AMAL: Meta Knowledge-Driven Few-Shot Adapter Learning

S. K. Hong*
Samsung SDS
s.k.hong@samsung.com

Tae Young Jang
Samsung SDS
tae10.jang@samsung.com

Abstract

NLP has advanced greatly together with the proliferation of Transformer-based pre-trained language models. To adapt to a downstream task, the pre-trained language models need to be fine-tuned with a sufficient supply of annotated examples. In recent years, Adapter-based fine-tuning methods have expanded the applicability of pre-trained language models by substantially lowering the required amount of annotated examples. However, existing Adapter-based methods still fail to yield meaningful results in the few-shot regime where only a few annotated examples are provided. In this study, we present a meta-learning-driven low-rank adapter pooling method, called AMAL, for leveraging pre-trained language models even with just a few data points. We evaluate our method on five text classification benchmark datasets. The results show that AMAL significantly outperforms previous few-shot learning methods and achieves a new state-of-the-art.

1 Introduction

Since Transformer-based (Vaswani et al., 2017) pre-trained language models (PLMs) on massive corpora made a big impact on NLP, fine-tuning PLMs (Devlin et al., 2019; Lan et al., 2019; Liu et al., 2019) has led to large improvements in a variety of downstream NLP tasks. Yet, it is still challenging to fine-tune PLMs (Zhang et al., 2020) in the few-shot regime. Recently, *Adapters* (Houlsby et al., 2019a; Ben Zaken et al., 2022; Fu et al., 2022; Hu et al., 2021) have provided a method of fine-tuning PLMs more efficiently, by tuning some extra weights (the *Adapters*) while freezing the rest. Nevertheless, existing *Adapters* still fail to yield significant results in the few-shot regime. Refer to the Appendix table 4 for the performance of the prior Adapters on the few-shot classification problems.

*corresponding author

Since GPT-3 (Brown et al., 2020) was introduced, prompt tuning has swept the machine learning community. However, finding proper prompts (Schick and Schütze, 2020) is still a delicate task — requiring labor-intensive manual handcrafting with domain expertise as well as an in-depth understanding of the language model's inner mechanisms.

In this paper, we present a cost-effective method for language model fine-tuning that is applicable, without customization, to a variety of language models and Adapter types. We focus on small to mid-sized language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), BART (Lewis et al., 2020), or DeBERTa (He et al., 2020), because they are widely deployed in production systems due to their economy and low carbon footprint.

In this paper, we propose a meta-knowledge-driven few-shot adapter learning method, called AMAL (*Adapter-by-MetA-Learning*), based on a novel meta-learning framework, through which meta-level layer-wise adaptation kernels are derived in an end-to-end manner. Our design takes inspiration from (Aghajanyan et al., 2020), which proves that the over-parameterized pre-trained language models actually have low intrinsic dimension. We hypothesize that language model fine-tuning can be accomplished on a low intrinsic rank while keeping the pre-trained weights frozen, leading to our proposed low-rank adapter pooling approach.

AMAL includes two key ideas: (1) construction of language model adapters' intrinsic kernels from tasks and (2) inference of the optimal task-specific language model adapter for a given task, by referring to a meta-level latent embedding space over all tasks.

2 Related Work

Few-shot Text Classification: DS (Bao et al., 2019) refers to the underlying word distributions

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 10381–10389
December 7-11, 2022

across all available classes and specifies important lexical features for new classes. Frog-GNN (Xu and Xiang, 2021) focuses on all query-support pairs and proposes a multi-perspective aggregation-based graph neural network to explicitly reflect intra-class similarity and inter-class dissimilarity. LEA (Hong and Jang, 2022) proposes a meta learning-based document embedding approach and derives the meta-attention aspects dictionary to be reused when given a new task.

Parameter-Efficient Fine-Tuning: Houlsby et al. (2019a) proposed two trainable adapter layers per Transformer block where each adapter has two feedforward linear layers: one down-project and one up-project layer. BitFit (Ben Zaken et al., 2022) shows that tuning just the bias terms of a PLM is almost as effective as full fine-tuning. AdapterBias (Fu et al., 2022) improves on BitFit by changing the bias terms to be token-specific, with less trainable parameters. LoRA (Hu et al., 2021) is also an adapter-based fine-tuning approach where trainable rank decomposition matrices are injected into each layer of the Transformer architecture while the weights of the pre-trained model are frozen.

AMAL can be seen as similar with LoRA in terms of using the low-rank decomposition technique. However as a meta learning-based approach, AMAL can be applied to a broad range of language models and all existing adapter-based methods, including LoRA.

3 Background

3.1 Few-Shot Text Classification

We deal with the few-shot text classification problem to demonstrate AMAL's few-shot language model adaptation performance. As usual, C-way K-shot indicates that K-annotated examples are only given for each of the C number of classes for a task (denoted as τ_i), leading to the total number of examples as $K_{\tau_i} = K \times |\mathcal{C}|$.

3.2 Pre-Trained Language Models

We experiment with BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), BART (Lewis et al., 2020) and DeBERTa (He et al., 2020) as the underlying PLMs. They add a dummy token to an original tokens sequence so that the PLMs end up with providing the corresponding embedding (denoted [CLS]). In this study, the [CLS] plays an role in probing the distinctive properties for every incoming task.

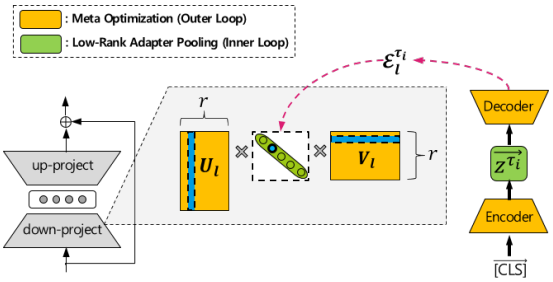

Figure 1: Low rank adapter pooling

3.3 Meta Learning

In the meta-learning setting, tasks are divided into a meta-training set ($\mathcal{S}^{tr}$), meta-validation set ($\mathcal{S}^{val}$), and meta-test set ($\mathcal{S}^{test}$) as disjoint sets of classes. Our meta-learning strategy follows the overall procedure of optimization-based meta-learning (Finn et al., 2017) so that our proposed low-rank adapters are learned by alternating between two different complementary processes: (1) low-rank adapter pooling (inner update) and (2) meta-optimization (outer update). For a task $\tau_i \sim p(\tau)$, the task data $\mathcal{D}_{\tau_i} = \{(x^i, y^i)\}$ consist of $\mathcal{D}_{\tau_i}^{tr}$ and $\mathcal{D}_{\tau_i}^{val}$ during the meta-training phase. In the meta-testing, the dataset of a new task τ_i is given as $\mathcal{D}_{\tau_i} = \left(\mathcal{D}_{\tau_i}^{tr}, \mathcal{D}_{\tau_i}^{te}\right)$ with a few annotated data points in the study.

4 Proposed Method: AMAL

In this section, we present the implementation of AMAL. The design implies the hypothesis that the language model adaptation can be performed on a low intrinsic rank. Here, we describe AMAL by employing the original Adapter (Houlsby et al., 2019b) method. Importantly, AMAL is orthogonal to existing Adapter methods and can be combined with any of them. AMAL offers a task-specific adapter for an incoming task. AMAL alternates between two update processes during meta-training: (1) low-rank adapter pooling and (2) meta-optimization.

4.1 Low Rank Adapter Model

As shown in Figure 1, as an element of the adapter, each projection matrix $\mathcal{P}_l \in \mathbb{R}^{d \times m}$ of the l-th layer is decomposed into three matrices:

$$\mathcal{P}_l = \mathcal{U}_l \times \mathcal{E}_l^{\tau_i} \times \mathcal{V}_l^T \quad (1)$$

where l is the layer's index, and $\mathcal{U}_l \in \mathbb{R}^{d \times r}$, $\mathcal{E}_l^{\tau_i} \in \mathbb{R}^{r \times r}$, $\mathcal{V}_l \in \mathbb{R}^{m \times r}$ given the PLM's original dimension d, the adapter's bottleneck dimension m

and the rank r ($r \ll \min(d, m)$). $\mathcal{E}_l^{\tau_i}$ is a diagonal matrix. For notational simplicity, we drop the distinction for the two different adapters (i.e., lower and upper) and likewise the distinction between up and down-projections. Importantly, $\mathcal{E}_l^{\tau_i}$ is the l-th layer's low-rank adapter pooler for the task τ_i, $\mathcal{U}_l$ the l-th layer's left adapter kernels, and $\mathcal{V}_l$ the right adapter kernels.

4.2 Low Rank Adapter Pooling (*inner update*)

The aim of the pooling is to derive the task-specific composition from the established adapter-kernels, $\mathcal{U}$ and $\mathcal{V}$, which are obtained in the meta-optimization process.

To obtain the optimal adapter for a task τ_i, there are two important steps in the pooling process: (1) encoding the task τ_i into a low-dimensional latent embedding space $\mathcal{Z}$ and (2) producing the task-specific adapter pooler from the latent embedding z^{τ_i}. The encoding pipeline is taken from Rusu et al. (2018). The reason why we employ the latent embedding space is to enable AMAL to summarize the properties extracted from tasks into the low-dimensional latent space $\mathcal{Z}$, instead of operating directly in the high dimensional parameter space. First, each task is fed into the encoding process, which is formulated as follows:

$$z_n^{\tau_i} = \frac{1}{NK^2} \sum_{k_n=1}^{K} \sum_{n=1}^{N} \sum_{k_m=1}^{K} f_{\theta_r}\left(f_{\theta_e}(c_{k_n}^{\tau_i}), f_{\theta_e}(c_{k_m}^{\tau_i})\right), \tag{2}$$

where $z_n^{\tau_i}$ denotes the latent space embedding for the particular class n under a given task τ_i, N indicates the total number of classes under the task, K denotes the total number of examples under each class, f_{θ_r} indicates the relation network (Sung et al., 2018), and f_{θ_e} is an encoder network to transform the delegate embedding [CLS] (denoted as $c_j^{\tau_i}$ for the case of the jth text instance of a specific task τ_i) prior to the relation network. As a result, the class embedding $z_n^{\tau_i}$ is led to keep track of the pairwise relationship with other classes, and the task-specific embedding z^{τ_i} is the concatenation of $z_1^{\tau_i}, ..., z_n^{\tau_i}$.

Subsequently, the task-specific latent embedding is delivered to the decoding process, which renders the latent embedding to generate the associated low-rank pooler. The decoding process is formulated as follows:

$$\mathcal{E}^{\tau_i} = f_{\theta_d}(z^{\tau_i}) \tag{3}$$

where $\mathcal{E}^{\tau_i}$ denotes the low rank adapter pooler for the task τ_i, f_{θ_r} indicates the decoder neural network, and z^{τ_i} is the task's latent embedding. To sum up, a new task is eventually converted into the task-specific low-rank adapter pooler via modulation on the low-dimensional latent space.

Algorithm 1 Our Proposed Meta-Training

Require: Meta training set $\mathcal{S}^{tr} \in \tau$, r (*rank*), d, m
Require: Learning rates α, β, λ, γ
Output: $\mathcal{U}, \mathcal{V}, \theta_e, \theta_r, \theta_d, \theta_\tau$

Randomly initialize $\mathcal{U}, \mathcal{V}, \theta_e, \theta_r, \theta_d, \theta_\tau$
Let $\phi = \{\mathcal{U}, \mathcal{V}, \theta_e, \theta_r, \theta_d, \theta_\tau\}$
while not converged **do**
  for number of tasks in batch **do**
    Sample task instance $\tau_i \sim \mathcal{S}^{tr}$
    Let $(\mathcal{D}^{tr}, \mathcal{D}^{val}) = \tau_i$
    Initialize $\theta'_{\tau_i} = \theta_\tau$ and ${z^{\tau_i}}' = z^{\tau_i}$
    for number of adaptation steps **do**
      Encode $[CLS]$ to ${z^{\tau_i}}'$ using f_{θ_e} and f_{θ_r}
      Produce $\mathcal{E}'_{\tau_i}$ from ${z^{\tau_i}}'$ using f_{θ_d}
      Generate document embeddings using H^{τ_i}
      Compute Task-Adaptation loss $\mathcal{L}_{\tau_i}^{tr}$
      Perform gradient step w.r.t. ${z^{\tau_i}}'$ and θ'_{τ_i}
      ${z^{\tau_i}}' \leftarrow {z^{\tau_i}}' - \alpha \bigtriangledown_{{z^{\tau_i}}'} \mathcal{L}_{\tau_i}^{tr}$
      $\theta'_{\tau_i} \leftarrow \theta'_{\tau_i} - \alpha \bigtriangledown_{\theta'_{\tau_i}} \mathcal{L}_{\tau_i}^{tr}$
    end for
    Generate document embeddings using H^{τ_i}
    Compute Meta-Optimization loss $\mathcal{L}_{\tau_i}^{val}$
  end for
  Perform gradient step w.r.t ϕ
  $\phi \leftarrow \phi - \beta \bigtriangledown_\phi \sum_{\tau_i} \mathcal{L}_{\tau_i}^{val} + \lambda \cdot \Omega + \gamma \cdot \mathcal{R}$
end while

4.3 Meta-Optimization (*outer update*)

As noted in Algorithm 1, AMAL updates three neural network blocks (i.e., θ_e, θ_r, θ_d) as well as the left adapter kernels $\mathcal{U}$ and the right adapter kernels $\mathcal{V}$, by minimizing the following objective function in the meta-optimization process:

$$\min_{\theta_e, \theta_r, \theta_d, \mathcal{U}, \mathcal{V}} \sum_{\tau_i} (\mathcal{L}_{\tau_i}^{val} + \lambda \cdot \Omega + \gamma \cdot \mathcal{R}) \tag{4}$$

where Ω indicates a weighted KL-divergence term, i.e., $D_{KL}(q(z^{\tau_i}|\mathcal{D}_n^{tr})||p(z^{\tau_i}))$ where $p(z^{\tau_i}) = \mathcal{N}(0, \mathcal{I})$, to regularize the latent space with the aim to learn a disentangled embedding. $\mathcal{R}$ denotes a penalty term to attain near-orthogonality in the construction of $\mathcal{U}$ and $\mathcal{V}$, and is formulated as follows:

$$\mathcal{R} = \|\mathcal{U}\mathcal{U}^T - \mathcal{I}\|_F + \|\mathcal{V}\mathcal{V}^T - \mathcal{I}\|_F \tag{5}$$

where F denotes the frobenius norm, and both $\mathcal{U}$ and $\mathcal{V}$ are randomly initialized. All the hyperparameters are equivalently kept all over the layers.

Table 1: Results of 5-way 1-shot and 5-way 5-shot classification

		Amazon		Huffpost		RCV-1		Reuters		20 Newsgroup	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML (Finn et al., 2017)		50.36 %	59.58 %	43.04 %	55.17 %	51.15 %	66.98 %	46.31 %	70.31 %	31.39 %	45.05 %
Proto (Snell et al., 2017)		45.54 %	71.30 %	34.70 %	50.69 %	44.77 %	58.91 %	62.41 %	73.05 %	31.38 %	37.02 %
LEO (Rusu et al., 2018)		49.09 %	59.48 %	45.07 %	60.69 %	51.30 %	63.90 %	59.13 %	73.10 %	37.72 %	48.08 %
Induction (Geng et al., 2019)		45.17 %	62.69 %	46.51 %	49.02 %	43.82 %	59.94 %	61.48 %	70.09 %	32.12 %	45.72 %
DS (Bao et al., 2019)		62.6 %	81.2 %	43.0 %	63.5 %	54.1 %	75.3 %	81.8 %	96 %	52.1 %	68.3 %
Frog-GNN (Xu and Xiang, 2021)		71.5 %	83.6 %	54.1 %	69.6%	-	-	-	-	-	-
LEA (Hong and Jang, 2022)		63.6 %	82.69 %	46.98 %	64.4 %	51.96 %	73.81 %	71.64 %	83.07 %	43.56 %	65.29 %
P-tuning v2 (Liu et al., 2022)	BERT	32.75 %	66.87 %	27.59 %	50.67 %	21.88 %	36.67 %	30.81 %	84.80 %	28.00 %	59.67 %
	RoBERTa	27.89 %	71.13 %	31.69 %	58.93 %	22.33 %	39.53 %	29.61 %	70.67 %	25.36 %	47.13 %
AMAL	BERT	**80.18 %**	89.07 %	56.27 %	74.31 %	63.73 %	83.11 %	90.84 %	**97.87 %**	56.80 %	70.49 %
	ALBERT	47.20 %	78.49 %	41.60 %	61.66 %	47.29 %	76.09 %	84.62 %	94.49 %	42.04 %	65.60 %
	RoBERTa	76.36 %	90.13 %	55.11 %	74.04 %	**71.73 %**	**87.02 %**	**92.0 %**	97.78 %	**60.27 %**	73.24 %
	BART	77.16 %	89.60 %	57.22 %	75.02 %	70.84 %	86.22 %	90.76 %	97.42 %	59.29 %	73.51 %
	DeBERTa-base	76.71 %	88.00 %	54.31 %	73.42 %	71.56 %	82.76 %	85.42 %	95.02 %	52.18 %	69.42 %
	DeBERTa-large	79.20 %	**90.58 %**	**60.27 %**	**78.04 %**	71.43 %	84.44 %	91.29 %	97.86 %	53.87 %	**75.29 %**

Note: The highest performance in each dataset is highlighted in **Bold**.

5 Experimental Results

5.1 Document Embedding for Classification

Here, we briefly explain how we generated document embeddings for our experiments. For a text input with length L, we utilize the embedding vectors for the individual tokens from the last layer of the given PLM, which are denoted as $H_j^{\tau_i} = [h_{j,1}^{\tau_i}, \ldots, h_{j,L}^{\tau_i}]$ for the jth text example of the task τ_i. For text classification, we average $H_j^{\tau_i}$ column-wise and then feed it into a fully connected neural network with the parameters θ'_{τ_i}, which are optimized for the inner-update.

5.2 Dataset and Baselines

We evaluate AMAL on five text classification datasets: 20 Newsgroups (Lang, 1995), Huffpost headlines (Misra and Grover, 2021), Reuters-21578 (Lewis., 1997), RCV1 (Lewis et al., 2004) and Amazon product reviews (He and McAuley, 2016). We compare AMAL with eight baseline methods: **MAML** (Finn et al., 2017), **PROTO** (Snell et al., 2017), **LEO** (Rusu et al., 2018), **INDUCTION** (Geng et al., 2019), **DS** (Bao et al., 2019), **Frog-GNN** (Xu and Xiang, 2021), **LEA** (Hong and Jang, 2022) and **P-tuning v2** (Liu et al., 2022) as a prompt-based fine-tuning method. We follow the same experimental settings as in (Bragg et al., 2021) for all datasets, except for RCV-1 for which we use the split of (Bao et al., 2019).

5.3 Overall Performance

We evaluate AMAL in both 5-way 1-shot and 5-way 5-shot settings and the results are shown in Table 1. All the scores were calculated as the average of three trials. In the results, the BERT$_{base}$ was used as the base PLM if there is no explicit indication. For MAML, Proto, LEO and Induction, the document embeddings are formed as explained in 5.1. For DS and Frog-GNN, we quoted the reported results from (Bao et al., 2019) and (Xu and Xiang, 2021), since the experiment settings are identical. We applied AMAL to a wide range of small to mid-sized PLMs: BERT$_{base}$ (Devlin et al., 2019), ALBERT$_{base}$ (Lan et al., 2019), RoBERTa$_{base}$ (Liu et al., 2019), BART$_{base}$ (Lewis et al., 2020) and DeBERTa (He et al., 2020) to verify the applicability of AMAL. For BART$_{base}$, we treated the decoder's final hidden state embedding of the last token as the [CLS] embedding, as in (Lewis et al., 2020).

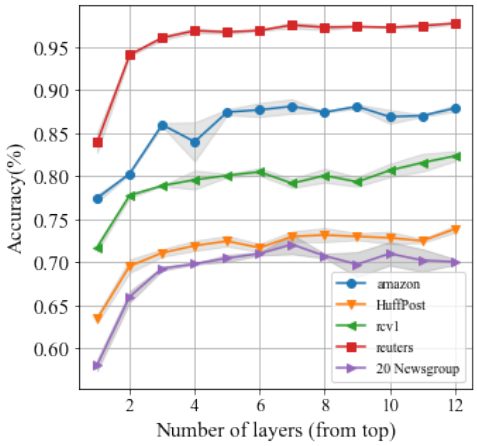

Figure 2: 5-way 5-shot prediction accuracy depending on the number of top-most layers with adapters.

As shown in Table 1, AMAL outperforms the previous methods specialized for few-shot classification over all of the datasets by a large margin: 27.69% in 5-way 1-shot classification and 22.03% in 5-way 5-shot classification. The evaluation results demonstrate that AMAL offers agile adaptation of diverse small to mid-sized PLMs in the few-shot regime.

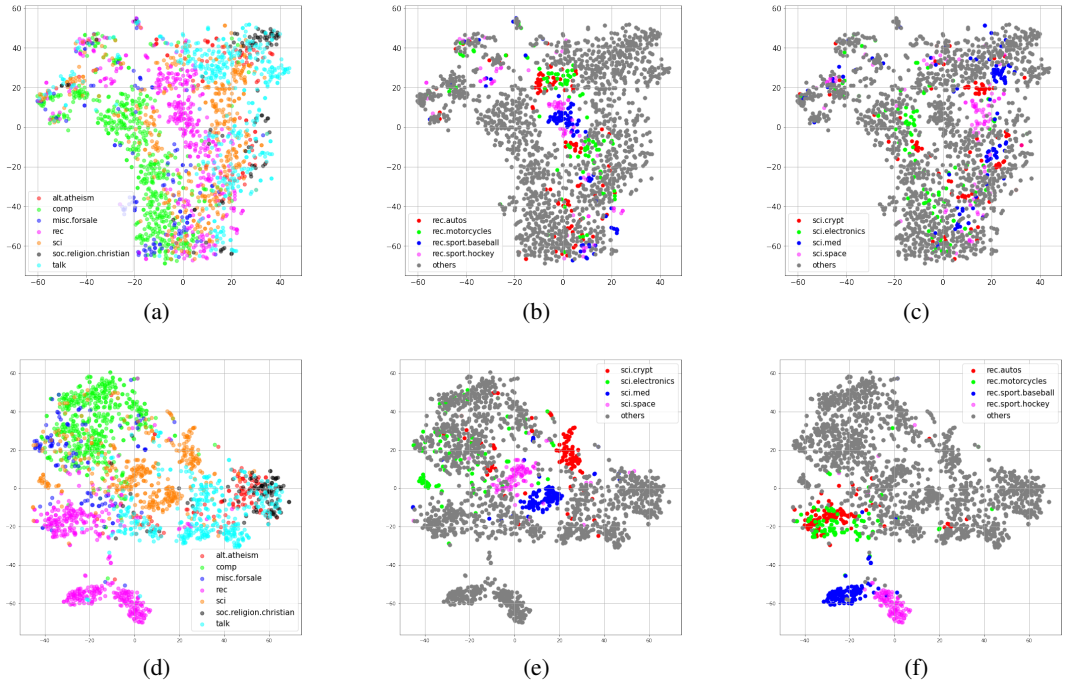

Figure 3: t-SNE plot of the embedding space before and after adaptation with 20 newsgroup. (a)-(c) exhibit it before the low rank adapter pooling process. (d)-(f) show the task-specific embedding space after the pooling process. (a), (d) the embedding for seven top-level macro domains. (b), (e) Same as (a) and (d) but highlighted for the classes under the *science* domain. (c), (f) Same as (a) and (d) but highlighted for the four classes under *recreation* domain.

5.4 The Impact of the Number of AMAL-equipped Layers

We explore the effect of the number of layers equipped with AMAL. Here, the BERT$_{base}$ is employed as the base PLM. We monitored performance while incrementally extending the number of AMAL-equipped layers, starting from the last layer and proceeding towards the input layer. As shown in Figure 2, giving priority to the top-most layers is an even more cost-effective way to apply AMAL. It is also evident that starting from the sixth or seventh layer from the top, the benefit of inserting AMAL into the next lower layer becomes insignificant. These results show that the performance of few-shot learning can be greatly improved even with a small number of adapters. According to this empirical analysis, we can maximize the efficiency in a fine-grained manner by adjusting the number of AMAL-equipped layers.

5.5 Visualization of Task-Specific Document Embeddings

We plot the initial document embeddings and the corresponding fine-tuned embeddings obtained by AMAL for 20 Newsgroups dataset (Figure 3). For the visualization, we randomly sampled four hundred tasks, each of which is composed of 5-way 1-shot from all available classes. All of the embeddings were projected into 2-D space via t-SNE. Figures 3a, 3b, and 3c show the initial embeddings before the adaptation, and Figures 3d, 3e, and 3f exhibit their adapted embeddings. Figures 3a and 3d show the adapted embeddings for 'atheism', 'computer', 'for-sale', 'recreation', 'science', 'religion', and 'talk'. Figures 3b and 3e show the topics for the 'science' domain. Figures 3c and 3f show the topics for the 'recreation' domain.

6 Conclusion

We hypothesized that language model adaptation can be performed on a low intrinsic rank, especially when only a few examples are offered. We designed a novel meta-learning-based low-rank adaptation method for leveraging small to mid-sized pre-trained language models, allowing a new task to be cost-effectively learned in the few-shot regime. We demonstrated that the combination of low-rank matrix decomposition and meta learning is so effective, that we can reap the benefits of small to mid-sized pre-trained language models in practical scenarios with scarce annotated data.

7 Limitations

AMAL may be difficult to apply to unidirectional language models such as GPT2 (Radford et al., 2019) and GPT3 (Gao et al., 2021). This is because unidirectional models only encode the context that resides to the left of the [CLS] token in the input.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2019. Few-shot text classification with distributional signatures. In *International Conference on Learning Representations*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

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

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

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-yi Lee. 2022. Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks. *arXiv preprint arXiv:2205.00305*.

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.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. 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 3904–3913.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

S. K. Hong and Tae Young Jang. 2022. Lea: Meta knowledge-driven self-attentive document embedding for few-shot text classification. In *North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019a. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019b. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier.

David D. Lewis. 1997. Reuters-21578, distribution 1.0.

David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

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.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

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*.

Rishabh Misra and Jigyasa Grover. 2021. *Sculpting Data for ML: The first act of Machine Learning*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2018. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Shiyao Xu and Yang Xiang. 2021. Frog-gnn: Multi-perspective aggregation based graph neural network for few-shot text classification. *Expert Systems with Applications*, 176:114795.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

A Appendix

A.1 Datasets

We introduce the datasets and the split (i.e., train/val/test) which had been maintained in our experiments.

20 Newsgroups is a collection of discourses in newsgroup posts for 20 topics (Lang, 1995).

Huffpost Headlines is a collection of news headlines published in the Huffington Post from 2012 to 2018 (Misra and Grover, 2021). It is composed of 41 topics.

Reuters-21578 is composed of documents that appeared on the Reuters newswire in 1987 (Lewis., 1997). In addition, we adopted the ApteMod version and discarded the documents with more than one label to avoid ambiguity, and thus 31 classes remain.

RCV-1 is a set of newswire stories published by Reuters journalists from 1996 to 1997 (Lewis et al., 2004) and comprises 71 topic classes.

Amazon data is a real-world dataset collected from Amazon.com as a set of customer reviews from 24 types of product categories(He and McAuley, 2016).

To train and evaluate the models, we divided each of the aforementioned datasets into a meta-training set ($\mathcal{S}^{tr}$), meta-validation set ($\mathcal{S}^{val}$), and meta-test set ($\mathcal{S}^{test}$) as disjoint sets of classes within the experimental setting. In this work, we followed the same split of classes as in (Bragg et al., 2021) for all datasets.

A.2 Implementation Details

The Table 2 specifies the detailed architecture of AMAL. In the encoder module, the [CLS] vector, which is the same size of the output of the BERT-base-uncased, ALBERT-base, and RoBERTa-base is linearly transformed into a 64-dimensional vector. The relation network module is a two-layers neural network with the ReLU activation. The the decoder module is a single-layer neural network with the ReLU. Finally, the classifier is a single-layer neural network with the ReLU.

Table 2: Architecture details of AMAL

Module Name	Architecture	Shape of (input, output)	The number of Params
Encoder	linear	(768, 64)	153.6K
Relation Network	2-layer MLP with ReLU	First layer (2X64, 2X64) ReLU Second layer (2X64, 2X64) ReLU	32.8K
Decoder	1-layer MLP with ReLU	Input layer : (64, 150) ReLU Output layer : (150, 154)	32.7K
Task Classifier	1-layer MLP with ReLU	Input layer : (768, 300) ReLU Output layer : (300,5)	231.9K

A.3 Training Details and Hyperparameter Tuning

We summarize the details of the model training and evaluation in Table 3. "# of tasks" means the number of tasks in each batch during model training. For example, for the 20 newsgroups dataset, the 20 classes of news topics are split into 8 classes for meta-training, 5 classes for meta-validation, and 7 for meta-testing. When composing a batch for meta-training, since # of tasks is 4, the following is repeated 4 times: the 5 classes (since our few-shot setup is 5-way) for a task are randomly selected from the given 8 classes.

In table 3, "# of queries" indicates the number of data points for a class, where the data points are used for the calculation of the meta-optimization loss and the accuracy in the outer-loop of the meta-training and meta-testing step, respectively. During the meta-training, we sample four tasks with 15 queries from $\mathcal{S}^{tr}$, hence performing the low rank adapter pooling four times per meta-optimization.

Early-stopping was employed during model training: model training was stopped if the validation loss did not improve for 20 steps. For both the validation and testing, we sample 15 tasks with 15 queries from $\mathcal{S}^{val}$ and $\mathcal{S}^{test}$. We used the Adam optimizer with learning rates of 0.1 and 0.001 in the inner and outer updates, and the inner update is repeated 40 times. During the meta-optimization process (outer loop), we apply weight decay scheduling. In addition, the coefficient λ of the KL-Divergence term in eq. 4 was set to 0.001 and the coefficient γ of the penalty term in eq. 4 was set to 0.1. We performed all the experiments on a single NVIDIA A100 80GB GPU.

Table 3: Hyperparameters for Model Training

Hyperparameters		
meta-training set	# of tasks	4
	# of queries	15
meta-validation set	# of tasks	15
	# of queries	15
meta-test set	# of tasks	15
	# of queries	15
learning rates in inner loop		0.1
learning rates in outer loop		0.001
scheduler steps		5
λ, weight of KL-Divergence (eq. 4)		0.001
γ, weight of latent variable penalty (eq. 4)		0.1
number of adaptation steps		40

Table 4: Few-shot Classification Performance of Adapters

	Amazon	Huffpost	RCV1	Reuters	20 newsgroup
Freeze	25.33 %	30.67 %	17.33 %	26.67 %	29.33 %
Full fine-tuning	25.33 %	29.33 %	18.67 %	28.00 %	32.00 %
Adapter (Houlsby et al., 2019b)	30.67 %	28.00 %	18.67 %	25.33 %	26.67 %
BitFit (Ben Zaken et al., 2022)	26.67 %	28.00 %	22.67 %	28.00 %	26.67 %
AdapterBias (Fu et al., 2022)	29.33 %	28.00 %	12.00 %	29.33 %	25.33 %
LoRA (Hu et al., 2021)	28.00 %	28.00 %	17.33 %	26.67 %	26.67 %

A.4 Few-Shot Performance of Adapter-based Fine-tuned Methods

In addition, to verify the validity of our assumption that is introduced in section 1, we find the performance of parameter efficient fine-tuned methods, i.e., (Houlsby et al., 2019b; Ben Zaken et al., 2022; Fu et al., 2022; Hu et al., 2021), using 5-way 5-shot. As shown in Table 4, we cannot find the meaningful results in a few-shot settings with parameter efficient fine-tuned methods.

A.5 The Number of Fine-tuning Parameters of Adapters

The Table 5 shows the number of parameters of the Adapter-based fine-tuning methods: the original Adapter (Houlsby et al., 2019b), BitFit (Ben Zaken et al., 2022), Adapter-Bias (Fu et al., 2022), LoRA (Hu et al., 2021) and AMAL(Ours) on the

Table 5: The number of the fine-tuning parameters of Adapters

	# of fine-tuned params.
Adapter (Houlsby et al., 2019b)	1.23M
BitFit (Ben Zaken et al., 2022)	0.10M
Adapter-Bias (Fu et al., 2022)	12K
LoRA (Hu et al., 2021)	0.294M
AMAL Ours	0.595M

Table 6: The performance for 5-way 5-shot depending on the bottleneck size and the rank size.

bottleneck size(m	rank(r)	accuracy
32	32	89.42%
	16	87.64%
	8	88.89%
64	64	88.36%
	32	88.27%
	16	89.69%
	8	88.44%
128	128	88.44%
	64	88.09%
	32	90.31%
	16	88.89%
	8	89.60%

BERT$_{base}$(12 layers). Here, AMAL's latent embedding space is set to 64 dimensions. As revealed, AMAL requires the smallest amount of fine-tuning parameters.

A.6 The effect of the bottleneck size and the rank size

We observed the effect of the two hyper-parameters, each of which is the adapter size and its rank, respectively. The base language model is the BERT$_{base}$. In the experiment, we changed the bottleneck size as 32, 64, 128 and their related, diverse rank sizes on the Amazon product reviews data. The table 6 shows the performance for the 5-way 5-shot. It is observed that the adaptation of language models can be settled on a low intrinsic dimension as mentioned in (Hu et al., 2021) and (Aghajanyan et al., 2020).