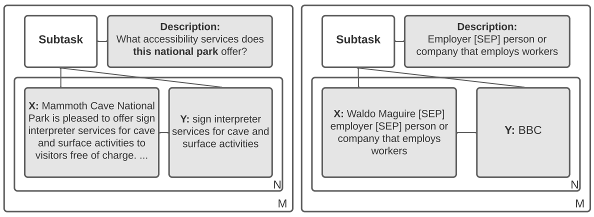
Zero-shot Learning by Generating Task-specific Adapters

Qinyuan Ye **Xiang Ren**
University of Southern California
{qinyuany,xiangren}@usc.edu

Abstract

Pre-trained text-to-text transformers achieve impressive performance across a wide range of NLP tasks, and they naturally support zero-shot learning (ZSL) by using the task description as prompt in the input. However, this approach has potential limitations, as it learns from input-output pairs at *instance* level, instead of learning to solve tasks at *task* level. Alternatively, applying existing ZSL methods to text-to-text transformers is non-trivial due to their text generation objective and huge size. To address these issues, we introduce HYPTER, a framework that improves zero-shot transferability by training a hypernetwork to generate task-specific adapters from task descriptions. This formulation enables learning at task level, and greatly reduces the number of parameters by using light-weight adapters. Experiments on two datasets demonstrate HYPTER improves upon fine-tuning baselines.



(a) Zero-shot Learning from Task Description, ZEST dataset (Weller et al., 2020)

(b) Zero-shot Relation Extraction, ZSRE dataset (Levy et al., 2017)

Figure 1: **In-task Zero-shot Learning.** The train set contains M sub-tasks, and the i-th sub-task contains N_i examples of (x, y) pairs in text format. During test time, the learned model is required to perform zero-shot inference on a new sub-task with task description.

1 Introduction

Recently there is a surge of interest in developing pre-trained text-to-text models (Raffel et al., 2020; Brown et al., 2020), which provide a unified formulation and off-the-shelf weights for a variety of NLP tasks, such as question answering (Khashabi et al., 2020) and commonsense reasoning (Bosselut et al., 2019). In addition to achieving strong performance, these text-to-text models naturally support zero-shot learning (ZSL) by prompting the task (description) in the model input (Brown et al., 2020; Raffel et al., 2020).

However, several disadvantages exist for this prompt-based approach: (1) predictions are sensitive to the prompt that are heuristically designed (Jiang et al., 2020); (2) this approach still learns from individual input-output examples, instead of learning to solve tasks (see Fig. 1). Alternatively, one may consider applying existing ZSL methods to text-to-text transformers to improve zero-shot generalization; however, such adaptation is non-trivial. Given the large size of text-to-text transformers, generating a full model from the task description (Jin et al., 2020) is infeasible; classical ZSL methods such as prototypical networks (Snell et al., 2017) does not extend beyond classification problems.

In this paper, we aim to improve zero-shot learning ability of text-to-text transformer models by better incorporating task descriptions. Our study focuses on the problem of *in-task* zero-shot learning, (*e.g.*, inference on an unseen relation for slot filling; see Fig. 1), as opposed to the more challenging task of *cross-task* zero-shot learning (*e.g.*, train a textual entailment model and test on question answering, Yin et al. 2020).

We introduce HYPTER, a framework towards our goal by using a hypernetwork to generate task-specific adapters from task descriptions. The generated parameters will inform the original model of task information and adapt it towards the described task. To reduce output dimensionality and avoid generating a full model (Jin et al., 2020), we generate light-weight, ad-hoc adapters (Houlsby et al., 2019), which are small layers that can be inserted into transformer layers for parameter-efficient transfer. Such formulation also effectively enables learning at task level, by learning to generate appropriate parameters for a task, instead of learning at instance level, by learning to generate the correct output for one specific input sequence. Consequently this will enable the learned hypernetwork to generalize at task level, supporting better zero-shot transferbility to unseen tasks.

We apply HYPTER to two zero-shot learning datasets: ZEST (Weller et al., 2020) and ZSRE (Levy et al., 2017). Additional training with HYPTER achieves 0.45% absolute improvement (11.3% comparative improvement) in Competence@90 metric on ZEST, when BART-Large is used as the main network.

2 Problem Definition

We refer to different NLP tasks (*e.g.*, classification/slot filling/question answering) as "tasks", and refer to each class/relation/question within a task as a "sub-task". In this work we focus on developing zero-shot generalization ability to new sub-tasks within a task, with only task description available for each sub-task. Therefore, we also refer to this problem as *in-task* zero-shot learning, as opposed to the more challenging problem of zero-shot transfer between different NLP tasks (Yin et al., 2020), or *cross-task* zero-shot learning.

Formally, a sub-task is denoted as a tuple of $(d, \mathcal{D})$, where d stands for a natural language description of the sub-task, and $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ contains examples of this specific sub-task. See Fig. 1 for an illustration. In our text-to-text formulation, both x_i and y_i are text. At train time, both d and $\mathcal{D}$ are available, while at test time, an unseen description d is given, and the model is expected to predict the correct y given input x without further training.

Several existing datasets fall into our in-task zero-shot setting. Zero-shot text classification (Yin et al., 2019) assumes unseen classes at test time, but the test cases will not go beyond the scope of text classification. Similarly, zero-shot slot filling (Levy et al., 2017) assumes unseen relations at test time. ZEST dataset (Weller et al., 2020) assumes new sub-tasks at test time, whereby the sub-task description is formatted as a generalized question.

3 Approach

Overview. Fig. 2 provides an overview of our HYPTER framework. HYPTER has two major parts:

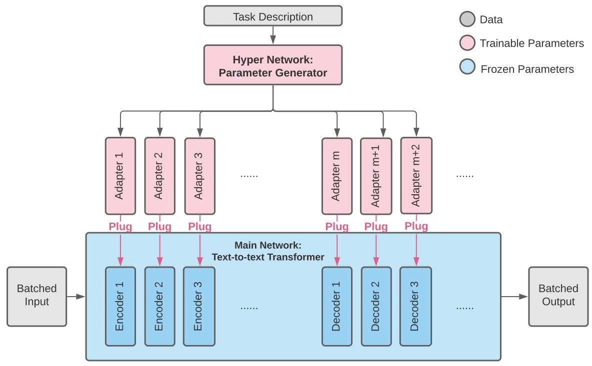


Figure 2: **Overview of HYPTER Framework.** We use a hypernetwork to generate adapters from task descriptions, and insert them to transformer layers in the main network. The resulting main network is ready to do zero-shot inference for the described sub-task.

(1) A main network, which is a pre-trained text-to-text model that is responsible for performing sub-tasks. We instantiate the main network with BART-Base/Large (Lewis et al., 2020). (2) A hypernetwork, which generates adapters to be plugged into the main network. For more background on adapters, see Fig. 3 and Appendix A.

Hypernetwork. The hypernetwork includes two parts: an encoder that encodes the task description d to latent representation $\mathbf{h}_0$, and multiple decoders that maps the representation to adapter parameters ϕ in parallel. The encoder of the hypernetwork is instantiated with a RoBERTa-base model (Liu et al., 2019). Given a task description d, the encoder maps it to a hidden representation $\mathbf{h}_0$.

$$\mathbf{h}_0 = \text{RoBERTa}(d) \tag{1}$$

We use one dedicated decoder for one transformer layer in the main network. Each decoder contains two linear layers; the final layer of decoder i will output the parameters ϕ_i that will be used in adapters for transformer layer i.

$$\begin{aligned} \mathbf{h}_{i,1} &= \text{ReLU}(\mathbf{W}_{i,1}\mathbf{h_0} + \mathbf{b}_{i,1}) \\ \phi_i &= \mathbf{W}_{i,2}\mathbf{h}_{i,1} + \mathbf{b}_{i,2} \end{aligned} \tag{2}$$

Here $\mathbf{W}_{i,1}, \mathbf{b}_{i,1}, \mathbf{W}_{i,2}, \mathbf{b}_{i,2}$ are trainable parameters. The generated parameters ϕ_i will be sliced and reshaped to become parameters $[\mathbf{W}_{id}, \mathbf{b}_{id}, \mathbf{W}_{iu}, \mathbf{b}_{iu}]$ used in the adapter (see Appendix A).

Model Training. We adopt a training schedule where we first train the main network, then train the hypernetwork while the main network is frozen.

Conceptually, the first stage ensures that the main network captures the general ability across different sub-tasks; the seconds stage allow the hypernetwork to learn to adapt the main network to a specific sub-task. During main network training, the text-to-text model is fine-tuned with all $(\text{Concat}(x, d), y)$ examples in the dataset. Here $\text{Concat}(x, d)$ means the concatenation of input x and task description d. The learned main network also serves as our prompt-based baseline (see Sec 4.1). During hypernetwork training, we sample a task $(d, \mathcal{D})$ from the train set and sample a mini-batch of (x, y) examples from $\mathcal{D}$. Given a description d, we first generate adapter parameters with the hypernetwork and insert them to the main network, then compute the cross entropy loss of generating y given input $\text{Concat}(x, d)$. The loss is end-to-end differentiable and is back-propagated to update the hypernetwork, while the main network is frozen.

Model Inference. At test time, when given an unseen task description d, we generate adapters and insert them to the main network, similar to the procedures during training. After this step, we immediately obtain a model that is ready to do zero-shot inference for the described sub-task.

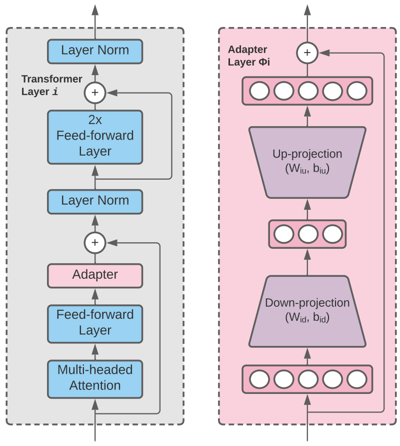


Figure 3: **Simplified Adapters Used in This Work.** The adapter layer for the i-th transformer layer includes two linear layers, $(\mathbf{W}_{id}, \mathbf{b}_{id})$ and $(\mathbf{W}_{iu}, \mathbf{b}_{iu})$, separated by a non-linearity activation. An adapter can be "plugged" into a transformer layer, by directly adding it after the multi-headed attention and feed-forward layer. This figure is adapted from (Houlsby et al., 2019).

ZEST **Input:** zest question: What accessibility services does this national park offer? zest context: Mammoth Cave National Park is pleased to offer sign interpreter services for cave and surface activities to visitors free of charge. Arrangements can be made by contacting the park at the number located below. ... **Output:** sign interpreter services for cave and surface activities
ZSRE (without description) **Input:** Waldo Maguire [SEP] employer **Output:** BBC
ZSRE (with description) **Input:** Waldo Maguire [SEP] employer [SEP] person or company that employs workers **Output:** BBC

Table 1: **Examples of the input and targets.** Task descriptions are included as part of the input as "prompt".

4 Experiments

4.1 Experiment Setup

Datasets. We use two datasets that fit our in-task zero-shot setup. **(1)** Zero-shot learning from Task Descriptions (ZEST, Weller et al. 2020)[1], which formulates task descriptions as generalized questions, and provides multiple input-output examples for each question. The performance is evaluated with a novel metric: "Competence@K", along with standard mean F1 score. For example, Competence@90=5 suggests 5% of all sub-tasks can be solved with mean F1 better than 90% at test time. There are 538/114/599 sub-tasks in train/dev/test set. **(2)** Zero-shot Relation Extraction (ZSRE, Levy et al. 2017) is a slot filling task that aims to predict the object given the subject and a relation. We use the processed dataset provided in the KILT benchmark (Petroni et al., 2020)[2] and adopt their implicit knowledge setting (*i.e.*, having no access to external knowledge bases). Following KILT, we report accuracy for this task. There are 84/12/24 non-overlapping relations in train/dev/test set. Additionally, since all relations in ZSRE come from Wikidata (Vrandecic, 2012), we gather relation descriptions from wikidata.org as task description.

Baselines. For both datasets, we include the baseline of direct fine-tuning with task description as prompt. For ZSRE, we include an additional baseline that does not use full descriptions but only use the relation name as description (*e.g.*, "employer"). See Table 1 for detailed examples.

Training Details. We discuss the training details, including hyperparameters, in Appendix B.

4.2 Results

We present the results for ZEST in Table 2 and results for ZSRE in Table 3. On ZEST dataset, we observe that the Competence@90 metric is

[1]ZEST leaderboard: https://leaderboard.allenai.org/zest/submissions/public

[2]KILT leaderboard containing ZSRE dataset: https://eval.ai/web/challenges/challenge-page/689/overview

improved from 3.98 to 4.43 when using BART-Large, yielding an 11.3% relative improvement. On BART-Base model, we observe an improvement from 2.23 to 2.53. On ZSRE dataset, we also observe that HYPTER brings improvement to the fine-tuning baseline. In addition, we observe that: (1) By comparing the two categories of with/without description, we conclude that including informative description of relations in the prompt improves performance. (2) The improvement brought by HYPTER is less significant, or sometimes marginal, on ZSRE dataset. We suspect this is due to ZSRE dataset's small size (84 sub-tasks in train set compared to 553 in ZEST dataset); this creates additional challenges for training the hypernetwork.

	ZEST-Dev			ZEST-Test		
Model	Mean	C@75	C@90	Mean	C@75	C@90
Bart-Base	29.72	7.87	**4.05**	31.97	**7.03**	2.23
+ HYPTER	**29.81**	**8.67**	**4.05**	**32.32**	6.72	**2.53**
Bart-Large (reported)	40†	13	8	37.93	11.19	3.96
Bart-Large	42.10	16.72	8.85	40.13	10.91	3.98
+ HYPTER	**43.50**	**17.46**	**9.64**	**40.41**	**11.35**	**4.43**

Table 2: **Performance on ZEST dataset.** "Mean" refers to mean F1 score, "C@75" refers to Competence@75. Additional training with HYPTER improves fine-tuning results. †The original paper (Weller et al., 2020) reported rounded numbers.

	ZSRE-Dev	ZSRE-Test
Without Description.		
Bart-Base	1.83	9.10
+ HYPTER	1.96	9.16
Bart-Large (reported)	3.03	9.14
Bart-Large	2.95	9.93
+ HYPTER	2.98	9.95
With Description.		
Bart-Base	2.34	9.57
+ HYPTER	2.60	9.81
Bart-Large	3.33	11.44
+ HYPTER	**3.57**	**11.48**

Table 3: **Performance on ZSRE dataset.** "Bart-Large (reported)" results are from (Petroni et al., 2020).

5 Related Work

Zero-shot Learning with Transformers. Zero-shot setting has been explored in various NLP tasks, including text classification (Yin et al., 2019), entity linking (Logeswaran et al., 2019) and entity typing (Obeidat et al., 2019). Several works study cross-task transfer by unifying the input-output format, *e.g.*, relation extraction as machine reading comprehension (Levy et al., 2017), named entity recognition as machine reading comprehension (Li et al., 2020), coreference resolution as textual entailment (Yin et al., 2020). Such formulation also allows generalization to unseen relation or named entity types at test time. A separate line of work studies zero-shot learning in cross-lingual settings, with unseen languages at test time (Nooralahzadeh et al., 2020).

Adapters for Transformers. Houlsby et al. (2019) proposed adapter layers for parameter-efficient transfer learning in NLP. Adapter layers, which adopt a bottleneck architecture with two linear layers, are added after each multi-headed attention layer and each feed-foward layer in a pre-trained transformer. Adapters has been recently applied to multi-lingual settings, with successes in NER, QA and commonsense reasoning (Pfeiffer et al., 2020; Philip et al., 2020; Artetxe et al., 2020).

Hypernetworks and Contextual Parameter Generators. Hypernetwork (Ha et al., 2017) is a broad concept of "using one network to generate the weights for another network". This concept is applied to model architecture search (Brock et al., 2018), visual reasoning (Perez et al., 2018) and zero-shot image classification (Jin et al., 2020). Notably, Platanios et al. (2018) introduced contextual parameter generator (CPG), which extends the idea of hypernetworks by conditioning on "well-defined context based on the input data". Closely related to our work, UDapter (Üstün et al., 2020) studies multilingual dependency parsing by generating adapter parameters using CPG. Our work is more generalizable as we do not restrict the format of task (dependency parsing v.s. general text-to-text tasks) or relations between sub-tasks (cross-lingual tasks v.s. tasks with text-form descriptions).

6 Conclusion

In this paper, we introduced HYPTER, a framework for improved zero-shot learning from task descriptions. HYPTER enhances task-specific abilities by inserting adapters generated with a hypernetwork, meanwhile it maintains the model's general task-solving ability by freezing main model parameters. We demontrated the effectiveness of HYPTER on two datasets. In the future, we will include more empirical analysis of the proposed approach, including joint training of the main network and the hypernetwork; ablation study of our design choices. We will also explore the more challenging problem of cross-task zero-shot learning.

Acknowledgments

This work is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. 2018. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. *ArXiv*, abs/1609.09106.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Tian Jin, Zhun Liu, Shengjia Yan, Alexandre Eichenberger, and Louis-Philippe Morency. 2020. Language to network: Conditional parameter adaptation with natural language descriptions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6994–7007, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

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.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

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*.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

Farhad Nooralahzadeh, Giannis Bekoulis, Johannes Bjerva, and Isabelle Augenstein. 2020. Zero-shot cross-lingual transfer with meta learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4547–4562, Online. Association for Computational Linguistics.

Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi, and Prasad Tadepalli. 2019. Description-based zero-shot fine-grained entity typing. 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 807–814, Minneapolis, Minnesota. Association for Computational Linguistics.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *AAAI*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vassilis Plachouras, Tim Rocktäschel, et al. 2020. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435, Brussels, Belgium. Association for Computational Linguistics.

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.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4077–4087. Curran Associates, Inc.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Denny Vrandecic. 2012. Wikidata: a new platform for collaborative data collection. *Proceedings of the 21st International Conference on World Wide Web*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. 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.

A Adapters

Our work is built on adapters (Houlsby et al., 2019), light-weight modules that can be placed into transformer layers for parameter-efficient transfer learning. During learning, the main model is frozen, while only layer norm and adapter parameters are learnable. In this paper, we adopt a simplified design compared to the original paper (see Fig. 3) — In each transformer, exactly one adapter module will be added after the multi-headed attention. One adapter module contains two linear layers separated by an non-linearity activation layer. We denote W_{id}, b_{id} as the parameter for down-projection for the adapter in layer i, and W_{iu}, b_{iu} for the up-projection.

B Training Details

We use transformers (Wolf et al., 2020) for all our experiments. For hypernetwork training, we train up to 100 epochs (one epoch here refers an iteration over all sub-tasks). We update the hypernetwork every b tasks, and we call b as task batch size. When learning from one sub-task, we sample b' examples within this task, and we call b' as the example batch size. We select b from $\{4,8,16,32\}$, b' from $\{4,8,16,32\}$, adapter width d from $\{4,8,16,32\}$, learning rate α from $\{$3e-6, 1e-5, 3e-5$\}$, based on dev set performance.