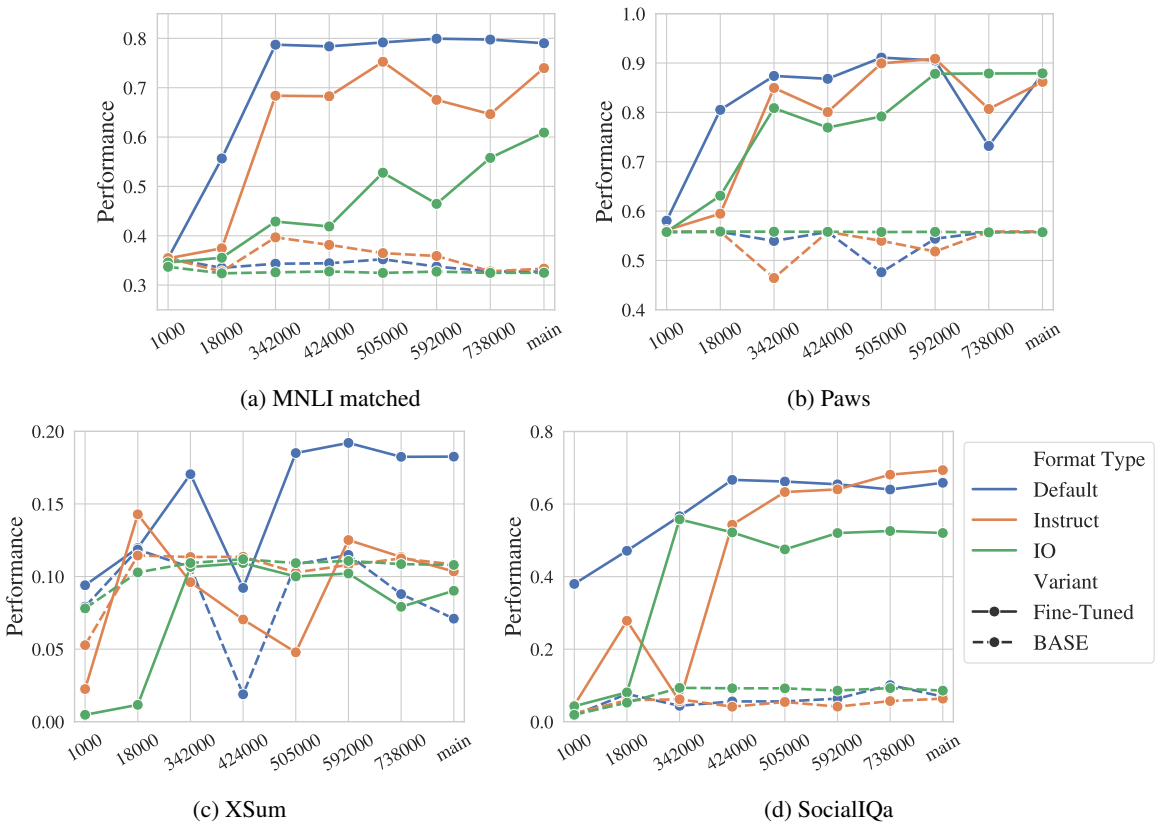
ence benefits with different tasks after fine-tuning, but both model shows an inconsistent change across tasks.

H License of Artifacts

We include the license of artifacts used in this paper in Table 7

I Full Performance Table

Due to the availability of space and the amount of fine-tuned checkpoints, we omit displaying all exact metric values in the paper. The performance of each fine-tuned variant on each dataset can be found in the `csv` file under directory `results` in



(a) MNLI matched

(b) Paws

(c) XSum

(d) SocialIQa

Figure 14: Model performance with different task formats.

Dataset	Model	Free	Constrained	TokenProb
MNLI	Fine-tuned	0.786	0.791	0.792
	BASE	0.0	0.0	0.327
RTE	Fine-tuned	0.658	0.662	0.66
	BASE	0.0	0.0	0.241
Paws	Fine-tuned	0.871	0.878	0.878
	BASE	0.0	0.0	0.558
STS-B	Fine-tuned	0.775	0.741	0.744
	BASE	0.0	0.0	0.964

Table 3: Performance of final checkpoint with different prediction generation method.

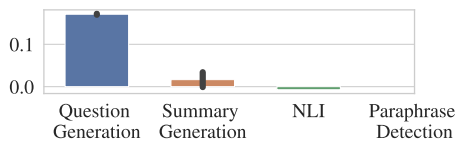


Figure 15: Ratio of out-of-domain performance change for each task on the final checkpoint of LLAMA3-8B.

the code base.

J Performance Difference Numbers

The average performance change before and after fine-tuning for each checkpoint is shown in Table 6. The data in this table is used to create Figure 8.

K Generalization Taxonomy

Following the generalization taxonomy in Hupkes et al. (2023), the evaluation card is included in Table 8.

Prelinminary Experiments				
Description			**GPU Hours**	
Instruction Tuning on LIMA, TULU, and NaturalInstructions			∼300	
Model Performance Verification, Dataset Selection			120	
Instruction Tuning				
Instruction Tuning			360	
Evaluation			10	
Total			**370**	
Fine-Tuning				
	XSum	**SocialIQa**	**MNLI**	**Paws**
Training	12	6	4.6	5.3
Evaluation	8	5.3	3	2
OOD Evaluation	96	32	11	25.6
CrossTask Evauation	5.2	6.5	7.7	8.15
Task Format Evaluation	16	12.8	6	4
Total	137.2 + 62.6 + 32.3 + 45 = 277.1			

Table 4: GPU hours for each experiment. The total amount of GPU hours spent in this project is approximately 1067 A100 hours.

Task	Default Prompt	Instruction Prompt	IO Prompt	Expected Output
Summary Generation	### Input: {document} ### Summary:	Please read the following text: {document} Provide a summary:	{document}	{summary}
Question Generation	### Input: {context} ### Answer: {answer} ### Question:	Given the context: {context} And the answer: {answer} Generate a suitable question:	{context} {answer}	{question}
Natural Language Inference	### Input_1: {premise} ### Input_2: {hypothesis} ### Inference:	Consider the following texts: Text 1: {premise} Text 2: {hypothesis} The relation is	{premise} {hypothesis}	{label}
Paraphrase Detection	### Input_1: {sentence1} ### Input_2: {sentence2} ### Paraphrase Classification:	Let's compare the two sentences: Sentence_1: {sentence1} Sentence_2: {sentence2} Are they paraphrasing?:	{sentence1} {sentence2}	{label}

Table 5: Formatting of the prompts

Checkpoint	Learned in Pre-train	Learned in Fine-Tune
1000	0.048	0.062
18000	0.048	0.149
342000	0.004	0.286
424000	0.01	0.297
505000	0.03	0.304
592000	0.027	0.297
738000	0.021	0.264
main	-0.005	0.290

Table 6: Average performance change before and after fine-tuning for each checkpoint (Perf(Fine-tuned) - Perf(BASE)). The group that is never learned during pre-training is picked up by the model during fine-tuning.

Name	License	Name	License
OLMo-1b	Apache 2.0	SocialIQa	CC-BY
TULU	ODC-BY	CNN/DailyMail	Apache 2.0
ARC	CC BY-SA	TweetQA	CC BY-SA-4.0
OpenbookQA	Apache 2.0	MNLI	CC-BY-3.0
Hellaswag	MIT	GPT3NLI	MIT
BoolQ	Apache 2.0	RTE	N/A
SciQ	CC-BY-NC-3.0	Paws	Free
XSum	MIT	QQP	Non-Commercial
XLSum	CC-BY-NC-SA 4.0	STS-B	Other

Table 7: License of artifacts used in this paper.

Motivation					
Practical	*Cognitive*	*Intrinsic*			*Fairness*
□ △					
Generalisation type					
Compositional	*Structural*	*Cross Task*	*Cross Language*	*Cross Domain*	*Robustness*
		△		□	
Shift type					
Covariate	*Label*	*Full*			*Assumed*
□	△				
Shift source					
Naturally occuring	*Partitioned natural*	*Generated shift*			*Fully generated*
□ △					
Shift locus					
Train–test	*Finetune train–test*	*Pretrain–train*			*Pretrain–test*
	□ △				

Table 8: Generalization experiment summary following taxonomy in Hupkes et al. (2023).