Single-dataset Experts for Multi-dataset Question Answering

Dan Friedman **Ben Dodge** **Danqi Chen**
Department of Computer Science, Princeton, NJ
{dfriedman,bd4,danqic}@cs.princeton.edu

Abstract

Many datasets have been created for training reading comprehension models, and a natural question is whether we can combine them to build models that (1) perform better on all of the training datasets and (2) generalize and transfer better to new datasets. Prior work has addressed this goal by training one network simultaneously on multiple datasets, which works well on average but is prone to over- or under-fitting different sub-distributions and might transfer worse compared to source models with more overlap with the target dataset. Our approach is to model multi-dataset question answering with an ensemble of single-dataset experts, by training a collection of lightweight, dataset-specific adapter modules (Houlsby et al., 2019) that share an underlying Transformer model. We find that these **M**ulti-**A**dapter **D**ataset **E**xperts (MADE) outperform all our baselines in terms of in-distribution accuracy, and simple methods based on parameter-averaging lead to better zero-shot generalization and few-shot transfer performance, offering a strong and versatile starting point for building new reading comprehension systems.[1]

1 Introduction

The goal of reading comprehension is to create computer programs that can answer questions based on a single passage of text. Many reading comprehension datasets have been introduced over the years, and prior work has explored ways of training one network on multiple datasets to get a model that generalizes better to new distributions (Talmor and Berant, 2019; Fisch et al., 2019; Khashabi et al., 2020). Our goal is to build a multi-dataset model that performs well on the training distributions and can also serve as a strong starting point for transfer learning to new datasets. Multi-dataset training provides a way to model the regularities between datasets but it has the following shortcomings. First, multi-task models are liable to over- or under-fit different tasks (Gottumukkala et al., 2020), which can result in worse in-distribution accuracy. Second, given a particular target dataset, multi-dataset models might achieve worse transfer performance compared to a specialized model trained on a more similar source dataset.

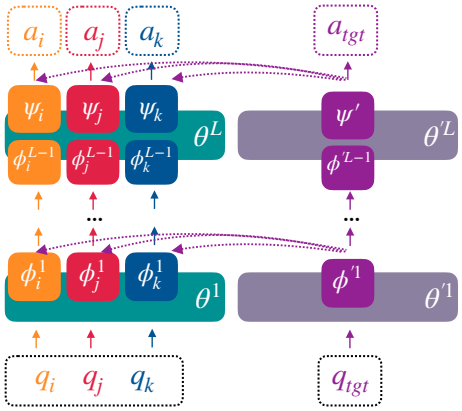

Figure 1: MADE consists of a set of dataset-specific adapters and classifiers $\{(\phi_t, \psi_t)\}_{t=1}^{|\mathcal{S}|}$ with a shared Transformer model θ. They are optimized jointly on a set of training datasets (left). To transfer to a new dataset (right), we either average the parameters of the adapters, or fine-tune the ensemble on the target dataset and take the weighted average at the end (Section 3).

Our idea is to combine the benefits of single- and multi-dataset by training a collection of single-dataset experts that share an underlying Transformer model (Figure 1). This system is based on adapters (Houlsby et al., 2019), lightweight task-specific modules interleaved between the layers of a pre-trained Transformer (e.g., BERT; Devlin et al., 2019). The standard use of adapters is as a parameter-efficient alternative to fine-tuning: task-specific adapters are trained separately on top of a frozen Transformer, which means the

[1]Our code and models are available at https://github.com/princeton-nlp/MADE.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6128–6137
November 7–11, 2021.

adapters cannot directly learn cross-task regularities. We instead first train a shared Transformer in a multi-adapter setup before refining adapters for individual datasets, which we call **M**ulti-**A**dapter **D**ataset **E**xperts (MADE). Our intuition is that the shared parameters encode regularities across different reading comprehension tasks while the adapters model the sub-distributions, resulting in more accurate and robust specialized models that transfer better to a variety of target datasets.

We apply this approach to a range of extractive question answering datasets from the MRQA 2019 shared task (Fisch et al., 2019), training MADE on six in-domain datasets and evaluating generalization and few-shot transfer learning to six out-of-domain datasets. The resulting system outperforms single- and multi-dataset models in terms of in-domain accuracy, and we find that a simple approach to transfer learning works well: averaging the parameters of the MADE adapters results in a single model that gets better zero-shot generalization and few-shot transfer performance compared to both baselines as well as a state-of-the-art multi-dataset QA model, UnifiedQA (Khashabi et al., 2020). Our experiments illustrate the benefits of modeling both cross-dataset regularities and dataset-specific attributes, and the trained models offer a strong and versatile starting point for new question-answering models.

2 Related Work

Prior work has addressed multi-dataset reading comprehension by fine-tuning a pre-trained Transformer language model (Devlin et al., 2019) simultaneously on examples from multiple datasets (Talmor and Berant, 2019; Fisch et al., 2019). Several works explore different multi-task sampling schedules, as a way of mitigating training set imbalances (Xu et al., 2019; Gottumukkala et al., 2020). Another line of work focuses on training models to answer a wider variety of question types, including UnifiedQA (Khashabi et al., 2020), a T5 model (Raffel et al., 2020) trained on datasets with different answer formats, such as yes/no and multiple-choice, using a unified text-to-text format.

Adapters (Houlsby et al., 2019; Rebuffi et al., 2018) are task-specific modules interleaved between the layers of a shared Transformer. Stickland and Murray (2019) trained task adapters and the Transformer parameters jointly for the GLUE benchmark (Wang et al., 2019) but achieved mixed results, improving on small datasets but degrading on larger ones. Subsequent work has used a frozen, pre-trained Transformer and trained task adapters separately. Researchers have explored different methods for achieving transfer learning in this setting, such as learning to interpolate the activations of pre-trained adapters (Pfeiffer et al., 2021).

3 Method

3.1 Problem Definition

The objective of reading comprehension is to model the distribution $p(a \mid q, c)$, where $q, c, a \in \mathcal{V}^*$ represent a *question*, supporting *context*, and *answer* respectively and consist of sequences of tokens from a vocabulary $\mathcal{V}$. For simplicity, we focus on extractive reading comprehension, where every question can be answered by selecting a span of tokens in the context, but the approach is generic and can be extended to other formats. We make the standard assumption that the probability of context span $c_{i...j}$ being the answer can be decomposed into the product of $p(\text{start} = i \mid q, c)$ and $p(\text{end} = j \mid q, c)$.

We consider a collection of source datasets $\mathcal{S}$ and target datasets $\mathcal{T}$, where each dataset $\mathcal{D} \in \mathcal{S} \cup \mathcal{T}$ consists of supervised examples in the form (q, c, a). The goal is to train a model on $\mathcal{S}$ that achieves high in-domain accuracy and transfers well to unseen datasets in $\mathcal{T}$, either zero-shot or given a small number of labeled examples.

3.2 Multi-dataset Fine-tuning

The standard approach to multi-dataset reading comprehension is to fit a single model to examples drawn uniformly from the datasets in $\mathcal{S}$:

$$\arg\min_{\theta,\psi} \mathbb{E}_{\mathcal{D}_i \sim \mathcal{S}} \left[\mathbb{E}_{q,c,a \sim \mathcal{D}_i} [-\log p_{\theta,\psi}(a \mid q, c)] \right],$$

where θ refers to the parameters of an encoder model (usually a pre-trained Transformer like BERT; Devlin et al., 2019), which maps a question and context to a sequence of contextualized token embeddings, and ψ denotes the classifier weights used to predict the start and end tokens.

The objective is approximated by training on mixed mini-batches with approximately equal numbers of examples from each dataset (Fisch et al., 2019; Khashabi et al., 2020), although some researchers have investigated more sophisticated sampling strategies (Xu et al., 2019). For example, Gottumukkala et al. (2020) introduce *dynamic sampling*, sampling from each dataset in inverse proportion to the model's current validation accuracy.

Model	SQuAD 1.1	HotpotQA	TriviaQA	NewsQA	SearchQA	NaturalQ	Avg.
Dynamic sampling	91.4	80.9	**80.5**	71.4	84.6	79.8	81.4
Single-dataset fine-tuning	90.9	78.6	79.3	70.3	84.5	79.2	80.5
Single-dataset adapters	91.4	78.5	79.6	70.9	85.1	79.2	80.8
Multi-dataset fine-tuning	91.8	81.0	80.1	**72.3**	84.7	79.5	81.6
MADE (w/o adapter tuning)	91.9	80.7	80.1	71.8	84.5	79.5	81.4
MADE (w/ adapter tuning)	**92.4**	**81.5**	**80.5**	72.1	**85.8**	**80.9**	**82.2**

Table 1: In-domain F1. *Dynamic sampling* is a reimplementation of the method from Gottumukkala et al. (2020) on the MRQA datasets. We compare MADE results at the end of joint optimization (*w/o adapter tuning*), and after freezing the Transformer and tuning the adapters separately (*w/ adapter tuning*). See Section 4.2 for details.

3.3 MADE

Our approach is to explicitly model the fact that our data represent a mixture of datasets. We decompose the model parameters into a shared Transformer, θ, and dataset-specific token classifiers $\boldsymbol{\psi} = \psi_1, \ldots, \psi_{|\mathcal{S}|}$ and adapters $\boldsymbol{\phi} = \phi_1, \ldots, \phi_{|\mathcal{S}|}$ (Figure 1). We use a two-phase optimization procedure to fit these parameters. In the *joint optimization* phase, we jointly train all of the parameters on the source datasets:

$$\arg\min_{\theta,\boldsymbol{\phi},\boldsymbol{\psi}} \mathbb{E}_{\mathcal{D}_i \sim \mathcal{S}} \left[\mathbb{E}_{q,c,a \sim \mathcal{D}_i} [-\log p_{\theta,\phi_i,\psi_i}(a \mid q, c)] \right]$$

After validation accuracy (average F1 scores of the source datasets) stops improving, we freeze θ and continue *adapter tuning*, refining each pair of (ϕ_i, ψ_i) separately on each dataset.

Zero-shot generalization We use a simple strategy to extend MADE to an unseen dataset: we initialize a new adapter and classifier (ϕ', ψ') by averaging the parameters of the pre-trained adapters and classifiers $\phi_1, \ldots, \phi_{|\mathcal{S}|}$ and $\psi_1, \ldots, \psi_{|\mathcal{S}|}$, and return the answer with the highest probability under $p_{\theta,\phi',\psi'}(a \mid q, c)$. Alternatively, we also consider an ensemble approach, averaging the token-level probabilities predicted by each adapter (this is less appealing as we need to run the model $|\mathcal{S}|$ times).

Transfer learning We also consider a transfer learning setting, in which a small number of labeled examples of a target domain (denoted $\mathcal{D}_{\text{tgt}}$) are provided. We explore two ways to build a single, more accurate model. The first is to initialize (ϕ', ψ') as a weighted average of pre-trained adapters, $\phi' = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \alpha_i \phi_i$, and classifiers $\psi' = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \alpha_i \psi_i$, using $\mathcal{D}_{\text{tgt}}$ to estimate the mixture weights. For each i, we set the mixture weight α_i to be proportional to the exponential of the negative zero-shot loss on the training data:

$$\alpha_i \propto \exp\left(\mathbb{E}_{q,c,a \in \mathcal{D}_{\text{tgt}}} [\log p_{\theta,\phi_i,\psi_i}(a \mid q, c)]\right),$$

and then tune θ and (ϕ', ψ') on the target dataset. The second approach is to first jointly tune θ, $\boldsymbol{\phi}$, and $\boldsymbol{\psi}$ on $\mathcal{D}_{\text{tgt}}$, maximizing the marginal likelihood:

$$\mathbb{E}_{q,c,a \sim \mathcal{D}_{\text{tgt}}} \left[\log \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} p_{\theta,\phi_i,\psi_i}(a \mid q, c) \right],$$

and then take the weighted average of the parameters, calculating the mixture weights α_i as above but using the loss of the fine-tuned adapters on a small number of held-out examples from $\mathcal{D}_{\text{tgt}}$. After training, both approaches result in a single model that only requires running one forward pass through (θ, ϕ', ψ') to make a prediction.

4 Experiments

4.1 Setup

We use the datasets from the MRQA 2019 shared task (Fisch et al., 2019), which are split into six large in-domain datasets,[2] and six small out-of-domain datasets.[3] Dataset statistics are in Appendix A.1. We use the RoBERTa-base model (Liu et al., 2019) with the default adapter configuration from Houlsby et al. (2019), which adds approximately 1.8M parameters to the ~128M in RoBERTa-base (1%).

[2] SQuAD 1.1 (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), NewsQA (Trischler et al., 2017), SearchQA (Dunn et al., 2017), and Natural Questions (Kwiatkowski et al., 2019)

[3] BioASQ (Tsatsaronis et al., 2015), DROP (Dua et al., 2019), DuoRC (Saha et al., 2018), RACE (Lai et al., 2017), RelationExtraction (Levy et al., 2017), and TextbookQA (Kembhavi et al., 2017).

Model	BioASQ	DROP	DuoRC	RACE	RelEx	TextbookQA	Avg.
UnifiedQA-base	59.7	45.7	30.4	51.4†	82.0	35.9	50.9
Multi-dataset fine-tuning	64.1	**51.5**	63.0	47.6	**87.3**	**59.0**	62.1
MADE (w/o adapter-tuning)	**66.5**	50.9	**67.2**	**47.8**	86.7	58.5	**62.9**
MADE (w/ adapter-tuning)	64.9	49.5	62.3	**47.8**	86.7	53.2	60.7
Single-dataset adapters (ensemble)	65.3	40.2	60.5	41.6	86.2	55.8	58.3
MADE (w/o adapter-tuning, ensemble)	65.2	52.0	**66.6**	47.7	86.4	58.8	62.8
MADE (w/ adapter-tuning, ensemble)	**66.3**	**52.5**	65.8	**48.3**	**87.5**	**59.9**	**63.4**

Table 2: Zero-shot generalization results (F1), either averaging parameters (middle) or ensembling predictions (bottom), and comparing MADE at the end of joint optimization (*w/o adapter tuning*), and after tuning the adapters separately (*w/ adapter tuning*). See Section 4.3. †: RACE is part of the UnifiedQA training data.

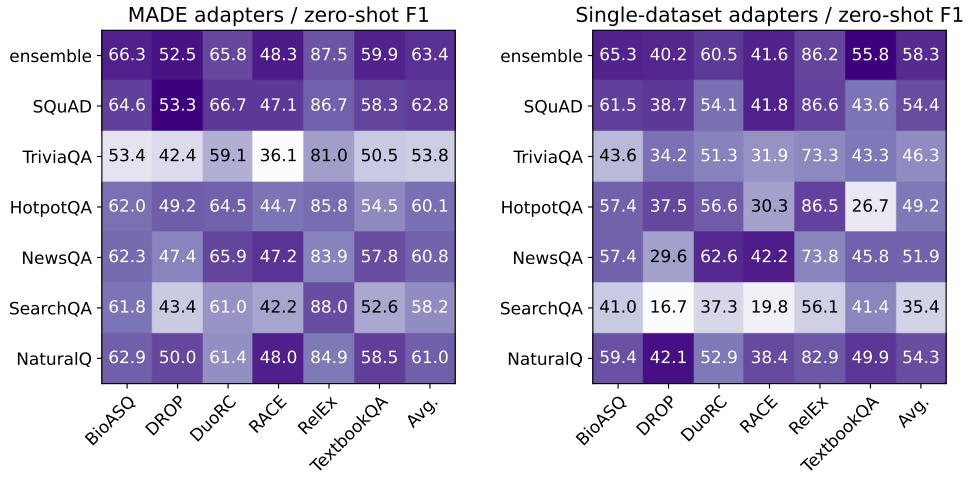

Figure 2: Out-of-domain accuracy (F1) of the different MADE adapters (left) compared with single-dataset adapters (right), which are trained trained separately with a frozen Transformer.

4.2 In-domain Performance

First we train MADE on the six training datasets and compare in-domain accuracy with single- and multi-dataset fine-tuning and standard adapter training (freezing the Transformer parameters). For context, we also compare with a method from recent work, *dynamic sampling* (Gottumukkala et al., 2020), by sampling from each dataset in proportion to the difference between the current validation accuracy (EM+F1) on that dataset and the best accuracy from single-dataset training. We train all models by sampling up to 75k training and 1k development examples from each dataset, following Fisch et al. (2019). More details are in Appendix A.2.

Table 1 shows that MADE scores higher than both single- and multi-dataset baselines. Both phases of MADE training—joint optimization followed by separate adapter tuning—are important for getting high accuracy. Jointly optimizing the underlying MADE Transformer improves performance compared to single-dataset adapters, suggesting that joint training encodes some useful cross-dataset information in the Transformer model. Adapter tuning is important because the multi-dataset model converges on different datasets at different times, making it hard to find a single checkpoint that maximizes performance on all datasets (see Appendix Figure 3 for the training curve). Some of the improvements can also be attributed to the adapter architecture itself, which slightly outperforms fine-tuning in most datasets. Dynamic sampling does not improve results, possibly because the datasets are already balanced in size.

4.3 Zero-shot Generalization

Table 2 shows the results of applying this model to an unseen dataset (zero-shot). We compare two methods for using MADE—averaging the parameters or ensembling the predictions—with the multi-dataset model from Section 4.2, an ensemble of single-dataset adapters, and the pre-trained

K	Model	BioASQ	DROP	DuoRC	RACE	RelEx	TextbookQA	Avg.
16	UnifiedQA-base	59.7	45.2	30.2	**52.9**	82.0	33.8	50.6
	Multi-dataset fine-tuning	64.7	52.2	63.2	47.5	87.3	57.9	62.1
	MADE (pre avg.)	**67.0**	**52.8**	65.7	47.5	**87.7**	**59.8**	63.4
	MADE (post avg.)	66.8	52.6	**67.2**	47.3	87.7	59.7	**63.5**
64	UnifiedQA-base	65.5	47.9	32.9	**54.3**	83.4	35.6	53.3
	Multi-dataset fine-tuning	66.1	54.9	64.3	48.3	87.8	**60.7**	63.7
	MADE (pre avg.)	69.4	**55.9**	66.8	47.6	88.4	60.2	64.7
	MADE (post avg.)	**70.9**	55.2	**66.9**	47.5	**88.5**	60.0	**64.8**
256	UnifiedQA-base	71.7	49.8	36.5	**55.2**	85.6	36.8	55.9
	Multi-dataset fine-tuning	73.9	**57.2**	65.2	49.1	89.1	**62.9**	66.2
	MADE (pre avg.)	73.9	56.9	65.3	48.7	88.9	61.5	65.9
	MADE (post avg.)	**75.1**	56.9	**66.9**	48.9	**90.2**	61.8	**66.6**

Table 3: Transfer learning to MRQA out-of-domain datasets with K training examples (F1, averaged over three random seeds). †: RACE is part of the UnifiedQA training data. pre avg.: average the adapter parameters at initialization. post avg.: fine-tune the adapters and jointly and average the parameters at the end. See Section 4.4.

UnifiedQA-base (Khashabi et al., 2020).[4] We compare MADE with and without the second phase of separate adapter-tuning.

Surprisingly, averaging the parameters of the different MADE adapters results in a decent predictor. Parameter averaging works best for MADE without adapter-tuning, possibly because the parameters diverge during adapter tuning and no longer interpolate well.[5] Ensembling the separately-tuned MADE adapters gives the best performance, at an additional computational cost.

Figure 2 compares the zero-shot accuracy obtained by the different MADE and single-dataset adapters. The two sets of adapters show similar patterns, with some adapters generalizing better than others, depending on the target, but all of the MADE adapters generalize better than the corresponding single-dataset adapters. This performance gap is considerably bigger than the gap in in-domain performance (Table 1), further illustrating the benefit of joint optimization.

4.4 Transfer Learning

Finally, we compare two ways of using MADE for transfer learning: either averaging the adapter parameters and then fine-tuning the resulting model (*pre avg.*), or first fine-tuning all of the adapters and then taking the weighted average (*post avg.*). In both cases, we also back-propagate through the Transformer parameters. We reserve 400 examples from each target dataset to use as a test set (following Ram et al., 2021) and sample training datasets of different sizes, using half of the examples for training and the other half as validation data for early stopping and to set the mixture weights for averaging the adapter parameters.

The results are in Table 3. On average, MADE leads to higher transfer accuracy compared to the baselines, especially for the smaller sizes of datasets, showing that an ensemble of robust single-dataset experts is a good starting point for transfer learning. The post-average method performs about the same as averaging at initialization in the lower-data settings, and better with $K = 256$. All models struggle to learn on two datasets, DuoRC and TextbookQA, which have long contexts and distant supervision, which might represent more challenging targets for few-shot learning. We also experimented with single-dataset adapters and with a frozen Transformer, which perform worse; detailed results are in Appendix B.2.

5 Conclusion

MADE combines the benefits of single- and multi-dataset training, resulting in better in-domain accuracy and transfer performance than either multi-dataset models or ensembles of single-dataset models, especially in low resource settings. For future work we plan to explore ensembling methods for better zero-shot prediction and interpolating MADE weights for better transfer learning.

[4] UnifiedQA was trained on different datasets with a different architecture, but represents an alternative off-the-shelf model for QA transfer learning. We compare to UnifiedQA-base because the encoder has approximately the same number of parameters as RoBERTa-base.

[5] We also tried applying this approach to single-dataset adapters but got an average F1 of 7%.

Acknowledgements

We thank the members of the Princeton NLP group and the anonymous reviewers for their valuable comments. This work is supported by a Graduate Fellowship at Princeton University.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186.

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 *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2368–2378.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A newQ&A dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *2nd Workshop on Machine Reading for Question Answering (MRQA)*.

Ananth Gottumukkala, Dheeru Dua, Sameer Singh, and Matt Gardner. 2020. Dynamic sampling strategies for multi-task reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 920–924.

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 *International Conference on Machine Learning*, pages 2790–2799.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*, pages 1601–1611.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*, pages 4999–5007.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single qa system. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics (TACL)*, 7:453–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Computational Natural Language Learning (CoNLL)*, pages 333–342.

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*.

Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. 2019. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In *2nd Workshop on Machine Reading for Question Answering (MRQA)*, pages 220–227.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 487–503.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–54.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text Transformer. *The Journal of Machine Learning Research (JMLR)*, 21(140).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Association for Computational Linguistics (ACL)*, pages 3066–3079.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8119–8127.

Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. 2018. DuoRC: Towards complex language understanding with paraphrased reading comprehension. In *Association for Computational Linguistics (ACL)*, pages 1683–1693.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning (ICML)*, pages 5986–5995.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Association for Computational Linguistics (ACL)*, pages 4911–4921.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *2nd Workshop on Representation Learning for NLP*, pages 191–200.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1–28.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 38–45.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2019. Multi-task learning with sample re-weighting for machine reading comprehension. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2644–2655.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380.

A Task Details

A.1 Dataset Details

We use the pre-processed datasets from the MRQA 2019 shared task (Fisch et al., 2019). Table 4 provides some dataset statistics.

A.2 Training Details

Our models are implemented in PyTorch (Paszke et al., 2019) using HuggingFace (Wolf et al., 2020) and the adapter-transformers library (Pfeiffer et al., 2020). For all in-domain experiments, we sample 75,000 training and 1,000 validation examples and train with a constant learning rate and a batch size of 8, taking checkpoints every 1024 steps and stopping if validation F1 fails to improve for 10 checkpoints up to a fixed maximum number of epochs (10 for single-dataset training and 3 epochs for multi-dataset training). We use a constant learning rate of 1e-5 for Transformer parameters and 1e-4 for adapter parameters, following standard settings for RoBERTa and adapters respectively (Liu et al., 2019; Houlsby et al., 2019), and use the AdamW optimizer (Loshchilov and Hutter, 2018) with the HuggingFace default parameters. For the multi-dataset models, we construct mini-batches of size B by repeating B times: pick a dataset uniformly, and pick an example uniformly from that dataset.

We train all models on single 2080Ti GPUs with 11GB of memory each. The multi-dataset models take around two days to train, the single-dataset models take less than 24 hours, and it takes about 2 hours to train one model sequentially on six transfer datasets for three values of K and three seeds.

Distant supervision Some datasets provide the gold answer string but do not mark the gold answer span in the context. We train the model to maximize the marginal likelihood of the gold answer string, marginalizing over all occurrences in the context. The set of possible answer spans are annotated in the pre-processed MRQA datasets.

Long contexts For inputs that are longer than the maximum input window for RoBERTa (512 tokens), we use a sliding window to split in the input into multiple "chunks": every input begins with the full question and the `[cls]` and separator tokens, and we fill the rest of the input window with tokens from the context, sliding the window 128 characters with each stride. At prediction time, we return the answer from the chunk with that has the highest predicted probability.

Negative examples We follow Longpre et al. (2019) and include "negative examples" during training. If a context chunk does not contain the answer span, we include the example as a training instance and train the model to indicate that the example does not contain the answer by selecting the `[cls]` token as the most likely start and end span. At prediction time, we discard "no answer" predictions and return the non-empty answer from the chunk with that has the highest predicted probability. For UnifiedQA, we train the model to predict an empty string for contexts that don't contain the answer to string and at prediction time return the non-empty answer with the highest probability.

A.3 Transfer learning details

For transfer learning, we take 1/2 of the K training examples for validation and train for 200 steps or until performance fails to improve on the validation loss fails to improve for 10 epochs, and we reduce the adapter learning rate to 1e-5. The other hyperparameters are the same as for in-domain learning.

Training UnifiedQA We download the pre-trained UnifiedQA-base model from HuggingFace and train it in the format described in Khashabi et al. (2020) and in the accompanying code release. [6] We lower-case the question and context strings and concatenate them with a special string "\n", which represents the backslash character followed by the letter n; and train the model to generate the answer string by minimizing cross-entropy loss. We use greedy decoding for prediction. In our pilot experiments, the recommended optimizer (Adafactor with a learning rate of 1e-3) quickly over-fits, so we use the same optimizer, learning rate, and batch size as for RoBERTa.

B Detailed Results

B.1 In-distribution Details

Figure 3 shows the training curve for training MADE, normalized by dividing each checkpoint score by the maximum validation accuracy obtained on that dataset during this run. The model reaches the maximum performance on the "easy" datasets early in training, which means that the model might over-fit to those datasets before converging on the more difficult datasets. MADE avoids this problem by tuning the adapter parameters separately after joint optimization. Inter-

[6] https://github.com/allenai/unifiedqa

Dataset	Domain	$\|c\|$	# train	# dev
SQuAD 1.1	Wikipedia	137	86,588	10,507
NewsQA	News articles	599	74,160	4,212
TriviaQA	Web snippets	784	61,688	7,785
SearchQA	Web snippets	749	117,384	16,980
HotpotQA	Wikipedia	232	72,928	5,904
Natural Questions	Wikipedia	153	104,071	12,836
BioASQ	Science articles	248	-	1,504
DROP	Wikipedia	243	-	1,503
DuoRC	Movie plots	681	-	1,501
RACE	Examinations	349	-	674
RelationExtraction	Wikipedia	30	-	2,948
TextbookQA	Textbook	657	-	1,503

Table 4: Information about the MRQA datasets, from Table 1 of Fisch et al. (2019), including the domain of the context passage; the average length of the context c in tokens; and the number of training and development examples. We downsample the in-domain datasets to 75K training and 1K development examples, and create few-shot training datasets from the out-of-domain datasets by reserving 400 examples as a test set and drawing different-sized training sets from the remaining examples, following Ram et al. (2021).

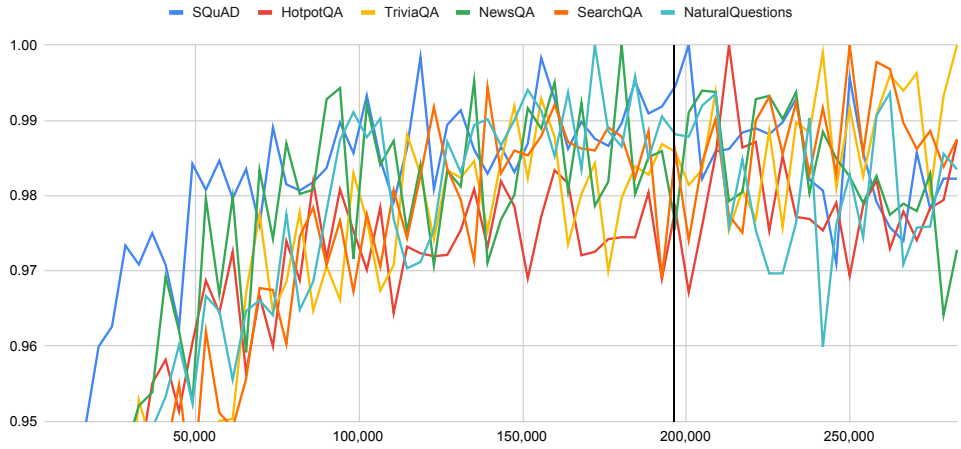

Figure 3: Validation F1 from the joint-optimization phase of MADE training, as a percentage of the maximum attained over the course of this run, plotted over number of optimization steps. The vertical black bar marks the checkpoint with the highest average F1, which is the checkpoint we select to keep, freezing the Transformer and continuing to tune the adapters separately (not shown).

estingly, adapter-tuning leads to improved performance on all datasets (Table 1), even datasets on which joint-optimization appears to have already converged.

B.2 Transfer Learning Details

Table 5 provides additional transfer learning results. Single-dataset adapters transfer worse than MADE. Freezing the Transformer parameters slighly improves results in the $K = 16$ setting but leads to worse performance with more data. The biggest drop is on BioASQ, possibly because it introduces new vocabulary and it is beneficial to update the token embeddings.

K	Model	BioASQ	DROP	DuoRC	RACE	RelEx	TextbookQA	Avg.
	UnifiedQA-base	59.7	45.2	30.2	**52.9**†	82.0	33.8	50.6
	Single-dataset adapters (pre avg.)	63.0	37.7	58.8	35.8	84.1	48.8	54.7
	Single-dataset adapters (post avg.)	62.1	40.9	58.8	30.4	84.0	48.8	54.2
16	MADE (pre avg., freeze θ)	66.7	52.8	66.0	47.4	**87.7**	60.2	63.5
	MADE (post avg., freeze θ)	66.5	52.6	66.8	47.5	87.7	**60.4**	**63.6**
	MADE (pre avg.)	**67.0**	**52.8**	65.7	47.5	**87.7**	59.8	63.4
	MADE (post avg.)	66.8	52.6	**67.2**	47.3	87.7	59.7	63.5
	UnifiedQA-base	65.5	47.9	32.9	**54.3**†	83.4	35.6	53.3
	Single-dataset adapters (pre avg.)	62.9	42.3	59.4	36.9	87.0	53.1	56.9
	Single-dataset adapters (post avg.)	60.7	39.2	59.0	36.4	84.1	49.6	54.8
64	MADE (pre avg., freeze θ)	66.9	53.3	**67.4**	47.9	87.9	**60.3**	64.0
	MADE (post avg., freeze θ)	65.4	52.2	66.8	47.9	87.8	59.9	63.3
	MADE (pre avg.)	69.4	**55.9**	66.8	47.6	88.4	60.2	64.7
	MADE (post avg.)	**70.9**	55.2	66.9	47.5	**88.5**	60.0	**64.8**
	UnifiedQA-base	71.7	49.8	36.5	**55.2**†	85.6	36.8	55.9
	Single-dataset adapters (pre avg.)	69.7	45.0	60.6	41.0	88.4	55.0	60.0
	Single-dataset adapters (post avg.)	71.3	44.4	58.8	35.8	86.5	49.2	57.7
256	MADE (pre avg., freeze θ)	66.8	53.5	66.9	48.0	87.9	60.2	63.9
	MADE (post avg., freeze θ)	66.6	52.0	**67.0**	47.8	87.8	59.9	63.5
	MADE (pre avg.)	73.9	56.9	65.3	48.7	88.9	61.5	65.9
	MADE (post avg.)	**75.1**	**56.9**	66.9	48.9	**90.2**	**61.8**	**66.6**

Table 5: Transfer learning to MRQA out-of-domain datasets with K training examples (F1, averaged over three random seeds), using the MADE model with adapter-tuning. †: RACE is part of the UnifiedQA training data. pre avg.: we take the weighted average of adapters at initialization before fine-tuning; post avg.: we fine-tune each adapter jointly and average them at the end. *Freeze* θ refers to experiments where we freeze the Transformer parameters rather than tuning them along with the adapters.