HyperLoader: Integrating Hypernetwork-Based LoRA and Adapter Layers into Multi-Task Transformers for Sequence Labelling

Anonymous ACL submission

Abstract

We present HyperLoader, a simple approach that combines different parameter-efficient fine-tuning methods in a multi-task setting. To achieve this goal, our model uses a hypernetwork to generate the weights of these modules based on the task, the transformer layer, and its position within this layer. Our method combines the benefits of multi-task learning by capturing the structure of all tasks while reducing the task interference problem by encapsulating the task-specific knowledge in the generated weights and the benefits of combining different parameter-efficient methods to outperform full-fine tuning. We provide empirical evidence that HyperLoader outperforms previous approaches in most datasets and obtains the best average performance across tasks in high-resource and low-resource scenarios.

1 Introduction

Parameter-efficient fine-tuning techniques emerge as an alternative to conventional fine-tuning, where only a small number of parameters is updated to a downstream task (Houlsby et al., 2019; Stickland and Murray, 2019; Karimi Mahabadi et al., 2021a). These methods aim to achieve comparable performance to full fine-tuning by updating as few parameters as possible. However, a less studied research direction related to these methods is whether one can perform better than full fine-tuning with fewer parameters (Mao et al., 2022).

Although these methods considerably reduce the number of parameters to achieve good performance on different tasks, a specialized model for each task remains necessary. Multi-task learning reduces the computational cost by training a single model while enabling information sharing by capturing the common structure underlying all tasks. Recently, researchers have developed approaches combining parameter-efficient fine-tuning techniques in multi-task settings (Stickland and Murray, 2019; Pfeiffer et al., 2020, 2021; Rücklé et al., 2021). Nevertheless, this approach can lead to underfitting in high-resource tasks and overfitting in low-resource tasks (Lee et al., 2017). Another potential issue is task interference, where an improvement in the performance of one task reduces the performance of other tasks (Wang et al., 2019).

An alternative to reduce the negative effects of multi-task learning is to use hypernetworks (Ha et al., 2017) to generate separate weights for each task (Karimi Mahabadi et al., 2021b; Ivison and Peters, 2022). However, combining different parameter-efficient methods in a multi-task setting using hypernetworks is an unexplored research area. To address this limitation, we propose HyperLoader, a simple method that employs a neural network to generate the weights for a combination of two parameter-efficient fine-tuning techniques: adapters (Houlsby et al., 2019) and LoRA (Hu et al., 2021) based on the task, the transformer layer of the model and the position of the method within this layer. We use the encoder-decoder T5 model (Raffel et al., 2020) for all experiments to take advantage of modelling the tasks as sequence-to-sequence tasks. We test our model in seven datasets from two Sequence Labelling tasks. The first task is Named Entity Recognition, a valuable tool in various real-world scenarios in the era of large language models such as healthcare and medical research (Raza et al., 2022; Hu et al., 2024), Finance and Business Intelligence (Zhou et al., 2023), and analyzing legal texts (Trias et al., 2021). The second task is slot-filling, a crucial step to enable dialogue systems to accurately understand and fulfil user requests by extracting and organizing essential information from user inputs (Cheng et al., 2023a,b; Firdaus et al., 2023).

Our model achieves the best average performance using the complete training and validation data for each task, as well as in a low-resource configuration with only 10% and 20% of the train-

ing and validation data available. We empirically demonstrate that the improvement is not only due to adding more trainable parameters but also by combining different parameter-efficient methods in a multi-task setting using hypernetworks, which adequately supports Sequence Labelling tasks in high- and low-resource scenarios.

Our main contributions are the following:

- We propose a multi-task learning approach that combines different parameter-efficient fine-tuning methods based on hypernetworks conditioned on the task, the layer in the transformer model, and the position of the method within this layer.
- We provide empirical results on different Sequence Labelling datasets demonstrating the effectiveness of our model compared to single-task and multi-task approaches using the entire datasets and in low-resource scenarios.

2 Related Work

Parameter-efficient fine-tuning techniques are an alternative to full fine-tuning by updating a small number of parameters per task, yielding similar performance to full fine-tuning. One of the most popular approaches in this area is the adapter (Houlsby et al., 2019), a small trainable bottleneck layer added in each transformer block of a model. It consists of down and up projections and has shown similar performance to full-fine-tuning. Other popular parameter-efficient fine-tuning approaches are Prefix-tuning (Li and Liang, 2021) and LoRA (Hu et al., 2021). Prefix-tuning prepends a fixed-length, learnable sequence of prefix tokens to the input embeddings while the original model parameters remain unchanged. This approach optimizes a continuous prompt, effectively guiding the model's behaviour for specific tasks with minimal computational overhead. LoRA adds trainable low-rank matrices and combines their outputs with the original matrices in the self-attention layer of the transformer; the main difference with the previous approaches is that it does not employ any activation function.

A less-studied area is how to achieve better performance than full-fine tuning using as few parameters as possible. The UniPELT framework (Mao et al., 2022) addresses this limitation. This framework incorporates several parameter-efficient fine-tuning methods as sub-modules and learns how to activate them dynamically depending on the input or task setup using a gating mechanism, which assigns more weight to the sub-modules that increase the performance of a given task. They considered three different methods: Adapters, LoRA, and prefix-tuning, and they used the BERT base model for all their experiments.

Although promising, this approach still needs a specialized model for each task. The use of hypernetworks to generate the weights of parameter-efficient methods in a multi-task learning approach is a promising research area. By using this type of network, the model can capture the common structure underlying all target tasks and encapsulate the specific task knowledge by generating different weights based on the task, reducing the negative effects of this setting such as underfitting and overfitting in high-resource and low-resource tasks, respectively or task interference.

The HyperFormer++ model (Karimi Mahabadi et al., 2021b) is the main foundation of our model. It uses task-conditioned hypernetworks to generate the weights of adapter layers and layer normalizations in a multi-task setting. The hypernetwork is trained to create adapter parameters specific to each task and layer, based on the embeddings of task and layer IDs. This hypernetwork is trained simultaneously for all tasks, allowing it to share knowledge across different tasks. At the same time, it reduces negative interference by producing distinct adapter layers for each task. The T5 model is used as the backbone model for all the experiments.

Another hypernetwork-based multi-task approach is Hyperdecoders (Ivison and Peters, 2022). This framework generates input-conditioned adapter layers for the decoder in an encoder-decoder model. This approach offers more flexibility by generating unique parameters for every input instead of using a learnt task embedding and allows making use of the similarities between samples across datasets.

Our HyperLoader model combines the benefits of both worlds. It uses different parameter-efficient fine-tuning techniques and task-conditional hypernetworks to generate its weights conditioned on the task, the transformer layer of the model, and the position of the parameter-efficient method within this layer. Our results show a performance improvement in all tasks and, on average, using the full training and validation data and simulating a low-resource setting.

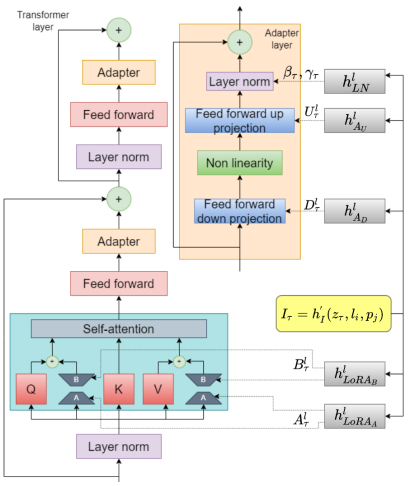

Figure 1: Diagram of the HyperLoader model. The Adapter hypernetworks $h^l_{A_D}$ and $h^l_{A_U}$ produce the weights D^l_τ and U^l_τ for task-specific adapter modules. The LoRA hypernetworks $h^l_{LoRA_A}$ and $h^l_{LoRA_B}$ generate the A and B matrices for task-specific LoRA modules. Finally, the hypernetwork h^l_{LN} creates the conditional layer normalization parameters β_τ and γ_τ.

3 Methodology

Multi-task hypernetwork-based approaches using parameter-efficient methods are a low-cost alternative for different NLP tasks that achieve comparable results to full-fine-tuning. Nevertheless, exploring approaches to obtain better results than updating all weights during fine-tuning is an unexplored research area. We introduce the HyperLoader model to address this gap. Our approach incorporates **hyper**network-based **Lo**RA and **Ad**apter layers into a multi-task transform**er** model, based on the HyperFormer++ model (Karimi Mahabadi et al., 2021b).

Our method enables the hypernetwork to generate the weights of the adapter layers and LoRA matrices based on the task, the transformer model's layer, and the position of the adapter and LoRA matrix within this layer. Figure 1 shows the diagram of our proposed approach. We generate an input embedding I_τ based on the task τ, the layer l_i and the position of the Adapter and the LoRA matrices p_j. The embedding I_τ is the input to a set of hypernetworks (h^l_{LN}, $h^l_{A_U}$, $h^l_{A_D}$, $h^l_{LoRA_B}$, $h^l_{LoRA_A}$) to generate the weights for the adapter layers, the LoRA matrices and the layer normalization. Therefore, we only update the hypernetwork parameters, the input embedding projector h' and the layer normalizations in $f_\theta(\cdot)$ during training, while the rest of parameters θ remain fixed.

In this section, we describe our approach. First, we provide preliminary information by defining multi-task learning, which is the foundation of our work. Second, we describe the main components of our proposed HyperLoader model. Finally, we describe the format for converting Sequence Labelling into a sequence-to-sequence task.

3.1 Preliminaries: Multi-task Learning

Our proposed model generates the weights of adapter layers and LoRA matrices based on the task, the transformer layer of the model, and the position of the adapter and the LoRA matrix in this layer in a multi-task setting. Given a set of tasks $\{D_{\tau_{\tau=1}^T}\}$ where T is the total number of tasks, $D_\tau = \{(x^i_\tau, y^i_\tau)\}_{i=1}^{N_\tau}$ is the training data of the τ-th task with N_τ instances and $f_\theta(\cdot)$ is a pre-trained model parameterized by θ, the multi-task fine-tuning minimizes the loss shown in equation 1 on the training set, where l is the cross-entropy loss and w_τ is the sampling weight for the τ-th task.

$$\mathcal{L}(\theta, \{D_\tau\}_{\tau=1}^T) = \sum_{\tau=1}^{T} \sum_{(x^i_\tau, y^i_\tau) \in D_\tau} w_\tau l(f_\theta(x^i_\tau), y^i_\tau) \quad (1)$$

3.2 HyperLoader: Task Conditional Adapter Layers

Adapters (Houlsby et al., 2019) are small sub-modules inserted within layers of a pre-trained transformer-based model before the skip connections. In the HyperLoader model, the conditional adapter A^l for layer l consists of a down-projection $\mathcal{D}^l_\tau \in \mathbb{R}^{h \times d}$ into a lower dimension $d_{bottleneck}$, a $GeLU$ non-linearity, and an up-projection $\mathcal{U}^l_\tau \in \mathbb{R}^{d \times h}$ that projects back into the original hidden layer dimension. The output of the adapter is defined in equation 2, where x is the input hidden state, LN^l_τ is the task conditional layer normalization and the adapter weights $(\mathcal{D}^l_\tau, \mathcal{U}^l_\tau)$ are generated by a hypernetwork. We use a reduction factor $r = 32$ for the down-projection in all the experiments.

$$A^l_\tau(x) = LN^l_\tau\left(\mathcal{U}^l_\tau(GeLU(\mathcal{D}^l_\tau(x)))\right) + x \quad (2)$$

3.3 HyperLoader: Task Conditional Layer Normalization

The task conditional layer normalization, defined in equation 3, involves element-wise multiplication denoted by $\odot$. Here, μ_τ and σ_τ represent the mean and standard deviation of the training data for the τ-th task, while γ^l_τ and β^l_τ are weights generated by a hypernetwork.

$$LN^l_\tau(x^i_\tau) = \gamma^l_\tau \odot \frac{x^i_\tau - \mu_\tau}{\sigma_\tau} + \beta^l_\tau \quad (3)$$

3.4 HyperLoader: Task Conditional LoRA Matrices

Low-Rank Adaptation (LoRA) (Hu et al., 2021) is a parameter-efficient fine-tuning technique that injects trainable rank decomposition matrices into each layer of the Transformer architecture. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, this method constrains its update by representing it with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$. The forward pass using LoRA is denoted in equation 4. Where x is the input hidden state, W_0 is frozen and does not receive gradient updates and matrices A and B are generated by a hypernetwork. Based on previous work (Hu et al., 2021; Mao et al., 2022), we only add Low-Rank matrices to the query and value matrices in the attention layers of the transformer. We use a low-dimensional rank $r = 8$.

$$h = W_0 x + \Delta W x = W_0 x + BAx \quad (4)$$

3.5 HyperLoader: Task Conditional Hypernetworks

A hypernetwork (Ha et al., 2017) is a neural network that generates the parameters for another neural network. It captures shared information across tasks, while the generated task-specific adapters and layer normalization allow the model to adapt to each task individually, reducing negative interference in a multi-task setting. The input for these networks is a task embedding z_τ and its concatenation with a layer *id* embedding $\mathcal{I} = \{l_i\}^L_{i=1}$ and an adapter position embedding $\mathcal{P} = \{p_j\}^6_{j=1}$. In the encoder-decoder model, we consider six positions for the parameter-efficient modules: the adapter layers and the LoRA matrices. We consider two options for the position of the adapter layers in each transformer block: after the self-attention layer or after the feed-forward layer. For LoRA matrices, we consider four positions: the A and B matrices in the self-attention layer and the A and B matrices in the cross-attention layer in the decoder of the model.

Using this configuration, the hypernetwork is able to generate different weights based on the task, position of the adapter, LoRA matrices and layer of the transformer network. The final input embedding I_τ for the hypernetwork is computed using a projector network h'_I as shown in equation 5, where z_τ, l_i and p_j are learnable parameters via back-propagation. We set the dimension of the task feature embedding z_τ to $d_{z_\tau} = 512$. For the layer and position embeddings, we use a dimension $d_{l,p} = 64$. Finally, for the final input embedding, we set a dimension of $d_{I_\tau} = 64$.

We use five hypernetworks for the HyperLoader model. For the Adapters, we consider two hypernetworks: one for the down projection and one for the up projection. Since we are using a reduction factor $r = 32$ and the hidden dimension of the model is $d = 768$, the down sample size is $\lfloor 768/32 \rfloor = 24$, the dimensions of the output matrices of the down and up projections are 768×24 and 24×768, respectively. We use two hypernetworks to generate the weights of the LoRA Matrices. One for the low-rank matrix A and one for the low-rank matrix B. We use a low-dimensional rank $r = 8$ for the Query and Value matrices in the self-attention and the cross-attention; therefore, the output matrices A and B dimensions are 768×8 and 8×768, respectively. Finally, we use one hypernetwork for the task-conditioned layer normalization that generates vector weights of 768 dimensions.

$$I_\tau = h'_I(z_\tau, l_i, p_j) \quad (5)$$

3.6 SentT' Format

Converting Sequence Labelling into a sequence-to-sequence task is essential to using an encoder-decoder model. We use the SentT' (Farina et al., 2023) format for the input and output sequences to achieve this goal. This format is obtained by turning a labelled sequence S into a sequence of tokens $x_1 \dots x_L$ of size L using white-space splitting and then interleaving these tokens with the spacial sentinel tokens used to pre-train T5 to obtain the input string $S_{in} = s_0 x_1 s1 \dots x_L s_L$. We use the *simplified Beginning, Inside and Outside* (sBIO) format to represent the output strings, where given a set T of labels to annotate a text span, $t \in T$ is used to represent any token that starts a labelled span, a single tag I for each token that continues a labelled span and O to tag tokens that do not belong to labelled spans. Analogous to the input sequence, we interleave the tokenized output string with the sentinel tokens to obtain the output sequence using the SentT' format $S_{out} = s_0 t_1 s_1 \dots t_L s_L$. Table 1 illustrates how to convert the text "play the song little robin redbreast" from the SNIPS dataset into the SentT' format.

Encoded input using the SentT' format													
<extra_id_0>	play	<extra_id_1>	the	<extra_id_2>	song	<extra_id_3>	little	<extra_id_4>	robin	<extra_id_5>	redbreast	<extra_id_6>	
Encoded output using the SentT' format													
<extra_id_0>	O	<extra_id_1>	O	<extra_id_2>	MUSIC_ITEM	<extra_id_3>	TRACK	<extra_id_4>	I	<extra_id_5>	I	<extra_id_6>	

Table 1: Example of an input/output instance transformed into the SentT' format. The text is tokenized with white-space splitting and then interleaved with sentinel tokens used for T5 pre-training to obtain the input string. The output uses the sBIO format: a label $t \in T$ is used to represent any token that starts a labelled span, a single tag I for tokens that continue a labelled span and O to tag tokens outside labelled spans.

4 Experiments

This section describes our experiments for Sequence Labelling using the HyperLoader model.

4.1 Corpora

We use seven publicly available corpora for Sequence Labelling tasks (slot-filling and Named Entity Recognition) to cover different domains and distributions and gather robust empirical evidence.

For slot-filling, we use four different dialogue-oriented datasets. First, the ATIS dataset (Hemphill et al., 1990) consists of manual transcripts of audio recordings about people asking for flight information on automated airline travel inquiry systems. Second, the SNIPS dataset (Coucke et al., 2018) comprises users' intent queries distributed in seven domains: search creative work, get weather conditions, restaurant reservations, play music, add elements to a playlist, and search screening events. The third and fourth datasets are the English portions of two multilingual task-oriented dialogue datasets: The mTOP dataset (Li et al., 2021) consists of eleven domains: alarm, calling, event, messaging, music, news, people, recipes, remainders, timer and weather conditions. Finally, the mTOD dataset (Schuster et al., 2019) contains nine labels across three domains: alarm, reminders and weather conditions.

For Named Entity Recognition, we use the MIT Corpora (Movie, MovieTrivia and Restaurant)[1]. In the case of Movie and MovieTrivia datasets, both are composed of 12 labels, and the latter contains more complex queries. Finally, the Restaurant dataset contains 8 labels such as restaurant names, ratings, dishes and opening times. We show the complete list of labels for each dataset in Appendix A. Based on the label's name, we obtained the label overlap across the datasets. Movie and MovieTrivia have the highest number of overlaps with four labels: genre, year, plot, director, and actor. These datasets also share the labels "genre" and "year" with the SNIPS dataset. The mTOD, mTOP and restaurant datasets share the label "location". Movie and Restaurant datasets share the label "rating". Finally, Restaurant and SNIPS datasets share two labels: "restaurant_name" and "cuisine". Nine different labels are shared across the datasets, representing 3.78% of the total of labels.

We used the original partitions for the mTOP and mTOD datasets, in the case of the ATIS and SNIPS datasets, we used the data partitions used originally by (Goo et al., 2018)[2], finally, we manually create a validation subset by sampling ten per cent of each original training partition for each MIT Corpus following previous methodologies (Raman et al., 2022; Farina et al., 2023). Table 2 shows the statistics for the corpora we used in this work. It is important to mention that although there are some duplicates in the datasets, we do not remove these instances.

Corpora	Train	Val	Test	Labels
ATIS	4,478	500	893	83
SNIPS	13,084	700	700	39
MovieTrivia	7,034	782	1,953	12
Movie	8,797	978	2,443	12
Restaurant	6,894	766	1,521	8
mTOP	15,667	2,235	4,386	75
mTOD	30,521	4,181	8,621	9

Table 2: Statistics per partition of the used datasets.

4.2 Experimental Details and Evaluation

We conduct experiments to compare our model with other parameter-efficient fine-tuning methods. Our first goal is to assess the advantages of combining different parameter-efficient techniques in a multi-task setting versus a single-task setting. The second goal is to evaluate our model against other hypernetwork-based approaches in a multi-task environment.

We compare our approaches with the UniPELT framework (Mao et al., 2022), HyperFormer model (Karimi Mahabadi et al., 2021b) and Hyperde-

[1]Downloaded from https://groups.csail.mit.edu/sls/

[2]Downloaded from https://github.com/MiuLab/SlotGated-SLU/tree/master/data

coder model (Ivison and Peters, 2022) described in section 2. We also compare our approach using Adapters and LoRA with T5 as the backbone model in a single-task setting. We found that the prefix-tuning and the gating mechanism proposed in the UniPELT framework decrease the performance using T5. Therefore, we only use Adapters and LoRA without a gating mechanism for our baseline comparison for the combination of UniPELT and T5. For all the implementations, we use the HuggingFace's Transformers (Wolf et al., 2020) and AdapterHub (Poth et al., 2023) libraries.

We use the batch sizes, learning rates and epochs reported in the original paper of each baseline to run the experiments in our work. For our proposed HyperLoader model, we used the same codebase[3] (under the Apache License, Version 2.0) and almost the same experimental details described by (Karimi Mahabadi et al., 2021b), that is, we employ the T5 base model, a batch size of 32, a constant learning rate of 3×10^{-4}, 2^{18} steps in all experiments, save a checkpoint every 1,000 steps, and sample tasks with conventional temperature-based with temperature $T = 10$ proportional to $p_\tau^{1/T}$, where $p_\tau = \frac{N_\tau}{\sum_{i=1}^{T} N_\tau}$ and N_τ is the number of training samples for the τ-th task. The only difference in our work is that we choose the best model based on the loss value instead of the average evaluation metric performance. We performed the experiments on 4 Nvidia A100 SXM 40G GPUs.

We use the micro-averaged F1-score as the evaluation metric for all experiments following the CoNLL convention (Tjong Kim Sang and De Meulder, 2003), where an entity is considered correct only if the entity is predicted exactly as it appears in the gold data. We use the SeqEval framework (Nakayama, 2018) to compute the scores in all the experiments.

5 Results

This section presents the results of the experiments we performed in this work described in section 4.2.

5.1 Full Dataset Performance

Table 3a shows the performance of our proposed approach and the baselines using the 100% of training and evaluation data. The UniPELT framework, Adapters and LoRA follow a single-task fine-tuning approach, therefore, the trainable parameters correspond to each dataset considered. For UniPELT, we report the results using BERT as in the original paper and T5 to compare the performance depending on the model's architecture. Initial experiments show that the use of the Prefix-tuning method and the gating mechanism decreases the performance of the encoder-decoder model; for this reason, we do not use these elements when applying UniPELT to T5.

Based on the results in Table 3a, it is possible to observe that our proposed HyperLoader model outperforms the baselines in three of the seven datasets and obtains the highest average with 0.8811, followed by the HyperFormer model (0.8795), UniPELT with T5 (0.8740), T5 with Adapters (0.8735), T5 with LoRA (0.8717), the Hyperdecoder model (0.8709) and UniPELT with BERT (0.8537). In terms of single-task and multi-task settings, it is possible to observe that the combination of UNiPELT with T5 outperforms the other models in two datasets (Movie with 0.8839, and ATIS with 0.9611); this result demonstrates that the use of hypernetworks improves the sharing information capacity of the models while reducing the negative transfer between tasks. Finally, when comparing the multi-task approaches based on hypernetworks, HyperLoader obtained a better average result than the HyperFormer model, which is the primary foundation of our work. In addition, since Hyperdecoder has more trainable parameters, the results indicate that the superior performance of our model is not only due to having more trainable parameters than the HyperFormer approach but also due to the combination of different parameter-efficient methods in a multi-task setting.

5.2 Low-resource Setting Results

We evaluate the performance of our proposed model and the baselines in a low-resource setting to measure its robustness with a data scarcity constraint, a common scenario in NLP (Hedderich et al., 2021). For this purpose, we down-sampled the training and validation partitions of all datasets to 10% and 20% of their original size and evaluated them in the full test subset. Table 3b shows the results of these experiments.

The first part of Table 3b contains the performance of the HyperLoader model and the baselines when only 10% of the training and validation data is used. It is evident that the combination of the UniPELT framework and the BERT model experiences the most significant performance decrease,

[3] https://github.com/rabeehk/hyperformer

Dataset	Adapters T5[222M/3.5M] ♣	LoRA T5[222M/0.84M] ♣	UniPELT BERT[110M/1.8M] ♣	UniPELT T5[220M/4.5M] ♣	HyperFormer T5[228M/5M]	HyperLoader T5[229M/6M]	Hyperdecoder T5[239M/16M]
MovieTriva	0.7184	0.7183	0.6873	0.7210	**0.7295**	0.7140	0.7072
Movie	0.8793	0.8709	0.8500	**0.8839**	0.8788	0.8745	0.8700
Restaurant	0.8069	0.8186	0.7689	0.8039	**0.8136**	0.7965	0.7891
ATIS	0.9581	0.9549	0.9446	**0.9611**	0.9595	0.9599	0.9519
SNIPS	0.9573	0.9506	0.9238	0.9478	0.9462	**0.9592**	0.9377
mTOP	0.8299	0.8254	0.8432	0.8374	0.8638	**0.8983**	0.8752
mTOD	0.9647	0.9633	0.9580	0.9627	0.9648	**0.9652**	0.9649
Average	0.8735	0.8717	0.8537	0.8768	0.8795	**0.8811**	0.8709

(a) Full dataset performance

Dataset	Adapters T5[222M/3.5M] ♣	LoRA T5[222M/0.84M] ♣	UniPELT BERT[110M/1.8M] ♣	UniPELT T5[220M/4.5M] ♣	HyperFormer T5[228M/5M]	HyperLoader T5[229M/6M]	Hyperdecoder T5[239M/16M]
10% of Training and validation data							
MovieTriva	0.6633	0.6695	0.5152	0.6717	0.6589	**0.6695**	0.6597
Movie	0.8281	0.8134	0.7809	0.8244	0.8311	**0.8368**	0.8202
Restaurant	0.7292	0.7171	0.5907	0.7049	0.7389	**0.7499**	0.7216
ATIS	0.9129	0.8329	0.5901	0.8919	0.9162	0.9172	**0.9193**
SNIPS	0.8647	0.8460	0.7398	0.8991	0.8943	**0.8970**	0.8727
mTOP	0.7559	0.6886	0.6048	0.7467	0.7976	**0.8114**	0.7760
mTOD	0.9459	0.9453	0.9351	0.9436	0.9467	**0.9500**	0.9436
Average	0.8143	0.7875	0.6795	0.8117	0.8262	**0.8331**	0.8162
20% of Training and validation data							
MovieTriva	0.6896	0.6934	0.6105	**0.7036**	0.6729	0.6863	0.6767
Movie	0.8492	0.8276	0.8047	0.8444	0.8407	**0.8546**	0.8377
Restaurant	**0.7699**	0.7522	0.7114	0.7562	0.7517	0.7601	0.7401
ATIS	0.9295	0.9222	0.8048	0.9281	0.9382	**0.9401**	0.9311
SNIPS	0.9250	0.9096	0.8474	0.9235	**0.9252**	0.9238	0.9044
mTOP	0.7798	0.7597	0.7052	0.7863	0.8399	**0.8474**	0.8140
mTOD	0.9424	0.9543	0.9433	0.9522	**0.9543**	0.9541	0.9529
Average	0.8408	0.8313	0.7753	0.8420	0.8461	**0.8523**	0.8367

(b) Low-resource setting results

Table 3: Performance of the HyperLoader model and baselines is shown for 100% (a), 10%, and 20% (b) of the training and evaluation data. Each method includes the model used and the total/trainable parameters in brackets. ♣ denotes a single-task fine-tuning approach, so trainable parameters are specific to each dataset. Bold fonts highlight the best result for each dataset and on average. Micro-averaged F1-score is reported.

from an average of 0.8537 to 0.6795. In contrast, the T5-based approaches show a smaller performance decrease in both single-task and multi-task settings. These results suggest that an encoder-decoder model is more capable of handling low-resource settings in both single-task and multi-task configurations. For the multi-task approaches, our HyperLoader model outperforms the HyperFormer and Hyperdecoder approaches in six out of seven datasets and on the average performance with a score of 0.8331, followed by HyperFormer (0.8262) and Hyperdecoder (0.8161).

The second part of Table 3b shows the results when 20% of the training and evaluation data is used. Using this amount of data our HyperLoader model outperforms the baselines in three of seven datasets and on the average performance. It is also noticeable that the combination of UniPELT and T5 using a single-task fine-tuning approach outperforms the other models only in the MovieTrivia dataset, with a micro-averaged F1-score of 0.7036 followed by HyperLoader with 0.6863, Hyperdecoder with 0.6767, HyperFormer with 0.6729 and finally UniPELT using BERT with 0.6105. These results suggest that combining various parameter-efficient methods in a multi-task setting and generating their weights based on the task, transformer layer, and the position where the parameter-efficient method is applied (Adapter layer or LoRA matrix) is effective for Sequence Labelling tasks, leading to good performance with a parameter-efficient approach. Figure 2 shows the relationship between average performance and the percentage of trainable parameters per model. The symbol ♣ denotes a single fine-tuning approach. Our approach consistently outperforms other methods across 10%, 20%, and 100% of the training and validation data. This evidence shows that combining various parameter-efficient fine-tuning techniques in a multi-task setting is effective for low-resource scenarios. The performance boost is not just due to adding more parameters, as seen with the hyperdecoder model and the combination of UniPELT with BERT, which have a higher percentage of trainable parameters. Therefore, the combination of parameter-efficient methods and weight generation is a potent alternative for Sequence Labelling tasks.

5.3 Ablation Experiments

We conducted ablation experiments to assess the effect of varying the number of trainable parameters in our model. Table 4 presents the results in descending order of performance. The first row (in blue) corresponds to the final version of our model,

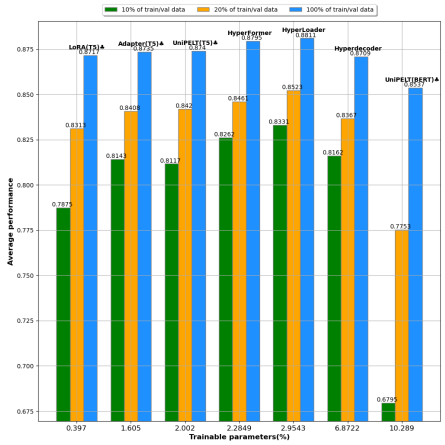

Figure 2: Average performance-percentage of trainable parameters plot using different portions of the datasets. ♣ indicates a single-task fine-tuning approach.

Variant	Total/trainable params	Avg score
HyperLoader	229M/6M	**0.8811**
+ 4 LoRA HNs	228M/8M	0.0808
+ 8 LoRA HNs	234M/11M	0.800
+ 2 LoRA HNs, r=16	229M/6M	0.8804
HyperFormer	228M/5M	0.8795
- Adapters	225M/1.8M	0.8794

Table 4: Average performance of the ablation experiments. We denote hypernetworks as HNs.

as described in section 3. The fifth row (in grey) corresponds to the HyperFormer model, which is the main foundation of our work.

The initial experiments (denoted by +) involved increasing the number of trainable parameters for generating LoRA's weight matrices. In the case of using four hypernetworks, the first one generates the Low-rank A matrix for Query (Q) and Value (V) in the self-attention module, and the second generates the Low-rank B matrix for Q and V in the self-attention module. Analogously, the third and fourth hypernetworks generate the A and B matrices in the cross-attention module of the decoder. In the case of 8 hypernetworks, each generates the weight of only one component: one for the A and one for the B matrices in the Query; another two hypernetworks for the A and B matrices in the Value of the self-attention. The remaining four hypernetworks are analogous to the former but in the cross-attention component of the decoder. Finally, the last experiment involves increasing the reduction factor ($r = 16$) of the LoRA matrices in the final version of the HyperLoader model. These results indicate that our model achieves a strong balance between performance and the percentage of trainable parameters (2.9543%), with only a 0.6694% increase compared to the HyperFormer model (2.3849%). It also shows that using only 2 hypernetworks for the LoRA matrices enables the model to generate better weights based on the task, the parameter-efficient method and its position within the Transformer layer of the model.

The second experiment involved removing (denoted by $-$) the Adapters from the model and adjusting only the task-conditioned LoRA matrices during training. This version showed the lowest average performance, though the difference from the HyperFormer model was minimal (0.0001). This suggests that while a single parameter-efficient technique can effectively address different tasks, combining such methods provides better results in a multi-task setting and shows that further research into this area is necessary.

6 Conclusions

We proposed a method that combines parameter-efficient methods in a multi-task setting. Using a hypernetwork, we generate the weights of the adapter layers and LoRA matrices conditioned on the task, the transformer layer of the model, and the position of these modules within this layer. Our method outperforms previous work in single-task and multi-task fine-tuning that combines different parameter-efficient methods and hypernetwork-based approaches using the full training and validation splits of each dataset and in a low-resource configuration using only 10% and 20% of those partitions to train the model. We provide empirical evidence that the improvement in performance is not only because of augmenting the trainable parameters since the hyperdecoder model has the largest number of trainable weights. Therefore, the combination of parameter-efficient methods and weight generation is a strong alternative to solving Sequence Labelling tasks in a multi-task setting.

7 Limitations and Future work

Our model excels in multi-task settings with a parameter-efficient fine-tuning approach that mitigates negative transfer, underfitting, and overfitting. However, it still requires access to all datasets during training and needs complete retraining when a new task is added. Curriculum learning (Bengio et al., 2009; Wang et al., 2021; Soviany et al., 2022; Piergiovanni et al., 2023) could address this limitation by enhancing learning efficiency, potentially leading to faster convergence and improved performance.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Xuxin Cheng, Zhihong Zhu, Bowen Cao, Qichen Ye, and Yuexian Zou. 2023a. MRRL: Modifying the reference via reinforcement learning for non-autoregressive joint multiple intent detection and slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10495–10505, Singapore. Association for Computational Linguistics.

Xuxin Cheng, Zhihong Zhu, Wanshi Xu, Yaowei Li, Hongxiang Li, and Yuexian Zou. 2023b. Accelerating multiple intent detection and slot filling via targeted knowledge distillation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8900–8910, Singapore. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Marco Farina, Duccio Pappadopulo, Anant Gupta, Leslie Huang, Ozan İrsoy, and Thamar Solorio. 2023. Distillation of encoder-decoder transformers for sequence labelling. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2539–2549.

Mauajama Firdaus, Asif Ekbal, and Erik Cambria. 2023. Multitask learning for multilingual intent detection and slot filling in dialogue systems. *Information Fusion*, 91:299–315.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, Online. Association for Computational Linguistics.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

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. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, Kirk Roberts, and Hua Xu. 2024. Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association*, page ocad259.

Hamish Ivison and Matthew Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1715–1730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 1022–1035. Curran Associates, Inc.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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 565–576, Online. Association for Computational Linguistics.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented

semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland. Association for Computational Linguistics.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

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.

AJ Piergiovanni, Weicheng Kuo, Wei Li, and Anelia Angelova. 2023. Dynamic pre-training of vision-language models. In *Workshop on Multimodal Representation Learning. ICLR 2023*.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160, Singapore. 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.

Karthik Raman, Iftekhar Naim, Jiecao Chen, Kazuma Hashimoto, Kiran Yalasangi, and Krishna Srinivasan. 2022. Transforming sequence tagging into a seq2seq task. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11856–11874.

Shaina Raza, Deepak John Reji, Femi Shajan, and Syed Raza Bashir. 2022. Large-scale application of named entity recognition to biomedicine and epidemiology. *PLOS Digital Health*, 1(12):e0000152.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. 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 3795–3805, Minneapolis, Minnesota. Association for Computational Linguistics.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Fernando Trias, Hongming Wang, Sylvain Jaume, and Stratos Idreos. 2021. Named entity recognition in historic legal text: A transformer and state machine ensemble method. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 172–179, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

Zirui Wang, Zihang Dai, Barnabas Poczos, and Jaime Carbonell. 2019. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. 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.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2023. Universalner: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*.

A Corpora labels

Table 5 shows each Sequence Labelling dataset's complete list of labels used to train and evaluate the HyperLoader model. Based on the labels' names, we calculate label overlap between the datasets. Nine different labels are shared across the datasets, representing 3.78% of the total labels.

ATIS labels			
fromloc.airport_name	arrive_time.start_time	flight_number	cost_relative
connect	flight_days	restriction_code	depart_date.date_relative
return_date.month_name	mod	arrive_date.month_name	city_name
depart_date.day_number	compartment	depart_time.start_time	airline_name
meal	depart_date.month_name	time_relative	return_date.today_relative
depart_time.period_mod	flight_mod	airport_name	stoploc.airport_code
depart_date.year	fare_basis_code	today_relative	airport_code
fromloc.state_name	toloc.city_name	economy	booking_class
arrive_date.today_relative	arrive_date.date_relative	toloc.airport_code	fromloc.airport_code
day_number	stoploc.city_name	state_code	month_name
arrive_date.day_name	arrive_time.period_of_day	state_name	aircraft_code
period_of_day	return_time.period_mod	day_name	stoploc.state_code
toloc.state_code	depart_time.time_relative	toloc.airport_name	return_date.date_relative
fromloc.city_name	return_date.day_number	depart_time.time	depart_date.day_name
arrive_time.time	meal_code	or	class_type
return_date.day_name	time	toloc.state_name	arrive_date.day_number
days_code	arrive_time.period_mod	arrive_time.time_relative	flight_stop
depart_time.period_of_day	transport_type	round_trip	meal_description
fare_amount	toloc.country_name	arrive_time.end_time	depart_time.end_time
flight	fromloc.state_code	depart_date.today_relative	flight_time
airline_code	return_time.period_of_day	stoploc.airport_name	
mTOP labels			
school	contact_related	recipes_diet	news_source
todo	recipes_unit_nutrition	person_reminded	attendee
recipes_source	date_time	news_topic	music_album_title
life_event	contact_method	recipes_time_preparation	type_contact
news_reference	similarity	name_app	recipes_qualifier_nutrition
recipes_cooking_method	timer_name	contact_removed	employer
recipes_excluded_ingredient	method_recipes	type_relation	group
news_type	content_exact	ordinal	news_category
recipes_unit_measurement	user_attendee_event	music_playlist_title	sender
music_provider_name	recipes_dish	location	amount
music_rewind_time	music_type	alarm_name	weather_temperature_unit
music_album_modifier	category_event	education_degree	recipes_meal
period	music_artist_name	music_radio_id	method_timer
recipes_cuisine	phone_number	music_genre	weather_attribute
music_track_title	recipes_included_ingredient	attribute_event	method_retrieval_reminder
major	contact_added	recipes_rating	contact
gender	age	recipient	job
recipes_type	recipes_type_nutrition	attendee_event	recipes_attribute
music_playlist_modifier	title_event	type_content	
SNIPS labels			
object_location_type	poi	state	genre
object_name	album	playlist	movie_type
movie_name	artist	restaurant_name	restaurant_type
spatial_relation	timerange	object_part_of_series_type	current_location
served_dish	city	object_select	music_item
country	cuisine	geographic_poi	sort
condition_temperature	object_type	party_size_description	service
track	entity_name	party_size_number	rating_value
playlist_owner	condition_description	rating_unit	location_name
year	facility	best_rating	
Movie labels			
genre	rating	year	plot
ratings_average	director	song	title
trailer	review	actor	character
MovieTrivia labels			
award	relationship	quote	genre
character_name	director	plot	year
soundtrack	origin	actor	opinion
mTOD labels			
demonstrative_reference	datetime	weather	negation
alarm	news	timer	reminder
location			
Restaurant labels			
restaurant_name	cuisine	rating	price
dish	hours	amenity	location

Table 5: List of labels for each used Sequence labelling dataset to evaluate our proposed approach.