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

Anonymous ACL submission

Abstract

The development of large language models leads to the formation of a 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. In this work, we investigate the relationship between pre-training and fine-tuning by fine-tuning multiple intermediate pre-trained model checkpoints, we find that i) continual pre-training improves the model in a latent way that unveils after fine-tuning; ii) with extra fine-tuning, the datasets that the model does not demonstrate capability gain much more than those that the model performs well during the pre-training stage; iii) although model benefits significantly through supervised fine-tuning, it may forget previously known domain knowledge and the tasks that are not seen during fine-tuning; iv) the supervised fine-tuned model resembles high sensitivity to few-shot evaluation prompts, but this sensitivity can be alleviated by more 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.

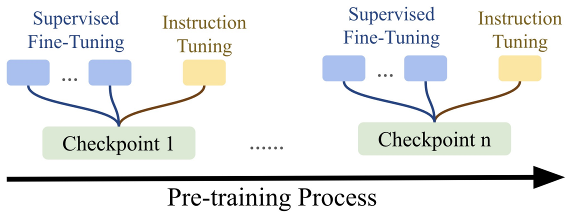


Figure 1: Illustration of the experimental scheme.

Based on this paradigm, work has focused on improving each of these two stages. For better pre-trained models, exploration is done on 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), safety tuning (Bianchi et al., 2023), among other approaches. 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). However, the benefits of each stage are largely explored independently, with improvements to pretraining being orthogonal to benefits from model alignment. Fine-tuning starts with the final pretraining model checkpoint.

Rather than explore these two training regimes independently, we question: **how do model pre-training and fine-tuning interact to affect the resulting abilities of the model?** Does more pre-

[1]Code, results, and data to reproduce the experiments are available at https://anonymous.4open.science/r/AmuroCharRelease-DEC5

training hinder better fine-tuning results? What does the model learn during pre-training, and what does it forget during fine-tuning? To answer these questions, we fine-tune **multiple pre-training checkpoints** of a large language model (Figure 1), evaluating each checkpoint and its fine-tuned version on downstream evaluation sets. We track model abilities during pre-training and compare them to improvements achievable after fine-tuning at each pre-training stage. We explore both supervised fine-tuning and instruction fine-tuning, testing the models' memorization and forgetting when learning specific tasks and serving as general-purpose language-AI tools. To the best of our knowledge, we are the first to explore fine-tuning intermediate model checkpoints.

Our experiments yield novel insights into LLM training. We find that (1) continued pre-training can improve a model in ways that are only revealed after fine-tuning (§5); (2) tasks for which the model performs well during pre-training benefit much less from fine-tuning than datasets where the model does not demonstrate capabilities (§4, §5); (3) although supervised fine-tuning can improve performance on in-distribution tasks, it can also cause the model to forget tasks that it was previously able to solve or domain knowledge (§6); (4) fine-tuned models show high sensitivity to few-shot evaluation prompts, but this sensitivity can be alleviated by more pre-training (§6). Our findings provide useful insights into model training and can inform methods for both pre-training and fine-tuning. Furthermore, our work shows the value of testing multiple model checkpoints, and we encourage model developers to release these checkpoints to aid future studies.

2 Background: Model Training

We begin with a brief survey of the core components of LLM training: pre-training, fine-tuning, and instruction fine-tuning. We also discuss the related topic of in-context learning as well as different efficient fine-tuning strategies.

In this work, "model alignment" is referred to as a general term that refers to aligning the model with a desired behavior, which can be accomplished by fine-tuning models after pretraining. The term is associated with other definitions (Shen et al., 2024). We also note several related studies that explore training dynamics to understand model behavior (Tirumala et al., 2022; Chen et al., 2023; Tian et al., 2023). With this in mind, we conduct an empirical study on how the amount of pre-training affects the effectiveness of fine-tuning.

Pre-training The first step of training a LLM is pre-training on a massive text corpus (Achiam et al., 2023; Touvron et al., 2023; Groeneveld et al., 2024). For decoder-only models in the GPT family, the subject of our paper, work since the introduction of GPT-2 (Radford et al., 2019) has focused on scaling up model training. Initial work increases model size to hundreds of billions of parameters (Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2023), along with explorations of the tradeoff between model and training corpus size (Hoffmann et al., 2022). Since the push towards large models, work has shifted to increasing the amount of pre-training data, with new models now reaching 15 trillion tokens (AI@Meta, 2024). Studies of model performance on various tasks at different model sizes introduced the idea of emergent model abilities (Wei et al., 2022), with new model abilities being revealed as model training grows.

We also recognize a particularly important trend for this paper: model openness. Early LLMs were proprietary models accessible only through an API. The first large open model, Bloom (Bloom Ström et al., 2023), allowed widespread evaluation of these models. Subsequent open models, such as OPT (Zhang et al., 2022), LLaMA (Touvron et al., 2023; Keles and Bayraklı, 2024) and others (Biderman et al., 2023; Gururangan et al., 2023; Almazrouei et al., 2023), have become the norm. In this paper, we study OLMo (Groeneveld et al., 2024), which is one of the only models to release individual pre-training checkpoints.

Fine-Tuning Early work on instruction fine-tuning using reinforcement learning with human feedback (RLHF) (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) demonstrate the dramatic effect that model alignment could have on a pre-training model. 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. 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).

Instruction Fine-Tuning Instruction fine-tuning is preferred when more general model behaviors are desired. Popularized through reinforcement-learning with human feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) and reinforcement-learning with AI feedback (RLAIF) (Lee et al., 2023), these methods utilize a reward model to simulate human feedback. Others explore human preference tuning without a reward model (Rafailov et al., 2024; Song et al., 2024; Xu et al., 2024), or study the effects of these tuning methods (Shen et al., 2024; Perez et al., 2023). Sharma et al. (2024) show that supervised fine-tuning can lead to similar performance as RLAIF.

In-Context Learning While not the subject of this paper since it does not make changes to model parameters, in-context learning utilizes a small amount of supervised data to improve model performance. ICL, also called few-shot learning, is also used as an evaluation strategy where the model is given a prompt composed of examples of tasks expected to be solved. The underlying model is evaluated based on its response to the input. ICL can benefit from a larger context window that adds more examples, which can spur work on the development of model quantization techniques (Dettmers et al., 2022) and the alleviation of hardware constraints (Brown et al., 2020; Xie et al., 2021; Min et al., 2022).

Fine-Tuning Techniques While model pre-training can be done by a few groups with large resources interested in developing new models, fine-tuning depends on the task and is of broad interest. Therefore, many techniques are developed to facilitate time-, memory-, and data-efficient model training through parameter-efficient fine-tuning (PEFT) (Hu et al., 2021), quantization (Jacob et al., 2018; Dettmers et al., 2022, 2024), and specialized data filtering (Xia et al., 2024; Zhou et al., 2024). This paper focuses specifically on full-parameter fine-tuning, while our findings suggest the potential for data-efficient and budget-friendly training by understanding the critical turning point of model training. Our findings are closely related to the recent study on *phase transition* of model training (Olsson et al., 2022; Wei et al., 2022; Chen et al., 2023).

3 Experimental Setup

In this section, we introduce the model choice and datasets used. The hyperparameter tuning procedure and setup for each fine-tuning setting can be found in Appendix A.

3.1 Model Choice

Our paper considers OLMo-1B (Groeneveld et al., 2024), a recently released high-performing open-source large language models. Several factors motivate the selection of this model. First, OLMo is one of the only models that released intermediate checkpoints available, a prerequisite of this study [2] [3]. Second, the model is fully open, including the training code and pre-training data. Full openness allows future studies to consider related issues. Third, this model size allows us to train a model efficiently on a single A100 GPU. This study requires a large amount of GPU time, which would have been prohibitive with a larger model. We select model pre-training checkpoints uniformly from the pre-training history and include the first and the final checkpoints.

3.2 Training Procedure

We 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 instructing fine-tuning is done once using the instruction dataset. The instruction-tuned model is evaluated on a suite of LLM benchmarks.

Supervised Fine-tuning We adapt the dataset choice from Yang et al., 2024 for supervised fine-tuning. For each in-domain dataset, one to two cross-domain evaluation datasets are supplied. Each pre-training checkpoint is fully fine-tuned for 3 epochs with a batch size of 8 and learning rates resulting from minimal hyperparameter tuning. Each task is formatted using a default prompt-completion format (Table 3).

[2] https://github.com/allenai/OLMo/tree/main/checkpoints

[3] We also experimented with RedPajama-INCITE (https://www.together.ai/blog/redpajama-models-v1), one of the few, if not only, other models to release checkpoints. After extensive experiments, we found it performed worse than OLMo, given the training data available. Several other models report that they release training checkpoints but have not done so.

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}.

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[4]
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.

3.3 Evaluation

Our evaluation challenge is to select a representative number of datasets for different types of tasks to test model abilities, recognizing that each dataset requires evaluating each model checkpoint and its fine-tuned counterparts. We also select datasets based on the availability of in-domain and out-of-domain samples.

Datasets Our datasets are summarized in Table 1. 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). Each training set is sub-sampled to a size of 6,000 for fair comparisons.

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), and SciQ (Welbl et al., 2017). In addition to the datasets above, the instruction-tuned models are evaluated on LLM-BAR (Zeng et al., 2024) to test for instruction-following ability.

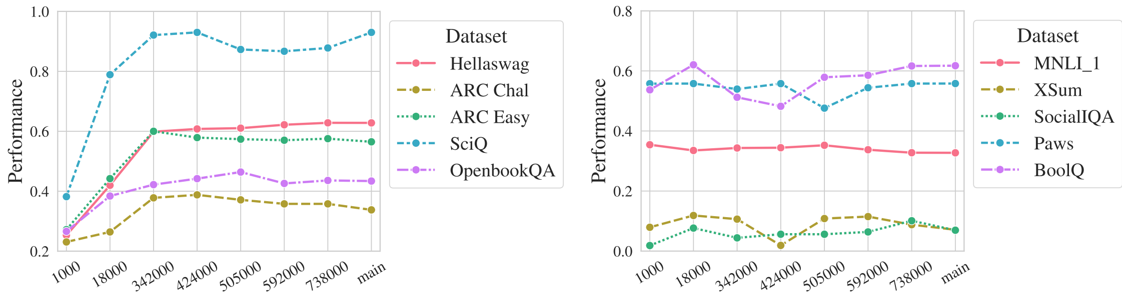
Metrics We use accuracy (Pedregosa et al., 2011) for classification tasks and ROUGE-L (Lin, 2004) for generation tasks. We set the maximum amount of newly generated tokens to 5 for classification tasks and 60 for generation tasks. Outputs are generated with greedy decoding. For classification tasks, 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.

4 How does the model change across pre-training?

We begin our evaluation by considering how the base model (no fine-tuning) changes with additional pre-training. Typically, researchers track the value of the training or held-out loss during training. However, performance improvements on downstream tasks do not always track these loss curves (Groeneveld et al., 2024).

We evaluate the pre-trained checkpoints using In-Context Learning (few-shot examples), as models without alignment tend to do poorly in a zero-shot context. We verify this by initial evaluations of the models in both zero-shot and few-shot settings. 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 2.

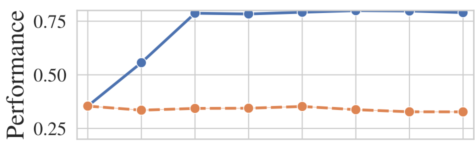
Broadly speaking, we find that all datasets fall into one of two groups. For the first group of datasets (Figure 2a), although the model shows clear improvement during the early stages of pre-training, performance levels off fairly early on and remains consistent. The dramatic improvements in the early stages of pre-training may partially come from learning rate warm-up; OLMo's learning rate is warmed up for the first 2000 steps for



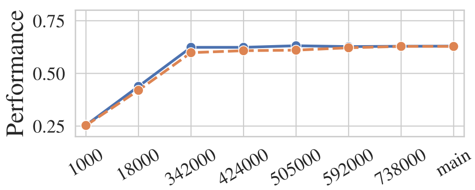
(a) Learned during Pre-training.

(b) Never learned during pre-training.

Figure 2: Few-shot performance on different pre-training steps.



(a) MNLI Matched



(b) Hellaswag

Figure 3: Example of few-shot performance on different pre-training steps of the models that benefited (3a) and did not benefit from fine-tuning (3b). 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 9 and Figure 8.

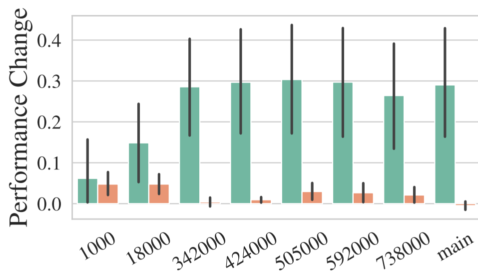


Figure 4: Amount of increase after fine-tuning between tasks that model can solve in pre-training (**mandarin orange**) and tasks that the model could not solve until fine-tuning (**sage green**). The exact number of mean increase is shown in Appendix G.

OLMo-1B of the training. We find improvements stop increasing past 342,000. The second group (Figure 2a) shows tasks that are never learned during pre-training. Performance remains constant throughout the whole pre-training process. These datasets include MNLI, XSum, and BoolQ, and we found no difference between zero-shot and few-shot evaluations.

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

5 Does more pre-training improve fine-tuning?

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 unaligned model does poorly. We observe (§4) that while some datasets improved during pre-training, there is a group of datasets for which a pre-trained model does poorly. What exactly is happening with the model on these datasets during pre-training? Does the model learn anything, and is fine-tuning required to do well on these tasks? Alternatively, does the model learn useful information for these tasks but cannot express it without fine-tuning? We explore these questions by examining fine-tuned checkpoints for each of the datasets.

Our results appear in Figure 3 and Figure 4. First, we consider those datasets which do well with a pre-trained model (Figure 2a). These datasets do not improve with fine-tuning, suggesting whatever is learned during fine-tuning, which we discuss below, the model already gains the knowledge during pre-training. We see this effect at all checkpoints; fine-tuning simply does not help.

However, a different story is observed for datasets that were not learned during pre-training. For these, fine-tuning yields significant improvements at every model checkpoint, with Figure 4 showing the magnitude of improvement on these datasets compared to no improvement to the datasets 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 3 shows representative plots comparing the performance of a pre-trained versus fine-tuned model at different checkpoints for two datasets (full list in Appendix D). 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, learning important information during pre-training, but it cannot express that information without fine-tuning.

Our findings suggest that early stopping in pre-training will not be detrimental to downstream fine-tuning performance, and **the benefits of fine-tuning an LLM can exceed the benefits of continued pretraining**, which sheds light on the potential of cost-effective fine-tuning with less pre-training. However, it is difficult to directly identify such a stopping criteria without fine-tuning intermediate checkpoints; the improvement trend is invisible before fine-tuning the checkpoints. Future work may reveal other signals of pre-training behavior that correlate with downstream task performance after fine-tuning. Overall, when resource-intensive pre-trained LLMs are not available, fine-tuning models on models with less pre-training may be a reasonable practical choice for obtaining a high-quality model.

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

What exactly is the model learning during fine-tuning such that it can reveal abilities in pre-trained models for some tasks but provide no benefit for other tasks? We analyze the supervised fine-tuning process to understand what is learned and what is forgotten. Specifically, we explore three dimensions: **task format, task transfer, and domain knowledge**.

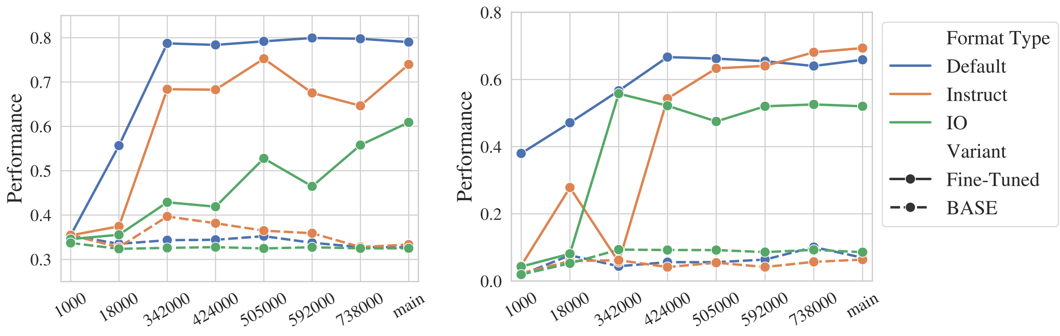
6.1 Task Format

Sclar et al., 2023 show that LLMs are extremely sensitive to prompt perturbation in few-shot settings. More broadly, extensive work on prompt engineering reveals the sensitivity of models to task format. 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 supervised fine-tuning, where we expect the model to benefit from learning the task format; 2) In contrast, raw input-output `IO` format reflects a common way of performing supervised fine-tuning by incorporating only unprocessed input and output; `Instruct` uses a human-readable instruction template to format the input. Table 3 shows an example of each format.

In the early pre-training steps, aligning the task format with fine-tuning data seems to play a crucial role. The model does not yet have enough information to overcome differences between the training and test format. However, when fine-tuned on later pre-training checkpoints, the model gradually becomes more flexible with different task formats, suggesting that model sensitivity to prompt formatting observed may be resolvable with more pre-training and a fine-tuning stage. In this view, fine-tuning teaches the model how to format a response for the task.

6.2 Task Transfer

Numerous studies examine model forgetting, where further model training causes improvements on some tasks but degradation on others (Mehta et al., 2023). 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 $\mathrm{Perf}^i_{BASE}(d)$ where $d \in D_E$, and performance of the i-th checkpoint fine-tuned on $t \in D_t$ be $\mathrm{Perf}^i_t(d)$. To normalize the effect caused by uneven performance across different datasets, we compute the mean ratio of change (MRC) in performance for each checkpoint



(a) MNLI matched

(b) SocialIQa

Figure 5: Example of model performance with different task formats. The figure of all datasets can be found in Figure 13.

as follows.

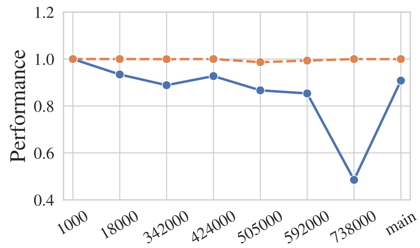
$$\text{MRC} = \frac{1}{|D_E \setminus \{t\}|} \sum_{\forall d \in D_E, d \neq t} \frac{\text{Perf}_t^i(d) - \text{Perf}_{BASE}^i(d)}{\text{Perf}_{BASE}^i(d)}$$

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.

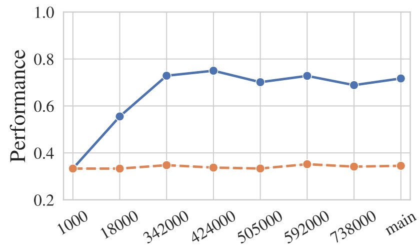
A model can maintain classification abilities when trained for generation, but it loses its generation abilities when trained for classification. This is perhaps not surprising given that classification tasks can be seen as a subset of generation, while the reverse is not true. The model follows a simplicity bias 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. Fine-tuning can cause a model to lose abilities when the desired fine-tuning behavior does not support those abilities.

6.3 Domain Knowledge

Finally, we explore how a model's generalization ability is affected by fine-tuning by inspecting whether the model forgets the domain knowledge it had before fine-tuning due to learning other abilities. An example of OOD model performance is



(a) Paws → QQP



(b) MNLI → GPT3NLI

Figure 6: 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 6a shows an example of fine-tuning hurting OOD performance, while Figure 6b shows an example of fine-tuning boosting OOD performance as pre-traininng proceeds.

shown in Figure 6, and the mean change ratio by datasets is presented in Figure 7.

The model does not benefit equally from the in-domain fine-tuning: all NLI datasets experience a boost when fine-tuning on MNLI, while fine-tuning on Paws is detrimental to other paraphrase detection datasets. This implies that both forgetting and learning are happening: 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. Questions remain,

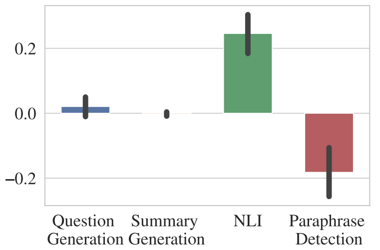


Figure 7: Ratio of out-of-domain performance change for each task, averaged across checkpoints

however, about whether there are different stages of learning and forgetting during fine-tuning and whether the model picks up different tasks in various stages, which requires further study of fine-tuning dynamics.

Overall, across these three lenses, we find that fine-tuning, although teaches a model how to perform a task, can sacrifice generalization abilities if such ability is not needed for the fine-tuned task. For some datasets learned with pre-training alone, the model can easily understand the task format, and the nature of the task is probably supported by the pre-training objective. For tasks that can only be learned with subsequent fine-tuning, the model may require additional examples to adapt to different task formats, or the task itself may be inconsistent with the pre-training objective.

7 Discussion

Our study uses fine-tuning of pre-training model checkpoints to understand the dynamics of pre-training and fine-tuning on model performance. While our insights suggest directions for future work, we note important limitations inherent in our experiments. This study considered a single, relatively small LLM on less than a dozen datasets, and still consumed thousands of hours of GPU training time at significant expense. Future work needs to confront these issues on larger models and more datasets.

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 (Section 6) 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. While we could identify what about the tasks placed them in the learned or not learnable during pre-training group, it may be possible to format tasks in a manner that better aligns with pre-training and makes them learnable.

Pre-training models can improve in undetectable ways without fine-tuning. Some datasets are not learnable during pre-training but benefit significantly from fine-tuning (§4). However, these datasets still benefited from additional pre-training, even though those benefits were not revealed without fine-tuning (§5). Clearly, the model is learning important information about the task, even though it cannot express that information. The identification of a measure available during pre-training that correlated with post-fine-tuning task performance could be used to guide pre-training and produce models that did 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 knew when to stop.

Fine-tuning teaches task format but leads to forgetting unused abilities. Our results show that fine-tuning guides the model to understand the format and complete a given task. As this information diminishes, the model's overall ability improves. However, fine-tuning comes at the expense of other model abilities, such as the capability of performing on tasks or domains that are unrelated 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).

8 Conclusion

In this work, we conduct straightforward experiments to understand the relationship between fine-tuning and pre-training LLMs. Our findings span from the latent benefits of pretraining to model learning and forgetting during fine-tuning. Overall, our results demonstrate the value of analyzing training dynamics, and we would like to call for the release of pre-training checkpoints to aid future studies.

Limitations

We discuss the weaknesses and limitations in the following section.

Computing Resource Due to computational constraints, we cannot explore larger models with richer data. The amount of GPU hours spent for each experiment in this study is listed in Table 2.

Availbility of Pre-training Checkpoints This study would benefit significantly from including a broader spectrum of models, but the public pre-training checkpoint releases are limited. Open-source LLMs with intermediate checkpoint release include OLMo (Groeneveld et al., 2024), TinyLLAMA, RedPajama-Incite, OpenLM, and Pythia. After a series of preliminary experiments, we select these models' best-performing and robust families.

Scaling Law Recent research shows that the model may resemble emergent capability (Wei et al., 2022) when scaled to a certain size. Our experiments are only conducted on the one-billion model, which may, therefore, conceal the emergent capability brought by larger models.

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. We have only looked at downstream performance as an analysis protocol. More investigation should be done into model internals during pre-training dynamics and how they relate to the effects of fine-tuning.

Training Paradigm Models are fine-tuned with a fixed amount of epochs. 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 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. Due to the computational constraints, we cannot reduce the randomness factor on this angle.

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.

AI@Meta. 2024. Llama 3 model card.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

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

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.

Eva-Marie Bloom Ström, Onelisa Slater, Aron Zahran, Aleksandrs Berdicevskis, and Anne Schumacher. 2023. Preparing a corpus of spoken Xhosa. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, pages 62–67, Gothenburg, Sweden. Association for Computational Linguistics.

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.

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*.

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*.

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, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

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

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, Mitchell Wortsman, Samir Yitzhak Gadre, Achal Dave, Maciej Kilian, Weijia Shi, Jean Mercat, Georgios Smyrnis, Gabriel Ilharco, Matt Jordan, Reinhard Heckel, Alex Dimakis, Ali Farhadi, Vaishaal Shankar, and Ludwig Schmidt. 2023. open_lm: a minimal but performative language modeling (lm) repository. GitHub repository.

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.

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*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

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*.

Onur Keles and Omer Turan Bayraklı. 2024. LLaMA-2-econ: Enhancing title generation, abstract classification, and academic Q&A in economic research. In *Proceedings of the Joint Workshop of the 7th Financial Technology and Natural Language Processing, the 5th Knowledge Discovery from Unstructured Data in Financial Services, and the 4th Workshop on Economics and Natural Language Processing @ LREC-COLING 2024*, pages 212–218, Torino, Italia. ELRA and ICCL.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

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.

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.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. 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.

Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. 2023. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13387–13434, Toronto, Canada. Association for Computational Linguistics.

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.

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.

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*.

Archit Sharma, Sedrick Keh, Eric Mitchell, Chelsea Finn, Kushal Arora, and Thomas Kollar. 2024. A critical evaluation of ai feedback for aligning large language models. *arXiv preprint arXiv:2402.12366*.

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*.

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.

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*.

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*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

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*.

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.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following. In *The Twelfth International Conference on Learning Representations*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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 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 $\{2 \times 10^{-5}, 2 \times 10^{-6}, 2 \times 10^{-7}\}$. Each model is fine-tuned for 3 epochs on the supervised fine-tuning tasks and 5 epochs on Tulu for instruction tuning. 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.

B Task Format

We adopt the task format from (Yang et al., 2024), with an additional task format of input-output.

C GPU Hours per-Experiment

We show a table of GPU hours spent for each experiment in Table 2. The total number of GPU hours spent in 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.

D Per-dataset Figures

We show the model performance on each dataset after supervised fine-tuning and instruction tuning correspondingly in Figure 9 and Figure 8. The datasets that already show improvement during pre-training do not benefit from fine-tuning, while performance improve drastically on the datasets that the model has never learned during pre-training.

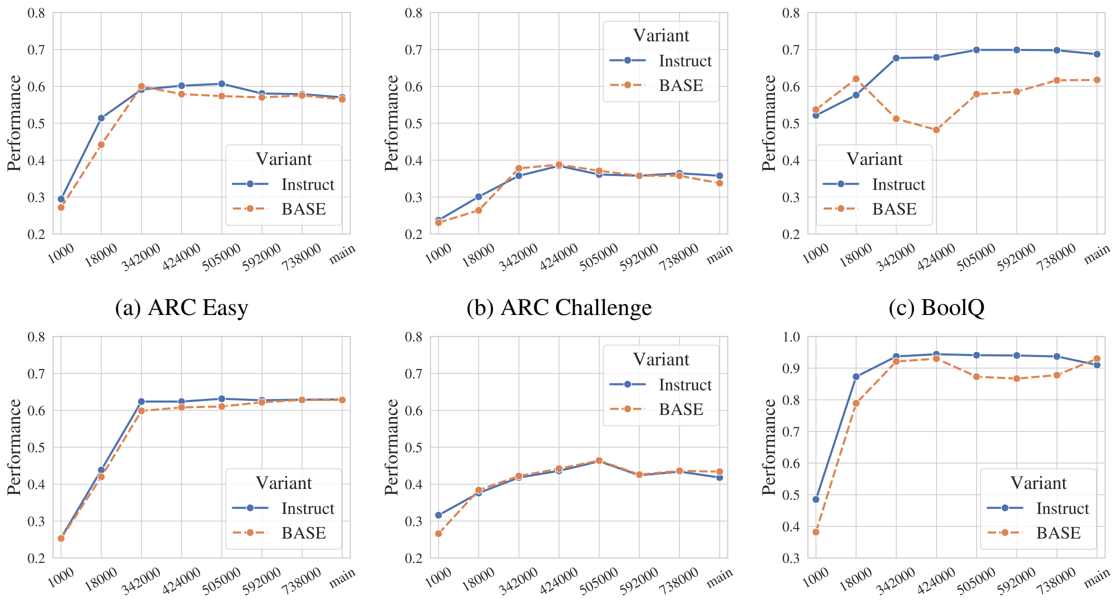
Out-of-domain Generalization The out-of-domain performance for each dataset with respect to pre-training steps is shown in Figure 10. Overall, the model generalizes well after fine-tuning on NLI tasks, while its performance deteriorates when evaluated on out-of-domain paraphrase detection tasks.

Cross-task Generalization The cross-task performance for each dataset with respect to pre-training steps is shown in Figure 11 and Figure 12.

Task-Format

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			

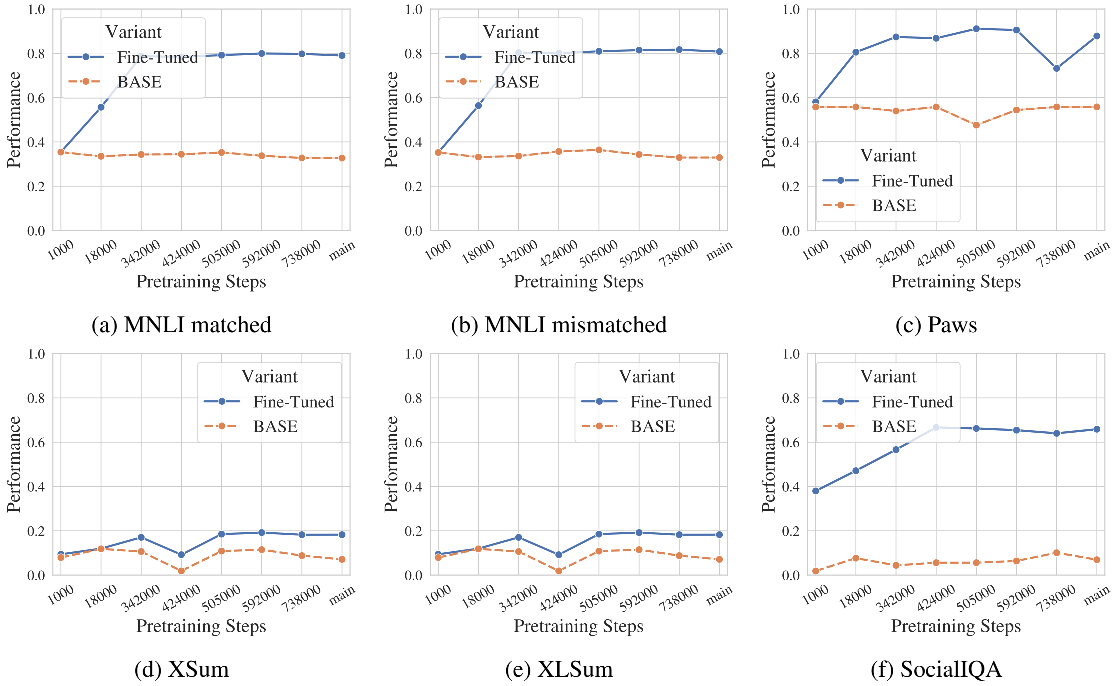
Table 2: GPU hours for each experiment. The total amount of GPU hours spent in this project is approximately 1067 A100 hours.



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

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

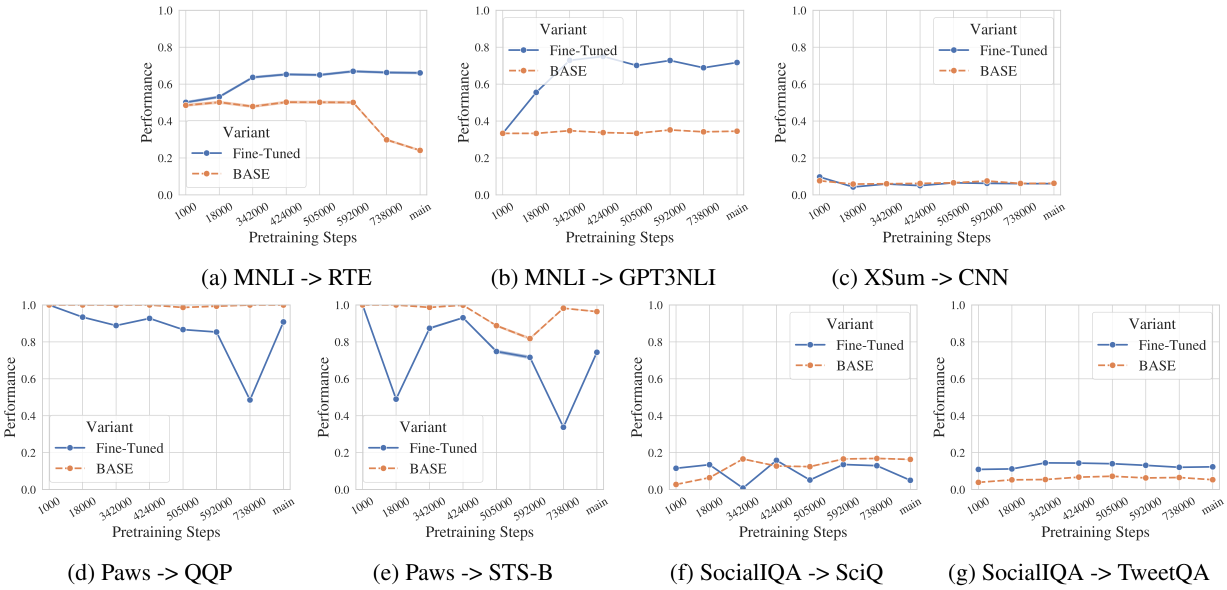
Figure 8: Model performance after instruction tuning on each pre-training step.



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

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

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



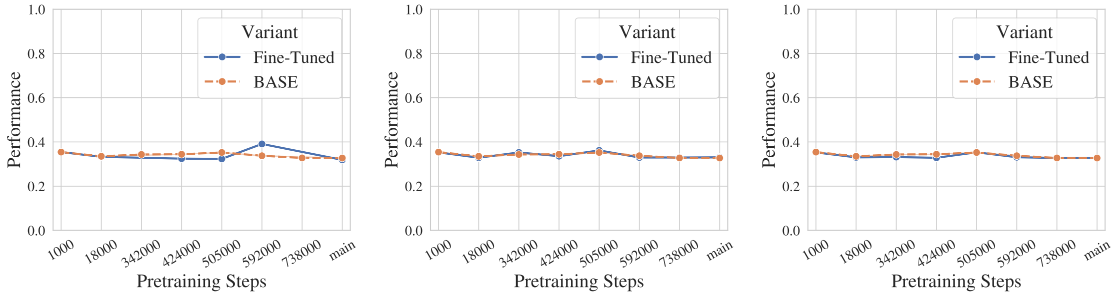
(a) MNLI -> RTE (b) MNLI -> GPT3NLI (c) XSum -> CNN

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

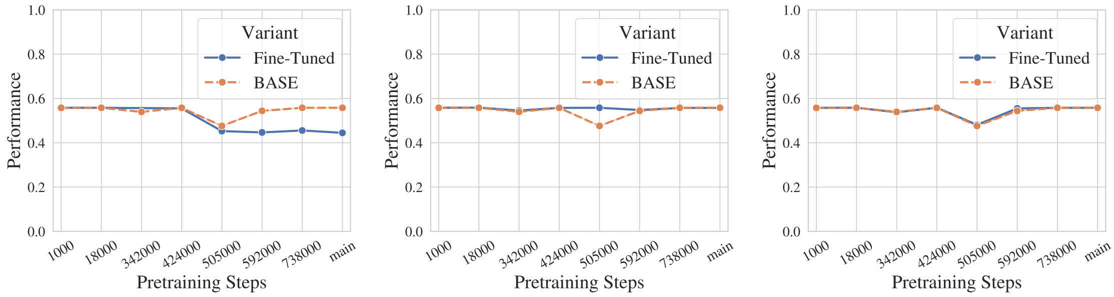
Figure 10: Out-of-domain performance after supervised fine-tuning on each pre-training step.

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 3: Formatting of the prompts

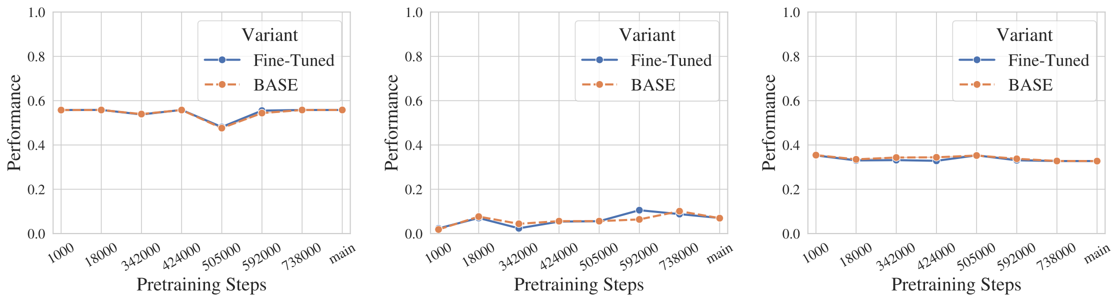


(a) Paws -> MNLI (b) SocialIQA -> MNLI (c) XSum -> MNLI

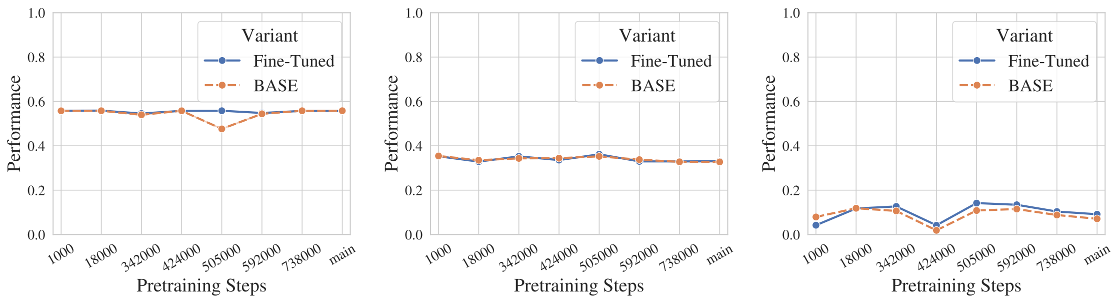


(d) MNLI -> Paws (e) SocialIQA -> Paws (f) XSum -> Paws

Figure 11: 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.

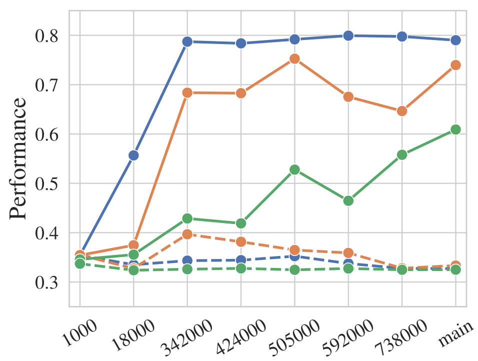


(a) Paws -> XSum (b) SocialIQA -> XSum (c) MNLI -> XSum

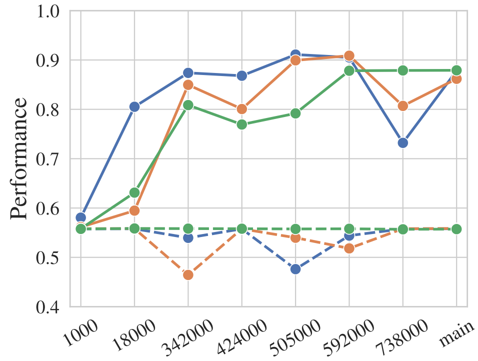


(d) Paws -> SocialIQA (e) MNLI -> SocialIQA (f) XSum -> SocialIQA

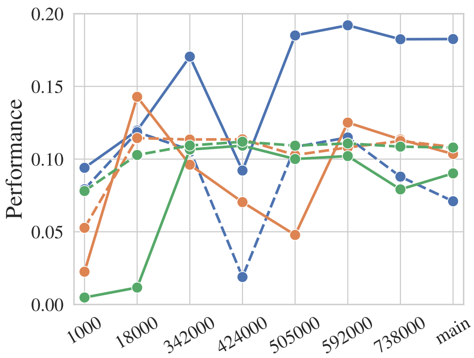
Figure 12: Cross-task performance after supervised fine-tuning on each pre-training step. The model is fine-tuned on a generation task and evaluated on a classification task.



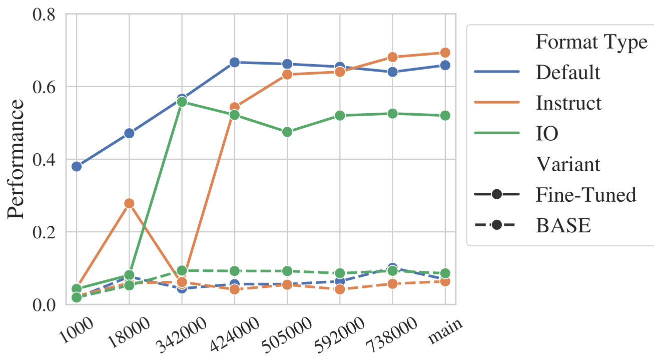
(a) MNLI matched



(b) Paws



(c) XSum



(d) SocialIQa

Figure 13: Model performance with different task formats.

E Generalization Taxonomy

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

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*		

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 4: License of artifacts used in this paper.

F License of Artifacts

We include the license of artifacts used in this paper in Table 4

G Performance Numbers

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 5: 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.

H Full Performance Table

All the exact metric numbers are shown in Table.