Improving In-Context Few-Shot Learning via Self-Supervised Training

Mingda Chen[1*] **Jingfei Du**[2] **Ramakanth Pasunuru**[2] **Todor Mihaylov**[2]
Srini Iyer[2] **Veselin Stoyanov**[2] **Zornitsa Kozareva**[2]
[1]Toyota Technological Institute at Chicago, IL, USA
[2]Meta AI
mchen@ttic.edu
{jingfeidu,rpasunuru,tbmihaylov,sviyer,ves,zori}@fb.com

Abstract

Self-supervised pretraining has made few-shot learning possible for many NLP tasks. But the pretraining objectives are not typically adapted specifically for in-context few-shot learning. In this paper, we propose to use self-supervision in an intermediate training stage between pretraining and downstream few-shot usage with the goal to teach the model to perform in-context few shot learning. We propose and evaluate four self-supervised objectives on two benchmarks. We find that the intermediate self-supervision stage produces models that outperform strong baselines. Ablation study shows that several factors affect the downstream performance, such as the amount of training data and the diversity of the self-supervised objectives. Human-annotated cross-task supervision and self-supervision are complementary. Qualitative analysis suggests that the self-supervised-trained models are better at following task requirements.

1 Introduction

In-context few-shot learning seeks to solve unseen tasks at inference time by conditioning on a few training examples. In particular, in this case we are interested in methods that forgo any weight updates (Brown et al., 2020). Prior work has been focused on improving inference time algorithms (e.g., rescoring generated outputs (Zhao et al., 2021), selecting (Liu et al., 2021) and ordering (Lu et al., 2021) the given few-shot examples) and incorporating extra resources (e.g., fine-tuning models on human-annotated datasets (Mishra et al., 2021; Ye et al., 2021; Wei et al., 2022)).

We hypothesise that a different way to improve in-context few-shot learning is through designing self-supervised objectives that more closely resemble the format of tasks that the model will be asked to perform. To do so, we cast the self-supervised training as an intermediate training stage between language model pretraining and downstream few-shot evaluation. In particular, we construct training datasets based on the self-supervised objectives following similar formats used in the downstream tasks, fine-tune pretrained language model checkpoints on the training datasets, and then evaluate the models on benchmarks.

In experiments, we consider four self-supervised objectives, including masked word prediction and classification tasks related to next sentence prediction (Devlin et al., 2019). We evaluate models on two benchmarks (13 tasks in total): SuperGLUE (Wang et al., 2019) and Natural-Instructions (Mishra et al., 2021). SuperGLUE focuses on discriminative tasks, and Natural-Instructions is a set of generative tasks.

Empirically, we experiment with pretrained language models of two sizes: 125 million parameters and 1.3 billion parameters. We show that in our best setting, the 1.3 billion parameters model trained by the self-supervision performs better than the initial pretrained language models and two strong baselines on average.

Further analysis reveals that (1) the effectiveness of the self-supervision depends on the amount of training data, but the benefit of adding more data is diminishing; (2) the improvements brought by the self-supervision are in part due to the semantic similarity between the training and evaluation tasks; (3) adding more self-supervised objectives may not help model performance because adding them does not contribute to the diversity of the self-supervised tasks; (4) choosing similar task templates for both self-supervised and downstream tasks plays a vital role in improving model performance; (5) self-supervised tasks and human-annotated datasets are complementary; (6) generation examples show that compared to the initial pretrained language models, self-supervised-trained models are better at following the task instructions.

[*] Work done during an internship at Meta AI.

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3558 - 3573
July 10-15, 2022 ©2022 Association for Computational Linguistics

2 Related Work

In-Context Few-Shot Learning. Brown et al. (2020) discover that large pretrained language models can solve unseen tasks at inference time. Recent work has improved the in-context few-shot performance by rescoring generated outputs (Zhao et al., 2021), selecting (Liu et al., 2021) and ordering (Lu et al., 2021) the given few-shot examples. Other work studies pretrained language models' cross-task generalization abilities for in-context few-shot or zero-shot learning using human-annotated datasets (Ye et al., 2021; Wei et al., 2022; Sanh et al., 2022; Min et al., 2021; Xu et al., 2022) via instructions (Weller et al., 2020; Efrat and Levy, 2020; Mishra et al., 2021; Ouyang et al., 2022) and retrieved examples (Hu et al., 2022; Lin et al., 2022). Our work differs in that we focus on self-supervised training.

Fine-Tuning for Few-Shot Learning. Pretrained language models for few-shot learning typically follows the "pretrain then fine-tune" paradigm (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019, *inter alia*), where recent work has focused on designing templates for few-shot fine-tuning (Reynolds and McDonell, 2021; Schick and Schütze, 2021a,c,b; Le Scao and Rush, 2021; Tam et al., 2021; Gao et al., 2021; Sorensen et al., 2022), and optimizing soft prompts (Li and Liang, 2021; Qin and Eisner, 2021; Lester et al., 2021; Gu et al., 2021; Zhang et al., 2022). Other work focuses on unifying task formats to maximize the benefits of human annotations, including question answering (Zhong et al., 2021), textual entailment (Yin et al., 2019, 2020; Wang et al., 2021a), and many other tasks (McCann et al., 2018; Keskar et al., 2019; Raffel et al., 2020; Bragg et al., 2021). In contrast, our focus is on in-context few-shot learning, without fine-tuning models on downstream task examples.

Pretraining for Few-Shot Learning. Several papers have adapted various resources for pretraining models to enhance their performances on few-shot learning, such as pretraining on hypertext (Aghajanyan et al., 2021b), question-infused pretraining (Jia et al., 2021), and self-training (Du et al., 2021; Vu et al., 2021; Wang et al., 2021b). Pretraining approaches have targeted specific tasks, such as task-oriented dialog (Mi et al., 2021), intent detection (Zhang et al., 2021), and data-to-text generation (Chen et al., 2020). Our work differs as we use plain text as opposed to (naturally-occurring) human-annotated resources. Relatedly, Bansal et al. (2020) used self-supervised meta-learning for few-shot text classification rather than in-context few-shot learning.

Intermediate Fine-Tuning. Since our approach involves an extra training stage between pretraining and downstream evaluation, it is also related to prior work that uses multi-stage fine-tuning on human-annotated datasets for generic tasks (Phang et al., 2018; Pruksachatkun et al., 2020; Chang and Lu, 2021; Aghajanyan et al., 2021a; Poth et al., 2021) and text classification (Zhang and Zhang, 2021). Relevant work also studies intermediate fine-tuning using crosslingual supervision (Phang et al., 2020; Moghe et al., 2021). Rubino and Sumita (2020) use an intermediate self-supervised training stage for machine translation quality estimation.

3 Method

We describe four self-supervised training objectives that will be used to train models before downstream evaluations.

We begin by defining the example and the instance used during our self-supervised training. An **example** is an input-output pair. To differentiate the input and the output, we append special tokens "Input:" and "Output:" to the beginning of input text and output text respectively where the two texts are also separated by the ⟨newline⟩ token (see Figure 1 for examples).[1]

An **instance** is a linearized string formed by several examples from the same task (e.g., see Figure 2). As we encode the text using causal attention, the examples closer to the beginning of input sequences can be seen as task demonstrations, resulting in efficient computation.

When constructing the training examples, we pick three or more consecutive sentences (depending on the minimum sequence length we enforce on the sentences) and then apply task-specific rules to automatically create training data. To form a training instance, we randomly select examples from the same task until reaching the maximum sequence length (i.e., 2048). During training, we compute a cross-entropy loss on tokens in the **output texts**.

[1] We chose this special symbol because we always start the self-supervised training from a pretrained language model checkpoint.

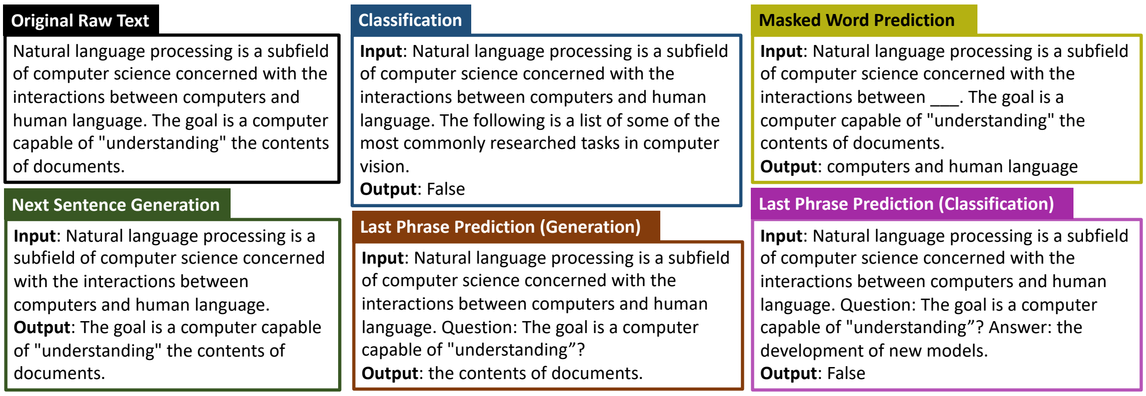

Figure 1: Examples of our self-supervised training tasks. Each example is an input-output pair constructed from the raw text.

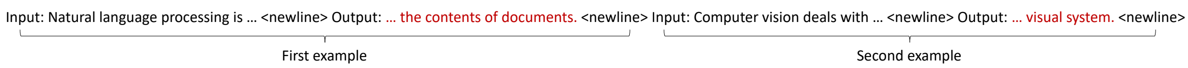

Figure 2: An example of a training instance. Each instance is formed by several training examples. During training, we use left-to-right language models and compute a cross-entropy loss on the output texts (indicated by the red color in the shown example). We note that when computing the loss on the second example, the first example can be seen as task demonstrations. For brevity, we show part of the input and output texts.

We describe details of the self-supervised tasks in the following subsections.

3.1 Next Sentence Generation

In light of the strong performance of language models on in-context few-shot learning (Brown et al., 2020), we incorporate the language modeling as one of our self-supervised tasks, which we call "next sentence generation" (NSG). NSG asks the model to generate the next sentence given previous sentences as context. When building data for this task, we use the last sentence as output and the rest of the sentences as input.

3.2 Masked Word Prediction

The second task we consider is based on masked word prediction (MWP) which is commonly used in pretraining generic text encoders (Devlin et al., 2019; Liu et al., 2019). The task asks the model to fill in the missing information based on the surrounding context. Specifically, MWP randomly replaces words in input sentences with a special symbol and requires models to recover the masked words in the input. For this task, we create input text by randomly replacing 1~20 words in the input text with a special token[2] and use the masked out words as the output text.

3.3 Last Phrase Prediction

Inspired by the LAMBADA dataset (Paperno et al., 2016), a question answering dataset which asks models to predict the last word in a sentence given several sentences of context, we create a "last phrase prediction" (LPP) task, which requires predicting the last phrase in a sentence. To solve this task, models need to draw relevant information from the context and the learned knowledge during pretraining. We cast LPP as either a generation task or a classification task. The latter variant of LPP is a binary classification task that labels if the given answer is the correct phrase. To facilitate a unified format of these two tasks, we append a special token "Question:" to the beginning of the last sentence and replace the last phrase with a question mark. For the classification LPP, we separate the given answer and the previous context and sentences with a special token "Answer:". An example of this task is shown in Figure 1.

More specifically, we identify the last phrase of a sentence based on a set of function words (see appendix A.5 for the list of function words). If there are multiple function words in a sentence, we pick the last one. Then we treat the text segment starting from the function word as the last phrase.[3]

[2] We randomly select the special token from the following list: ___, ⟨⟨⟩⟩, @@@, (()), $$$, %%%, ###, ***, and +++. We use random symbols instead of a fixed symbol because we found that it gives better performance in our preliminary experiments.

[3] We ensure that the last sentence in raw text for this task always has at least one valid function word and the function

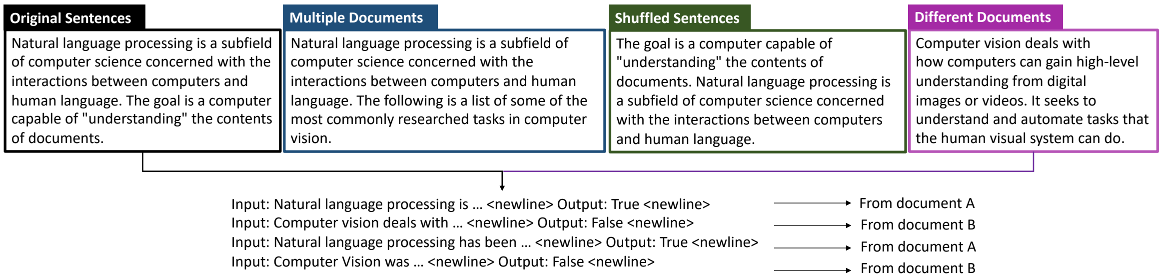

Figure 3: Example illustrating the construction of training instances for our classification task. There are four input types, and each training instance has two or three types. As the shown instance has the following two types: "original sentences" and "different documents", it comprises examples from two different documents. The instance resembles the next sentence prediction task, encouraging models to compare topical similarities between the two examples.

When selecting negative answers, we randomly choose from the phrases extracted from the same function words (to make the negative answers more challenging).

3.4 Classification

Similar to the next sentence prediction task (Devlin et al., 2019) and the sentence ordering prediction task (Jernite et al., 2017; Chen et al., 2019) for pretraining language representations, we create a classification task (CL) for our self-supervised training. As shown in Figure 3, for this task, we consider four types of input: original sentences, shuffled sentences, sentences from a different document, and sentences from multiple documents. In particular, for original sentences, we directly use text from original human-written documents. For shuffled sentences, we randomly shuffle all the input sentences. For sentences from multiple documents, we randomly replace 50% of the input sentences with sentences from another document. We also ensure that the selected sentences (from both the input and another document) are consecutive in their original documents. For sentences from different documents, we replace the input sentences with sentences from another document. See Figure 3 for an example of each type of input.

When constructing a training instance, we randomly pick one or two additional input types and combine them with the original sentences to form a binary or three-way classification task. We also randomly assign label strings to input types in each instance to ensure that models follow the information given by earlier examples when making predictions.

The classification task is different from the other self-supervised tasks described in earlier subsections. It explicitly requires models to compare inputs across examples in a training instance to determine if the given input shares similar properties with the others.

4 Experiment

4.1 Training Setup

For the pretrained language model checkpoints, we use the 125 million parameters (125M) and the 1.3 billion parameters (1.3B) dense model from Artetxe et al. (2021). These pretrained models have shown results comparable to GPT3 across various tasks.

For self-supervised training, we use a subset of documents from the RoBERTa training corpus (Liu et al., 2019) that contains four domains: BOOKCORPUS plus Wikipedia, CC-NEWS, OPENWEBTEXT, and STORIES. Specifically, we randomly sample 100k documents from each domain except STORIES where we only sample 10k documents as the documents there are much longer than the others. The final training data contains approximately 1 million instances with 250k training instances per task.[4] For the 125M model, we train for 10 epochs, which takes roughly 1 day on a V100 GPU. For the 1.3B model, we train for 5 epochs, which takes roughly 3 days on 2 V100 GPUs.

4.2 Evaluation Setup

The instance and example during evaluation shares similar definition as those in Sec. 3 except that each evaluation instance has only one example from test splits and it is placed at the last position in the instance. The other examples in the instance

word lies at the second half of the sentence.

[4]The average numbers of example per instance for each data source are: 6.9 for BOOKCORPUS plus Wikipedia, 5.3 for CC-NEWS, 3.5 for OPENWEBTEXT, and 7.2 for STORIES.

GPT3	\${Context}⟨newline⟩\${Question}⟨newline⟩ - **[\${Label}] \${Answer}**
Ours	Input: \${Context} Question: \${Question} Answer: \${Answer}⟨newline⟩Output: **\${Label}**

Table 1: Evaluation templates for MultiRC. \${·} represents values drawn from a particular data field. We alter the GPT3 template for this task to share a similar format with one of our self-supervised tasks (i.e., classification LLP in this case). The red, boldfaced texts are used to compute the language modeling perplexities for ranking the labels. We note that the shown template is for a single example, and there could be multiple examples within an instance.

(i.e., task demonstrations) come from either training splits or task-specific instructions depending on benchmarks.

We evaluate the models on two benchmarks: SuperGLUE and Natural-Instructions. SuperGLUE is a set of tasks focusing on natural language understanding. We use BoolQ (BQ; Clark et al., 2019), CB (De Marneffe et al., 2019), COPA (CA; Roemmele et al., 2011), MultiRC (MC; Khashabi et al., 2018), and RTE (RE; Giampiccolo et al., 2007; Bentivogli et al., 2009; Dagan et al., 2006; Bar Haim et al., 2006).[5] We report results for the official development sets. The task demonstrations are examples randomly selected from the training sets. We report mean and standard deviations of five runs with different random seeds. Following GPT3, we use a ranking based approach when evaluating the models (i.e., pick the best label based on language modeling perplexities).

Natural-Instructions. Natural-Instructions evaluates models' cross-task generalization abilities where all the tasks are generation tasks. It splits the tasks into two groups for training and evaluation. We use the same task split and evaluate models on the following task categories: question generation (QG), answer generation (AG), minimal modification (MM), and verification (VF).[6] Each task category has two tasks. Following the few-shot setting used in Mishra et al. (2021), we evaluate models using 100 examples per task, use greedy decoding, and report ROUGE-L (Lin, 2004) scores per task category. For task demonstrations, we use the positive examples in the instructions in Natural-Instructions.

SuperGLUE. As our self-supervised tasks are formatted as input-output pairs, we change the task-specific templates for SuperGLUE to make them more similar to our self-supervised tasks. For example, as shown in Table 1, we make MultiRC similar to the classification LPP. More details of the template changes are in appendix A.6.

For both benchmarks, we also report an averaged performance for each model. For SuperGLUE, the average performance is computed based on the means of task performances. When a task has two metrics, we take the average of the two as the task performance.

More details on the dataset statistics and metrics for each task for both benchmarks are in appendix A.2.

Baselines. We consider four baselines: (1) directly evaluating pretrained language models on the benchmarks (LM) ; (2) performing additional language modeling training on the subset of the original data that is used for constructing the self-supervised tasks (ExtraLM). We use ExtraLM to approximately measure the contribution of additional computation; (3) fine-tuning on training sets for the tasks outside the evaluation sets (CrossTask). We use CrossTask to estimate the performances of cross-task supervision from human-annotated datasets; and (4) fine-tuning on training sets for the tasks in the evaluation sets (SameTask). SameTask serves as an oracle baseline estimating the approximated upperbound performances of cross-task supervision.

Since SuperGLUE does not have an official split for the CrossTask setting, we split the datasets into two groups according to the task category and report the CrossTask results based on "CrossTask (QA→NLI)" and "CrossTask (NLI→QA)".[7] As we alter the task templates, we report results for evaluating the pretrained language model check-

[5]We exclude WSC (Levesque et al., 2011) and ReCoRD (Zhang et al., 2018) as pretrained models, including GPT3, require scoring algorithms at inference time to achieve competitive results. We exclude WiC (Pilehvar and Camacho-Collados, 2019) because GPT3-like models, including GPT3 and our models, do not give accuracies significantly better than random baselines.

[6]We discard training tasks that share the same source datasets with evaluation tasks as we found that tasks with the same source dataset may contain leaked labels. We exclude the binary classification tasks because the class labels are severely imbalanced (i.e., more than 80% of the class labels belong to one category).

[7]"QA→NLI" suggests that we train models on the NLI tasks and evaluate on the QA tasks. Similarly, for "NLI→QA", we train models on the QA tasks and evaluate on the NLI tasks.

Model	MS	BoolQ	MultiRC	COPA	RTE	CB	Avg.
LM	125M	52.1(1.7)	5.2(0.7)/49.5(1.1)	**67.6(2.3)**	52.0(1.2)	50.7(3.2)/34.8(2.5)	48.4
ExtraLM	125M	51.5(1.7)	5.1(0.8)/49.7(1.0)	68.0(1.6)	52.3(1.2)	49.5(4.6)/35.5(5.6)	48.3
NewTemplate	125M	52.2(1.8)	5.2(0.6)/47.9(1.4)	63.0(2.5)	50.8(2.0)	46.4(7.3)/30.1(6.4)	46.2
CrossTask(NLI→QA)	125M	38.1(0.3)	5.1(0.7)/43.5(2.5)	65.4(2.1)	-	-	42.2
CrossTask (QA→NLI)	125M	-	-	-	**53.6(0.5)**	39.6(1.5)/19.9(1.2)	
SameTask	125M	71.2	19.9/66.9	72.0	67.3	71.4/60.2	61.9
Self-Supervised	125M	**55.7(0.6)**	**7.0(1.0)/60.2(0.3)**	**67.6(2.1)**	53.0(1.5)	**50.0(5.2)/39.8(3.0)**	**51.0**
LM	1.3B	48.6(2.3)	5.5(0.5)/53.7(0.7)	83.4(1.7)	51.9(1.2)	53.6(5.2)/37.2(3.7)	51.8
ExtraLM	1.3B	49.6(1.9)	4.9(0.6)/54.8(0.6)	82.6(1.5)	52.9(1.9)	51.4(7.5)/35.6(5.3)	51.7
NewTemplate	1.3B	51.3(1.3)	5.0(0.4)/52.8(1.2)	81.2(2.4)	50.8(2.3)	49.3(4.7)/33.7(4.2)	50.7
CrossTask(NLI→QA)	1.3B	53.4(0.8)	1.2(0.3)/57.2(0.3)	76.2(2.9)	-	-	49.6
CrossTask (QA→NLI)	1.3B	-	-	-	**54.3(1.2)**	44.6(3.6)/25.2(4.9)	
SameTask	1.3B	77.1	27.5/71.6	85.0	68.1	75.2/64.3	69.9
Self-Supervised	1.3B	**61.7(0.3)**	**5.2(0.1)/62.1(0.3)**	**84.0(2.7)**	53.1(0.7)	**54.3(2.0)/37.0(1.9)**	**55.6**

Table 2: SuperGLUE results. We report mean and standard deviations (the numbers in parenthesis) of five runs. The best result (we take the average if there are two metrics) except SameTask in each column for each model size is boldfaced. MS=model size.

Model	MS	QG	AG	MM	VF	Avg.
GPT3	-	43.0	50.0	70.0	32.0	48.8
LM	125M	33.7	12.9	53.0	14.7	28.6
ExtraLM	125M	**34.4**	13.4	53.7	14.3	28.9
CrossTask	125M	22.0	**24.8**	66.9	17.9	**32.9**
SameTask	125M	54.8	42.3	77.3	78.3	63.2
SelfSup.	125M	16.9	14.6	**70.1**	**18.9**	30.0
LM	1.3B	40.9	32.5	74.0	27.8	43.8
ExtraLM	1.3B	41.1	32.7	**75.9**	25.2	43.7
CrossTask	1.3B	38.1	41.6	69.2	23.0	42.9
SameTask	1.3B	55.5	64.6	81.0	80.4	70.4
SelfSup.	1.3B	**43.9**	**37.5**	72.3	**28.6**	**45.5**

Table 3: Natural-Instructions results. The results for GPT3 are taken from Mishra et al. (2021). The best result except SameTask in each column for each model size is boldfaced. MS=model size.

Model	BQ	MC	CA	RE	CB	Avg.
LM	52.2	26.6	63.0	50.8	38.3	46.2
SelfSup.	55.7	33.6	67.6	53.0	44.9	51.0
NSG	52.1	25.9	64.0	51.0	41.2	46.9
CL	52.5	26.8	61.4	50.9	48.1	47.9
MWP	51.9	26.3	61.8	50.8	36.1	45.4
LPP	53.5	29.5	61.6	52.0	40.3	47.4

Table 4: SuperGLUE results when training the 125M model with only one of the self-supervised tasks.

points using the new templates (NewTemplate) to study the effect of new templates.

4.3 Results

We report the results for SuperGLUE and Natural-Instructions in Table 2 and Table 3. Our findings are as follows:

1. Our proposed self-supervised training achieves the best performance on average for both benchmarks.
2. ExtraLM and NewTemplate show similar performances as the pretrained language model checkpoints, suggesting that the improvements from our self-supervised training is unlikely to come from the additional training on the data and the task template changes.
3. Compared to the pretrained language model checkpoints, CrossTask shows worse performances on both benchmarks, which is likely due to the differences between training tasks and evaluation tasks.

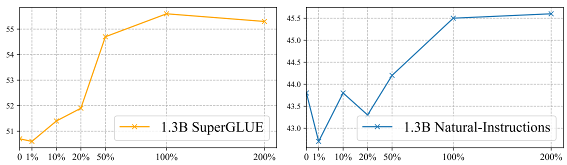

Figure 4: Average results for the 1.3B model on SuperGLUE and Natural-Instructions when varying the number of examples used for self-supervised training.

Model	QG	AG	MM	VF	Avg.
LM	33.7	12.9	53.0	14.7	28.6
SelfSup.	16.9	14.6	70.1	18.9	30.0
NSG	32.3	12.5	54.0	13.8	28.2
CL	8.3	0.3	1.0	2.7	3.1
MWP	15.2	19.4	50.5	17.8	25.7
LPP	11.3	16.6	49.5	19.9	24.3

Table 5: Natural-Instructions results when training the 125M model with only one of the self-supervised tasks.

5 Analysis

5.1 Effect of Amount of Data

In Figure 4, we report model performances for the 1.3B model on SuperGLUE and Natural-Instructions with 1%, 10%, 20%, 50%, and 200%

	ALL	ALL+MoreTask
SuperGLUE Results		
125M	51.0	50.9
1.3B	55.6	55.6
Natural-Instructions Results		
125M	30.0	31.7
1.3B	45.5	45.4

Table 6: Average results when adding denoising autoencoding and gap sentence prediction to the self-supervised training. ALL: use all of the self-supervision described in Sec. 3.

	LM	Correct Label	Random Label
SuperGLUE Results			
125M	46.2	51.0	38.2
1.3B	50.7	55.6	42.5
Natural-Instructions Results			
125M	28.6	30.0	19.1
1.3B	43.8	45.5	31.5

Table 7: Average model performance comparing whether we assign random labels to the self-supervised tasks.

of training examples.[8] We train the models for ten epochs.[9] As shown in the figure, when the amount of training data for self-supervised tasks is similar to that for the CrossTask setting (i.e., 1% data), the self-supervised tasks also lead to worse performances. The improvements become clearer when we increase the number of training data, but it begins to plateau at around 100% data. This suggests that one of the advantages of the self-supervised tasks compared to the tasks in the CrossTask setting is the amount of training data. We hypothesize that further increasing the amount of data not being helpful is because the data used for constructing the self-supervised tasks has already been used for language model pretraining. So, our models manage to learn to solve these tasks with a relatively limited amount of data. We have similar observations for the 125M model. See appendix A.8 for more details.[10]

5.2 Effect of Individual Self-Supervised Tasks

We investigate the effect of individual self-supervised tasks by training models with only one

	MS	GPT3 Template	Our Template
LM	125M	48.4	46.2
SelfSup	125M	47.2	51.0
LM	1.3B	51.8	50.7
SelfSup	1.3B	51.1	55.6

Table 8: Average results for SuperGLUE when using different task templates. MS=model size.

Model	MS	QG	AG	MM	VF	Avg.
LM	125M	**33.7**	12.9	53.0	14.7	28.6
CrossTask	125M	22.0	24.8	66.9	17.9	32.9
SelfSup.	125M	16.9	14.6	70.1	**18.9**	30.0
Combined	125M	23.5	**25.2**	**70.3**	18.5	**34.4**
LM	1.3B	40.9	32.5	74.0	27.8	43.8
CrossTask	1.3B	38.1	41.6	69.2	23.0	42.9
SelfSup.	1.3B	**43.9**	37.5	72.3	28.6	45.5
Combined	1.3B	42.1	**42.5**	**74.1**	**28.7**	**46.9**

Table 9: Natural-Instructions results when combining the self-supervised tasks and the tasks in the CrossTask setting. The best performance in each column for each model size is boldfaced. MS=model size.

task. We report the experiment results in Table 4 and Table 5. More results and discussions are in appendix A.9. Our findings are:

1. Combining all four self-supervised tasks results in the biggest improvements for most tasks, suggesting that the tasks are complementary.
2. Each self-supervised task improves a few downstream task performances (e.g., NSG helps COPA; CL helps MultiRC and CB). This is likely due to similarities between tasks.
3. It is worth noting that while CL hurts model performances on Natural-Instructions, it helps on the SuperGLUE. We hypothesis that this is because unlike Natural-Instructions, SuperGLUE is ranking based and, therefore, more favorable to classification-related training.
4. It is interesting to see that NSG and CL tasks are the two most beneficial to downstream performance among the four self-supervised tasks. This is likely due to (1) the generic task formulation of NSG, and (2) CL requires different inference abilities compared to the other self-supervised tasks. It is also interesting that training on only one of the self-supervised tasks can hurt the performance on Natural-Instruction.

5.3 Effect of More Self-Supervised Tasks

To investigate the effect of having more self-supervised tasks during training, we add two extra self-supervised tasks to the self-supervised training,

[8] We apply the same ratio to all the self-supervised tasks and use the same development sets for each task across these settings.

[9] Upon manual inspection, we found that the development set loss values in these experiments have converged.

[10] Our goal for this analysis is to show the rough trends of model performance when varying the amount of training data, rather than to provide an exact estimate of the training data required for the self-supervised training.

Task Prompt	Task Input	Reference	LM	Self-Supervised
Construct a question from the given fact by a simple rearrangement of words.	Fact: Pollen seeds come from male gametes of plants.	what seeds come from male gametes of plants?	What might cause harm to plants?	What would you use to measure the number of male gametes of plants?
Ask a question on "event duration" based on the provided sentence.	Sentence: At the sight of the great man, Spear flushed crimson, and then his look of despair slowly disappeared; and into his eyes there came incredulously hope and gratitude.	How long did Spear see the great man?	How long did he stay in the Embassy?	How long did it take for Spear to look at the great man?
Answer the given question. Your answer must be a single span in the passage.	Passage: ... The following year he won a scholarship to the Royal Academy of Music, ... The principal of the Academy, Sir Alexander Mackenzie, had forbidden ... Question: What was the full name of the school Sir Alexander Mackenzie was principal of?	Royal Academy of Music.	Oliver.	the Royal Academy of Music.
Answer the given question. Your answer must be a single span in the passage.	Passage: ... Epitaph Records, founded by Brett Gurewitz of Bad Religion, was the base for many future pop punk bands ... The mainstream pop punk of latter-day bands such as Blink-182 is criticized by many punk rock devotees; in critic Christine Di Bella's words ... Question: What is the full name of the person that is very critical of modern mainstream pop punk bands?	Christine Di Bella.	the "Bad Religion".	many punk rock devotees.

Table 10: Generation examples by the 1.3B model. The examples are taken from Natural-Instructions. The first two examples are from QG, and the other two are from AG. We only show part of the passages relevant to the outputs for QA for brevity.

following the same procedure as the other tasks. The additional tasks are: denoising autoencoding (Lewis et al., 2020) and gap sentence generation (Zhang et al., 2020). Denoising autoencoding is the task of reconstructing the original sentences from sentences corrupted by random noises, which has been shown effective for training generic language representations; gap sentence generation is to recover the missing sentence and has been found useful for abstractive summarization.

We report the results in Table 6 where we do not find adding the two tasks improves downstream tasks. This is likely because the two tasks share similarities with our existing tasks (e.g., gap sentence generation shares a similar inference style as MWP). So, adding them does not promote diversity in the self-supervised tasks, leading to the fact that the models are not encouraged to learn different information.

5.4 Effect of Few-Shot Templates

The self-supervised training brings two benefits: making models familiar with the few-shot templates and task semantics. To differentiate the effect of the two, we train models on the self-supervised tasks with random labels. For example, for NSG, we use random sentences as outputs rather than the true next sentences; for the binary classification tasks, we randomly select binary labels. As shown in the results in Table 7, random labels hurt model performances, suggesting that what the models have learned is more than the few-shot templates.

We also investigate the effect of task templates for SuperGLUE by evaluating models using different templates. We report results in Table 8 where we find that having the templates for downstream tasks similar to the ones used for self-supervised training gives the models significantly better performances.

5.5 Zero-Shot vs. One-Shot vs. Few-Shot

We show zero-shot, one-shot, and few-shot performances for the LM and the self-supervised model in Table 11. We find that among the three settings, the self-supervised training is the most helpful in the few-shot setting and does not help in the zero-shot setting, suggesting that the self-supervised training improves the models' in-context learning capabilities.

	Zero-Shot		One-Shot		Few-Shot	
	LM	SS	LM	SS	LM	SS
125M	46.7	44.3	42.6	46.1	46.2	51.0
Δ	(-2.4)		(+3.5)		**(+4.8)**	
1.3B	49.5	49.9	46.5	50.8	50.7	55.6
Δ	(+0.4)		(+4.3)		**(+4.9)**	

Table 11: Average results for SuperGLUE showing the zero-shot, one-shot, and few-shot model performances for the LM and the self-supervised model (SS). The numbers in parenthesis are the performance differences between the LM and the SS with the positive numbers indicating improvements. We boldface the largest improvement for each model.

5.6 Combine Self-Supervision with Cross-Task Human-Supervision

We investigate the relations between the self-supervised tasks and the human-annotated tasks. We combine the tasks from the self-supervision and those from the CrossTask and report the results in Table 9. Interestingly, combining the two kinds of tasks results in better performances on average, showing that they are complementary.

5.7 Generation Examples

We show generation examples in Table 10. In general, we find that compared to the vanilla pretrained language models, the self-supervised models are better at using information from task input following task requirements. Specifically, for the first two examples in Table 10, the LM suffers from more severe semantic drift than the self-supervised model (e.g., “male gametes of plants” is more specific and relevant to the task input than “plants”). We have similar observations for the third example, where “Oliver” is a name from the task demonstration rather than the passage. Interestingly, for the last example, the answer generated by the LM is from the passage but is actually “the base for many future pop punk bands” instead of what the question looks for (i.e., “very critical of modern mainstream pop punk bands”). While the answer generated by the self-supervised model does not exactly match the reference, it is partially correct as the mainstream pop punk “is criticized by many punk rock devotees”.

6 Conclusion

We evaluated four self-supervised objectives on two few-shot benchmarks by casting the self-supervised training as an intermediate training stage between language model pretraining and downstream few-shot evaluation. Empirically, we have shown that the models trained by the self-supervised objectives show the best performances compared to strong baselines on average. Analysis showed that (1) the amount of self-supervised training data and the diversity of the self-supervised tasks can affect the downstream performances.; (2) the self-supervised tasks are complementary to the human-annotated datasets; and (3) the self-supervised-trained models are better at following task requirements.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021a. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021b. Htlm: hyper-text pre-training and prompting of language models. *arXiv preprint arXiv:2107.06955*.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534, Online. Association for Computational Linguistics.

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.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. In *Advances in Neural Information Processing Systems*, volume 34, pages 15787–15800. Curran Associates, Inc.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ting-Yun Chang and Chi-Jen Lu. 2021. Rethinking why intermediate-task fine-tuning works. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 706–713, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mingda Chen, Zewei Chu, and Kevin Gimpel. 2019. Evaluation benchmarks and learning criteria for discourse-aware sentence representations. 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 649–662, Hong Kong, China. Association for Computational Linguistics.

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

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.

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.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. 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 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The Commitment-Bank: Investigating projection in naturally occurring discourse. To appear in proceedings of Sinn und Bedeutung 23. Data can be found at https://github.com/mcdm/CommitmentBank/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, Online. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

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, Prague. Association for Computational Linguistics.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. *arXiv preprint arXiv:2203.08568*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. 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 2391–2401, Hong Kong, China. Association for Computational Linguistics.

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Robin Jia, Mike Lewis, and Luke Zettlemoyer. 2021. Question answering infused pre-training of general-purpose contextualized representations. *arXiv preprint arXiv:2106.08190*.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying question answering, text classification, and regression via span extraction. *arXiv preprint arXiv:1904.09286*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. 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 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2017. Learning what is essential in questions. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 80–89, Vancouver, Canada. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Alexander Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI*.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Chen-Yu Lee, Chun-Liang Li, Chu Wang, Renshen Wang, Yasuhisa Fujii, Siyang Qin, Ashok Popat, and Tomas Pfister. 2021. ROPE: Reading order equivariant positional encoding for graph-based document information extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 314–321, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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.

Bill Yuchen Lin, Kangmin Tan, Chris Miller, Beiwen Tian, and Xiang Ren. 2022. Unsupervised cross-task generalization via retrieval augmentation. *arXiv preprint arXiv:2204.07937*.

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.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Fei Mi, Wanhao Zhou, Lingjing Kong, Fengyu Cai, Minlie Huang, and Boi Faltings. 2021. Self-training improves pre-training for few-shot learning in task-oriented dialog systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1887–1898, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Nikita Moghe, Mark Steedman, and Alexandra Birch. 2021. Cross-lingual intermediate fine-tuning improves dialogue state tracking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1137–1150, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Preprint*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Jason Phang, Iacer Calixto, Phu Mon Htut, Yada Pruksachatkun, Haokun Liu, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. English intermediate-task training improves zero-shot cross-lingual transfer too. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 557–575, Suzhou, China. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to pre-train on? Efficient intermediate task selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10585–10605, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Raphael Rubino and Eiichiro Sumita. 2020. Intermediate self-supervised learning for machine translation quality estimation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4355–4360, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021c. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Taylor Sorensen, Joshua Robinson, Christopher Michael Rytting, Alexander Glenn Shaw, Kyle Jeffrey Rogers, Alexia Pauline Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. 2022. An information-theoretic approach to prompt engineering without ground truth labels. *arXiv preprint arXiv:2203.11364*.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tu Vu, Minh-Thang Luong, Quoc Le, Grady Simon, and Mohit Iyyer. 2021. STraTA: Self-training with task augmentation for better few-shot learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5715–5731, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021a. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.

Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2021b. List: Lite self-training makes efficient few-shot learners. *arXiv preprint arXiv:2110.06274*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online. Association for Computational Linguistics.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. 2022. Zeroprompt: Scaling prompt-based pretraining to 1,000 tasks improves zero-shot generalization. *arXiv preprint arXiv:2201.06910*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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 3914–3923, Hong Kong, China. Association for Computational Linguistics.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020.

Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239, Online. Association for Computational Linguistics.

Jianguo Zhang, Trung Bui, Seunghyun Yoon, Xiang Chen, Zhiwei Liu, Congying Xia, Quan Hung Tran, Walter Chang, and Philip Yu. 2021. Few-shot intent detection via contrastive pre-training and fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1906–1912, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint 1810.12885*.

Shiwei Zhang and Xiuzhen Zhang. 2021. Does QA-based intermediate training help fine-tuning language models for text classification? In *Proceedings of the The 19th Annual Workshop of the Australasian Language Technology Association*, pages 158–162, Online. Australasian Language Technology Association.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2856–2878, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. 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 3363–3369, Hong Kong, China. Association for Computational Linguistics.

A Appendix

A.1 Additional Details for LPP and Classification Tasks

The label strings we used for LPP are as follows: Yes and No, Y and N, True and False, and T and F. We randomly choose from Yes, Y, True, and T as the label string for the positive label and use the other one in the selected pair as the negative label.

The label strings we used for the binary classification task are the same as the classification LPP task. For the three-way classification task, we use the following label strings: Positive and Negative and Neutral, True and False and Neither, T and F and N, Yes and No and Unknown, Y and N and U.

A.2 Dataset Statistics

We report dataset statistics for SuperGLUE and Natural-Instructions in Table 12 and Table 13, respectively.

A.3 Training Details

We train our models in PyTorch (Paszke et al., 2017) using FAIRSEQ (Ott et al., 2019).

A.4 More Details for Natural-Instructions

Dataset Sources. CosmosQA (Huang et al., 2019), DROP (Dua et al., 2019), EssentialTerms (Khashabi et al., 2017), MCTACO (Zhou et al., 2019), MultiRC (Khashabi et al., 2018), QASC (Khot et al., 2020), Quoref (Dasigi et al., 2019), ROPES (Lee et al., 2021) and Winogrande (Sakaguchi et al., 2020).

Training Datasets. We used the following 8 datasets when training models in the cross-task setting: subtask026_drop_question_generation, subtask060_ropes_question_generation, subtask028_drop_answer_generation, subtask047_misc_answering_science_questions, subtask061_ropes_answer_generation, subtask059_ropes_story_generation, subtask027_drop_answer_type_generation, subtask046_miscellaenous_question_typing.

Dataset	Task Category	Metrics	#Train	#Test	#Class
BoolQ	Question Answering	Accuracy	9427	3270	2
MultiRC	Question Answering	$F1_a$/EM	5100	953	2
COPA	Question Answering	Accuracy	400	100	2
RTE	Natural Language Inference	Accuracy	2500	278	2
CB	Natural Language Inference	Accuracy/F1	250	57	3

Table 12: Dataset statistics for SuperGLUE. We use the official development sets as test sets.

Dataset	Task Category	#Train	#Test
subtask003_mctaco_question_generation_event_duration	Question Generation	330	100
subtask040_qasc_question_generation	Question Generation	6400	100
subtask002_quoref_answer_generation	Answer Generation	6400	100
subtask033_winogrande_answer_generation	Answer Generation	6400	100
subtask034_winogrande_question_modification_object	Minimal Modification	6400	100
subtask045_miscellaneous_sentence_paraphrasing	Minimal Modification	93	100
subtask039_qasc_find_overlapping_words	Verification	6400	100
subtask044_essential_terms_identifying_essential_words	Verification	2138	100

Table 13: Dataset statistics for Natural-Instructions.

A.5 List of Function Words for the Last Phrase Prediction Task

We used the following function words for identifying the last phrase: the, a, an, for, including, and, in, is, are, were, was, neither, or, nor, be, at, in, on, by, to, would, will, before, after, of, about, from, excluding, except, during, under, above, then, into, onto, should, shall, must, may, might, than, with, using, can, could, about, as, from, within, without, have, had, been.

A.6 Templates for SuperGLUE

We show the SuperGLUE templates in Table 14.

A.7 Hyperparameters

We tune the hyperparameters based on development set performances. We tune the learning rate in {1e-7, 5e-7, 1e-6, 3e-6, 5e-6, 8e-6, 1e-5, 3e-5, 5e-5}, and the attention dropout rate in {0.0, 0.1}.

A.8 Effect of Amount of Data

In Figure 5, we report model performances for the 125M and 1.3B models on SuperGLUE and Natural-Instructions with 1%, 10%, 20%, 50%, and 200% of training examples.

A.9 Effect of Individual Self-Supervised Task to Downstream Tasks

We investigate the effect of individual self-supervised task by considering two experiment settings: training models with only one task and training models with one task excluded. We report the experiment results in Table 15, Table 16, Table 17, and Table 18. Our findings are:

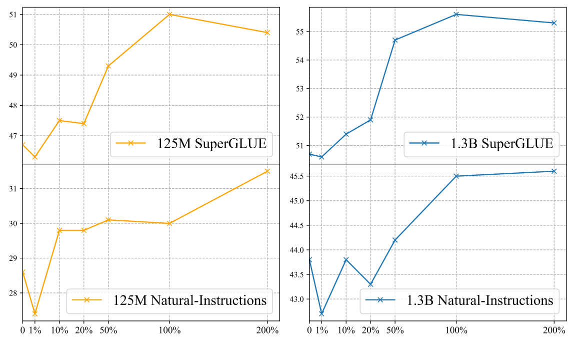

Figure 5: Average results on SuperGLUE and Natural-Instructions when varying number of examples used for training.

1. Combining all the four self-supervised tasks gives the largest improvements for most tasks, suggesting that these tasks are mostly complementary.

2. Each self-supervised task improves a few downstream task performances (e.g., NSG helps COPA; CL helps MultiRC and CB). This is likely due to the semantic similarities between tasks.

3. It is worth noting that (1) while CL hurts model performances on Natural-Instructions, it helps on the SuperGLUE; and (2) excluding NSG hurts the model performances most on Natural-Instructions whereas excluding CL hurts the most on SuperGLUE. This presumably is because SuperGLUE is ranking based and therefore is more favorable to classification-related training, whereas the tasks in Natural-Instructions are generation tasks and thus benefits more from generation-related tasks.

GPT3	${Context}⟨newline⟩ question: ${Question}⟨newline⟩answer:**${Answer}**
Ours	Input: ${Context} question: ${Question} answer: True⟨newline⟩Output: **${Answer}**

(a) BoolQ Template.

GPT3	${Context}⟨newline⟩ question: ${Question} True or False?⟨newline⟩answer:**${Answer}**
Ours	Input: ${Context} question: ${Question} answer: True⟨newline⟩Output: **${Answer}**

(b) RTE Template.

GPT3	${Context}⟨newline⟩**${Answer}**
Ours	Input: ${Context}⟨newline⟩Output:**${Answer}**

(c) COPA Template.

GPT3	${Context}⟨newline⟩ question: ${Question} true, false, or neither?⟨newline⟩answer:**${Answer}**
Ours	Input: ${Context} question: ${Question} true, false, or neither?⟨newline⟩Output: **${Answer}**

(d) CB Template.

Table 14: Evaluation templates for SuperGLUE. ${·} represents values drawn from a particular data field. We alter the GPT3 templates for these tasks to share similar formats with one of our self-supervised tasks. The red, boldfaced texts are used to compute the language modeling perplexities for ranking the labels. We note that the shown templates are for a single example, and there could be multiple examples within an instance.

Model	**BQ**	**MC**	**CA**	**RE**	**CB**	**Avg.**
LM	52.2	26.6	63.0	50.8	38.3	46.2
SelfSup.	55.7	33.6	67.6	53.0	44.9	51.0
NSG	52.1	25.9	64.0	51.0	41.2	46.9
CL	52.5	26.8	61.4	50.9	48.1	47.9
MWP	51.9	26.3	61.8	50.8	36.1	45.4
LPP	53.5	29.5	61.6	52.0	40.3	47.4

Table 15: SuperGLUE results when training the 125M model with one of the self-supervised tasks.

Model	**QG**	**AG**	**MM**	**VF**	**Avg.**
LM	33.7	12.9	53.0	14.7	28.6
SelfSup.	16.9	14.6	70.1	18.9	30.0
NSG	32.3	12.5	54.0	13.8	28.2
CL	8.3	0.3	1.0	2.7	3.1
MWP	15.2	19.4	50.5	17.8	25.7
LPP	11.3	16.6	49.5	19.9	24.3

Table 16: Natural-Instructions results when training the 125M model with one of the self-supervised tasks.

4. It is interesting to see that among the four self-supervised tasks, NSG and CL tasks are the two most important factors in terms of affecting the downstream performances. This is likely due to (1) the generic task formulation of NSG and it being the only sentence generation tasks; and (2) the drastic differences between CL and the other self-supervised tasks with respect to their inference styles. Unlike NSG/MWP/LPP, which models can rely on input within each example to solve the task, CL require models to make comparisons across examples in a training instance.

Model	**BQ**	**MC**	**CA**	**RE**	**CB**	**Avg.**
LM	52.2	26.6	63.0	50.8	38.3	46.2
ALL	55.7	33.6	67.6	53.0	44.9	51.0
ALL-NSG	55.0	31.8	62.7	52.5	45.9	49.6
ALL-MWP	55.6	33.5	67.3	52.7	45.5	50.9
ALL-LPP	53.5	30.5	67.6	51.9	46.6	50.0
ALL-CL	54.0	32.9	67.4	52.8	39.0	49.2

Table 17: SuperGLUE results when excluding one of the self-supervised tasks. The results are based on the 125M model.

Model	**QG**	**AG**	**MM**	**VF**	**Avg.**
LM	33.7	12.9	53.0	14.7	28.6
ALL	16.9	14.6	70.1	18.9	30.0
ALL-NSG	10.5	18.7	46.5	14.9	22.7
ALL-MWP	17.2	14.9	67.1	17.9	29.3
ALL-LPP	17.3	14.8	67.6	18.1	29.5
ALL-CL	23.1	15.1	59.0	18.2	28.9

Table 18: Natural-Instructions results when excluding one of the self-supervised tasks. The results are based on the 125M model.