Attention Fusion: a light yet efficient late fusion mechanism for task adaptation in NLU

Jin Cao*, **Chandana Satya Prakash***, **Wael Hamza**
Amazon AI
jincao, chanprak, waelhamz@amazon.com

Abstract

Fine-tuning a pre-trained language model using annotated data has become the de-facto standard for adapting general-purpose pre-trained models like BERT to downstream tasks. However, given the trend of larger pre-trained models, fine-tuning these models for each downstream task is parameter-inefficient and computationally-expensive deeming this approach sub-optimal for adoption by NLU systems. In recent years, various approaches have been proposed for parameter efficient task adaptation such as Adaptor, Bitfit, Prompt tuning, Prefix tuning etc. However, most of these efforts propose to insert task specific parameters in-between or inside intermediate layers of the pre-trained encoder resulting in higher computational cost due to back-propagation of errors to all layers. To mitigate this issue, we propose a light but efficient, attention based fusion module which computes task-attuned token representations by aggregating intermediate layer representations from a pre-trained network. Our proposed fusion module trains only 0.0009% of total parameters and achieves competitive performance to the standard fine-tuning approach on various tasks. It is also decoupled from the pre-trained network making it efficient during computation and scalable during deployment. Last but not the least, we demonstrate that our proposed attention-fusion mechanism can transfer effectively to different languages for further re-use and expansion.

1 Introduction

Aligned with recent advancements in deep learning research, most state-of-the-art (SOTA) NLU models are built upon neural networks, especially using transformer(Vaswani et al., 2017) based architectures. However, these models require a large amount of domain-specific labeled examples for training, which is prohibitively expensive. The recent adoption of self-supervised pre-training and transfer learning mitigates the issues stemming from scarcity of labeled data (Yang et al., 2017; Chen et al., 2019), by pre-training with unsupervised tasks established upon massive unlabeled corpora (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019; Raffel et al., 2020). The resulting models encode syntactic and semantic linguistic information and can be fine-tuned with limited labeled examples on downstream NLU tasks, such as Question-answering (QA) (Rajpurkar et al.), Textual-entailment (Dagan et al., 2006), Slot Labeling (SL) (Chen et al., 2019) etc. Fine-tuning is a commonly used method that adapts a pre-trained model to a downstream task and has been shown to achieve SOTA results in various NLU tasks. However, in the presence of larger pre-trained models and many downstream tasks, fine-tuning the whole model for each downstream task is inefficient and expensive due to reasons such as higher memory consumption since gradient and optimizer states need to be stored for all parameters, higher computational cost since error needs to be back-propagated through all layers and bigger cost of hosting large models for each task.

Parameter efficient domain adaptation has been an area of interest in recent literature comprising of various approaches such as Adaptor (Houlsby et al., 2019), Bitfit (Ben-Zaken et al., 2021), DiffPrune (Guo et al., 2020), Prompt tuning (Lester et al., 2021; Liu et al., 2021) etc. Most of these efforts propose to insert or append task specific parameters in-between or inside of the pre-trained encoder layers, we refer these approaches as early-fusion techniques, as task specific parameters are fused inside the pre-trained network. Some drawbacks of early fusion based methods are: during training, loss has to be back-propagated to all layers making them slower; hard to scale in NLU systems as pre-trained encoder and task specific modules are

*equal contribution

Findings of the Association for Computational Linguistics: NAACL 2022, pages 857 - 866
July 10-15, 2022 ©2022 Association for Computational Linguistics

tightly coupled together. In comparison to early fusion, one can place task specific modules after the pre-trained network, so the pre-trained network is untouched regardless of downstream tasks, we refer to this as late-fusion. One late-fusion option is to concatenate (Cao et al., 2020) all layers from a pre-trained network and project to a lower dimension for task-specific decoders. However, the projection matrices can be considerably big with larger models, e.g., concatenation then projecting hidden layers of a BERT-large model to a dimension of 256 amounts to 6.2 million parameters leading to increased computational cost. These challenges hinder the progress of deploying SOTA transfer learning based models to downstream NLU systems.

To address these challenges, we propose attention-fusion, a light but effective task-specific late-fusion based module, for adapting pre-trained models to downstream NLU tasks. Our proposed architecture decouples general purpose pre-trained models from downstream task-specific decoder layers with an attention-fusion module. The fusion module enables decoders to effectively adapt hidden representations from intermediate layers of the pre-trained network.

To examine the effectiveness of attention-fusion mechanism, we conduct experiments on popular language understanding tasks, including QQP (Quora Question Pair), QNLI (Question-answering NLI), SST-2 (Stanford Sentiment Treebank), CONLL-03 (Name Entity Recognition) and a multilingual Spoken Language Understanding (SLU) (Tür et al., 2002; Huang and Chen, 2019) task using mATIS dataset. Our results demonstrate that attention-fusion module achieves comparable performance to fine-tuning approach while only tuning a small amount of parameters. Our attention-fusion approach is a late-fusion based mechanism, thus, exhibiting lower computation cost since back-propagation is limited to task-specific fusion module and decoder layers. Furthermore, we empirically show that the task-specific attention-fusion module is transferable across languages. We aim to release our code on Github to support further experimentation. In summary, our primary contributions are three-fold:

- Propose a light but efficient task-specific late-fusion module called attention-fusion, which is capable of aggregating representations from intermediate layers of the pre-trained model to adapt to a downstream NLU task.
- Demonstrate the benefit of the proposed module by evaluating both accuracy and computation efficiency on various tasks.
- Analyze how the attention-fusion module interacts with pre-trained models and show that such a module is task-specific and can transfer effectively to different languages.

2 Related Work

The importance of efficiently fine-tuning and deploying pre-trained networks to NLU systems has gained wider recognition. In this section, we discuss various approaches proposed in literature.

Model Compression and Distillation One research direction focuses on building compact pre-trained networks with techniques like model compression (Bucilua et al., 2006; Ganesh et al., 2020), pruning (Gordon et al., 2020; Han et al., 2016; Wang et al., 2019b), quantization and knowledge distillation (Hinton et al., 2015). DistilBERT (Sanh et al., 2019) and TinyBERT (Jiao et al., 2020) suggested using knowledge distillation framework to train a smaller student network by matching the layer outputs with a larger teacher model. ALBERT (Lan et al., 2019) attempted to reduce parameters through weight-sharing across all transformer layers and factorizing the embedding matrix. Zafrir et al. (2019) applied an 8-bit integer quantization to reduce BERT model size by 4x. However, these approaches still suffer from sub-optimal performance in accuracy when the model size gets smaller.

Lightweight Fine-tuning Another line of research focuses on using a small amount of extra parameters along with the pre-trained network. Some popular methods include: Adaptor, proposed by Houlsby et al. (2019), suggested to insert a task specific bottleneck module between pre-trained network layers. Other ideas suggest to re-parameterize the pre-trained network partially: Ben-Zaken et al. (2021) proposed to tune only bias-terms of the pre-trained network for each task; Guo et al. (2020) formulates task-specific fine-tuning as learning a diff vector that is added to the pre-trained network, both shown to match the full fine-tuning approach on accuracy while only using less than 0.1% trainable parameters; more recently, Lester et al. (2021); Liu et al. (2021) suggested appending extra prompt tokens to layers of the model to control output while keeping the network frozen. There are also

efforts focusing on using intermediate layers of the pre-trained network for different tasks, Peters et al. (2018) proposed to learn a weighted sum representation from the intermediate layers of the model , while Cao et al. (2020) suggested concatenating the intermediate layers of the pre-trained network.

Probing in Transformers The significant performance gain brought about by pre-training has emphasized the need to better understand the co-relation between pre-trained network architecture and resulting language representations. Studies (Tenney et al., 2019b,a; Kovaleva et al., 2019) suggest that pre-trained models can encode a range of syntactic and semantic information in different layers of the network. Complex linguistic structures are represented hierarchically in the higher layers of the model. In comparison, simple language clues are encoded in lower layers. Inspired by these findings, we propose a task-specific attention-fusion architecture, to more effectively utilize hidden representations with different granularity from pre-trained networks.

3 Approach

In this section, we describe the proposed task-specific fusion model architecture, which augments general-purpose pre-trained models with task-dependent attention on encoded representations in multi-granularity and with prediction layers for NLU tasks. The attention-fusion module aims to improve performance and parameter efficiency by sharing parameters of the pre-trained model with other tasks.

3.1 Late-Fusion vs. Early-Fusion

Most of the existing methods adopt early-fusion for task adaptation, by either inserting light-weight task-specific module or appending prompts inside the pre-trained encoder; or tuning only a small portion of the parameters from the pre-trained network to adapt to a downstream task. With early-fusion, the light-weight module can take advantage of the depth of the pre-trained network, to adjust model output to downstream task. One drawback of early-fusion based approaches is that the task specific module is tightly coupled with the pre-trained network, making the training process costly and slow. Early-fusion based methods require loss/error to be back-propagated to all layers of the pre-trained network since light tune-able modules reside in each layer of the network. One solution to address this inefficiency is to adopt late-fusion, where the entire pre-trained network is kept frozen, and the task-specific module is placed after the pre-trained encoder. In such a setting, pre-trained network is de-coupled from the downstream task, training loss is only back-propagated to task-specific parameters i.e attention-fusion module and decoder layers, make the training process more efficient.

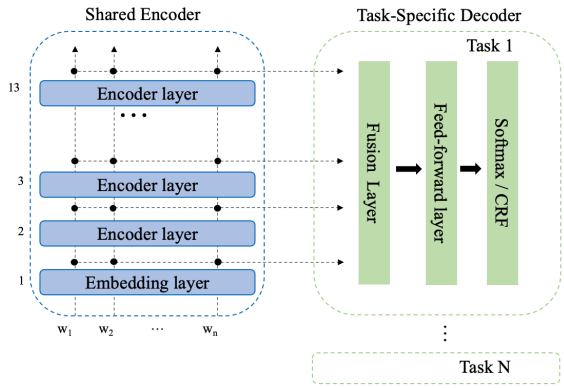

Figure 1: The architecture of the proposed task-specific attention-fusion module. It also depicts the shareability of a pre-trained network among different tasks.

3.2 Model Architecture

We propose attention fusion, a late-fusion module to utilize hidden representations from a shareable, general-purpose pre-trained model, such as BERT (Devlin et al., 2019) and RoBERTa(Liu et al., 2019), for downstream tasks adaptation. Figure 1 visualizes the proposed architecture, including the fusion layer and how it fits into a pre-trained network (in our case, BERT-large or multi-lingual BERT). The parameters in the pre-trained network (including the embedding layer), colored in blue, are frozen, making the pre-trained model shareable with other tasks. The pre-trained encoder takes a sequence of tokens as input and generates an encoded representation for every token at each layer. The task specific decoding network, colored in green, is tailored per downstream task. For NLU tasks, decoders typically include a feed-forward layer and a softmax layer for the final output. The attention fusion module, also colored in green, connects the pre-trained encoder to the task-specific decoder. This fusion module is used to extract useful features from the intermediate (and final) layers of the pre-trained encoder for the task-specific decoder. During training, errors are propagated only to the task-specific parts (the green components in the figure) hence it is memory and computation efficient.

The fusion module is also extremely light weight, for a BERT-large model, the attention fusion module only adds 0.0009% of the total parameters of pre-trained encoder.

The proposed approach is scalable when the system grows to include more tasks, as the most computation and memory-intensive component, the encoder, is shared and frozen, while the task-specific decoder and attention fusion module is decoupled from the pre-trained network and trained on each downstream task. Moreover, with the addition of a fusion module, we can achieve comparable performance to fine-tuning, without adapting the pre-trained network by surfacing the pertinent information already encoded and buried in the intermediate layers, for different downstream tasks.

3.3 Attention-Fusion Module

The encoded representation of a token is achieved by focusing on different layers of a pre-trained network for a given downstream task. The focus on different layers of the network shifts based on the task at hand. To attend to the corresponding token-level representation across different layers for a given downstream task, we propose an attention-fusion module to learn task-specific token representation, by pooling intermediate layer representations at a token level. More specifically, for each task, we use an attention query vector, denoted as Q^t. This query vector is a task-specific representation which can be either learned during training or adopted from a pre-trained one (learned on the same task but same/different datasets). We further denote the representation of token i at layer j as V_i^j and the attention weight of token i at layer j for task t as $\alpha_i^j(t)$, which can be calculated as:

$$\alpha_i^j(t) = \frac{exp(Q^t V_i^j)}{\sum_k exp(Q^t V_i^k)} \quad (1)$$

$$c_i(t) = \sum_{j=1} \alpha_i^j(t) V_i^j \quad (2)$$

Thus, the contextual representation of token i for task t, denoted as $c_i(t)$, can be calculated as weighted sum of token i across all vertical layers. We denote such attention-based pooling mechanism as attention-fusion in our experiments. The re-computed fused token representation is then projected and connected to feed-forward layers and final softmax layer.

There are other ways to extract token representations from an encoder. Peters et al. (2018) proposed all layers be combined with a weighted average pooling operation, $c_k = \sum_{j=0}^{L} s_j h_{k,j}$. The weight vector is optimized as part of the task model, so that it may preferentially mix contextual information represented in different layers of the model for the task. We refer to this approach as linear-fusion. Cao et al. (2020) proposed to concatenate all BERT layers, then project to a feed forward layer before passing to decoders. We refer to this approach as concat-fusion. For comparison purposes, we add linear-fusion and concat-fusion as our baseline alternatives for late-fusion approaches.

4 Experiment

We evaluate attention-fusion as well as other popular light-weight fine-tuning approaches on 5 popular general language understanding tasks.

4.1 Dataset

Sentiment Analysis: We use the SST-2 dataset belonging to GLUE benchmark (Wang et al., 2019a) to perform a single sentence binary classification task. We report accuracy on the developement set.

Paraphrase Similarity: We use the QQP dataset belonging to GLUE benchmark to perform a sentence-pair binary classification task. We report accuracy on the developement set.

Natural Language Inference: We use the QNLI dataset belonging to GLUE benchmark to perform a sentence-pair multi-class classification task. We report accuracy on the developement set.

Named Entity Recognition: We use the CoNLL-2003 dataset which is a widely adopted NER benchmark(Tjong Kim Sang and De Meulder, 2003). We report micro-f1 score on the test set.

Spoken Language Understanding: We use the public mATIS (Mansour and Batool, 2020) dataset to perform Intent Classification (IC) and Slot Labeling (SL) tasks. The dataset is originally transcribed in English and then manually translated into four languages: EN, FR, DE and ES; thus, the ontology of the data in all 4 languages is the same. We report micro F1 score on the test set.

4.2 Experiment Setup

For monolingual tasks, we compare the proposed attention-fusion module on four popular general language understanding tasks: QQP, QNLI, SST-2 and CONLL-03 with:

Table 1: Results on monolingual English tasks comparing attention-fusion module against various light weight fine-tuning approaches. We report accuracy metric on each of these tasks (higher scores indicate better). Additionally, we also present the percentage of trainable parameters for each of these approaches as well as fusion type. Score shown in bold with underscore indicates best score across all, while bold font indicates best score among light-weight fine-tuning approaches. * indicates accuracy from dev set. For results on Bitfit, Diff-Prune, Prompt-tuning v1 and v2, we quote the numbers listed in the paper if available, otherwise, we produce the numbers using their code and settings.

Model	**Fusion type**	**% params**	**QQP***	**QNLI***	**SST-2***	**CONLL-03**	**AVG**
Fine-tune	None	100%	**90.2**	91.5	93.4	**92.8**	**92.0**
Last-layer	None	0%	75.9	60.3	83.0	88.3	76.9
Adaptor	Early	3.6%	87.1	91.8	91.9	89.1	90.0
Bitfit	Early	0.08%	85.6	91.8	93.3	89.5	90.1
Diff-Prune	Early	0.1%	85.2	**92.7**	93.3	90.0	90.3
Prompt-tuning	Early	0.015[1]%	80.0	85.7	92.4	81.9	85.0
Prompt-tuning v2	Early	0.36[1]%	86.6	91.0	**93.6**	90.2	**90.4**
Linear-fusion	Late	0.000008%	78.4	72.1	85.7	90.6	81.8
Concat-fusion	Late	0%[2]	87.6	88.4	92.3	90.1	89.6
Attention-fusion	Late	0.0009%	**87.9**	88.2	93.3	**90.9**	90.1

Table 2: Results on public mATIS datasest for IC and SL tasks using the proposed task specific attention-fusion architecture and baseline models. All models are trained on full size training data in four languages. Results are F1 scores (higher scores indicate better results). Score shown in bold with underscore indicates best score across all, while bold font indicates best score among light-weight fine-tuning approaches.

Model	**Intent Classification**					**Slot Labeling**				
	EN	**ES**	**DE**	**FR**	**AVG**	**EN**	**ES**	**DE**	**FR**	**AVG**
Fine-tune	97.46	96.15	**96.60**	96.63	96.71	**96.94**	91.52	96.41	**95.20**	95.02
Last-layer	91.60	88.45	90.10	91.57	90.43	93.22	88.73	92.34	91.84	91.54
Linear-fusion	94.69	95.64	94.30	94.18	94.70	95.23	91.36	96.35	94.78	94.43
Concat-fusion	**97.76**	**96.68**	96.24	**97.85**	**97.13**	96.53	91.61	96.21	94.69	94.76
Attention-fusion	97.39	96.39	**96.30**	97.22	96.83	**96.91**	**92.07**	**96.54**	**95.09**	**95.15**

- two baselines: a standard fine-tuning mechanism that trains the entire network on the downstream task (denoted as fine-tune), and the case where the pre-trained encoder is kept frozen and only the parameters in decoder layers are fine-tuned (denoted as last-layer)
- different late-fusion mechanisms such as concat-fusion and ELMo style linear-fusion.
- light-weight fine-tuning techniques proposed in literature such as Adaptor, Bitfit, Diff-prune, Prompt-tuning v1 and v2.

We evaluate the multilingual and cross-lingual capabilities of attention fusion module on IC and SL tasks using mATIS dataset under two learning regimes: using full-sized training data and few-shot learning.

For monolingual tasks, we use a BERT-large encoder from public available gluonnlp; for multilingual tasks, we use an in-house pre-trained mBERT base encoder trained on 8 languages. Both pre-trained models are trained on public data, including Wikipedia, Books corpus, and CommonCrawl corpus. For decoders, we use two feed forward layers of hidden size 256, and a softmax layer for sequence classification task, a CRF layer followed by a softmax layer for sequence labeling tasks.

Throughout our experiments, we train all models (baselines and variations of fusions) with mini-batch sizes ranging between 16 to 64, on 2 Nvidia Tesla V100 GPUs. We adopt Adam optimizer for all our experiments and use a learning rate of 2e-5 for the fine-tune baseline and 2e-3 for other late-fusion models. For all experiments, we report mean statistic of 3 random seeds run.

[1]Optimal prompt length usually ranges from 20 to 100 tokens, we use an average of 50 tokens to estimate number of extra parameters.

[2]Although concat-fusion doesn't insert additional parameters explicitly, it does increase the size of the overall

Table 3: Results on public mATIS dataset evaluating the attention-fusion architecture in few-shot transfer learning setting for IC and SL tasks. Comparing to last-layer and fine-tuned mBERT baselines. Numbers in the table are F1 scores on IC and SL, averaged across target languages.

	Intent Classification			**Slot Labeling**		
source → target	**last-layer**	**fine-tune**	**attention-fusion**	**last-layer**	**fine-tune**	**attention-fusion**
EN→DE / ES / FR	84.67	**97.06**	96.09	89.06	**95.72**	94.48
DE→EN / ES / FR	85.38	**96.76**	96.14	87.39	**94.72**	92.96
FR→DE / ES / EN	82.78	**96.99**	96.73	86.55	**92.62**	91.75
ES→DE / EN / FR	80.89	**96.90**	96.76	84.72	**92.48**	90.16

5 Results and Discussion

In this section, we present the results on monolingual and multilingual tasks, compare the training efficiency and analyze the interactions of attention fusion with different layers across different tasks.

5.1 Evaluation on Monolingual Tasks

Table 1 compares the performance of attention-fusion module with various light-weight fine-tuning approaches proposed in literature on QQP, QNLI, SST-2 and CONLL-03 datasets. We quoted numbers on Bitfit, DiffPrune, Prompt-tuning and Prompt-tuning v2 from their published results, while for Adaptor, we reproduced results using code and settings suggested by the authors since their published numbers are on test set only. For late-fusion baselines, we compare against ELMo style linear-fusion and concat-fusion. We also record the percentage of additional trainable parameters to demonstrate the computational cost associated with each approach. Across all tasks, the attention-fusion module sees a significant improvement of an average 13.2 absolute points compared to the last-layer baseline model which indicates the need to harness intermediate layers representation of a network for a downstream task. Among the different lightweight fine-tuning mechanisms, attention-fusion achieves the best performance on QQP and CONLL-03 tasks, comparable performance on SST-2, while seeing a degradation of 4 points on QNLI, we hypothesis NLI task requires some hierarchical mapping of semantic features from representation space, thus limited performance of late-fusion based methods on such a task. On an average, attention-fusion is 1.9 absolute points behind the fine-tune approach and achieves comparable performance against other early-fusion approaches while only training a small fraction (0.0009%) of parameters. Among late fusion methods, ELMo style linear-fusion is behind attention fusion by 8.3 absolute points, with significant degradation in pair-utterance tasks like QQP and QNLI, indicating that using identical weight assignment for all tokens in the sequence is a sub-optimal approach and requires a more flexible and nuanced fusion mechanism. Concat-fusion shows comparable performance to attention-fusion but it is not as light-weight as attention-fusion due to extra parameters being added to project the concatenated layers to the downstream decoders.

5.2 Evaluation on Multilingual Tasks

Table 2 shows the performance of mBERT models on IC and SL tasks with different types of late-fusion modules applied, along with fine-tune and last-layer baselines on four languages in the mATIS dataset. We observe a significant improvement in performance for both IC and SL tasks across all languages (an average of 6.4 absolute F1 score increase in IC and 3.6 absolute F1 score increase in SL) using attention-fusion compared to the last-layer model. Attention-fusion also achieves comparable performance to fine-tune model in both IC and SL tasks (an average of 0.12 absolute F1 score increase in IC and 0.13 absolute F1 score increase in SL). Linear-fusion and concat-fusion also outperforms the last-layer, demonstrating the effectiveness of utilizing intermediate layer representations of the encoder.

With the rising popularity of NLU systems, there is a need to expand them to new languages. An open challenge with language expansion is the scarcity of annotated data in the new language. A popular way to tackle this challenge is through transfer learning; thus, we examine the language

model. When the concatenated hidden representations are utilized by the decoder, it increases the size of FFN layers in the decoder, for a BERT-large model with 1024 hidden size and 24 layers, the first FFN layer in the decoder is $(1024 * 24) * decoder\_hidden\_units$ which is 24x the size of FFN layer in other approaches.

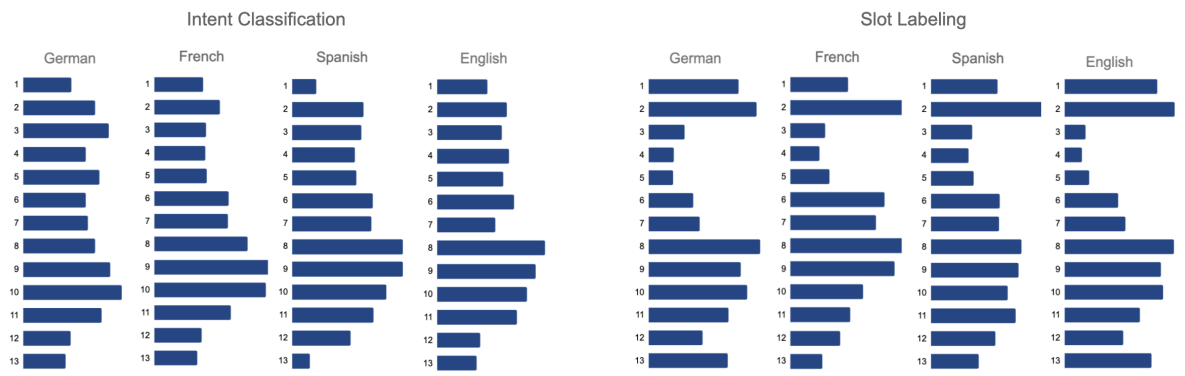

Figure 2: The distribution of attention weights for Intent Classification and Slot Labeling tasks on EN, ES, DE and FR languages of mATIS dataset.

transfer-ability of our proposed architecture in a few-shot learning setting compared to the baseline models. We measure F1 score on IC and SL tasks on four languages (EN, FR, DE, and ES) for the public mATIS dataset. We create a few-shot dataset for each one of the four languages by down-sampling the original training dataset to 5% of the original size. Table 3 summarizes the IC and SL performance of models fine-tuned on full-sized training data in the source language plus few-shot data in the target language, and evaluated on the target language. For example, EN -> FR indicates a model fine-tuned on full-size EN data and few-shot FR data, and evaluated on the FR dataset. We observe attention-fusion improves transfer-ability over last-layer baseline by a large margin (an average increase of 13 points F1 score for IC and 5.4 points F1 score for SL), achieving comparable performance with the fine-tune model on IC. Even though the F1 scores are lower than that of fine-tune model for SL in DE and ES, using attention-fusion allows for language expansion at a significantly lower cost, compared to the fine-tune baseline. The overall result suggests that attention-fusion can effectively improve knowledge transfer across languages.

5.3 Training Efficiency

In this section, we examine the training efficiency through number of trainable parameters and convergence speed of attention-fusion in comparison with other methods. A popular approach is Adapter in which the number of extra parameters is $num\_layers \times (2 \times m \times d + m + d)$, for BERT-large, m=1024, num_layers=24, with bottleneck dim of 256, results in 12.6 million trainable parameters. Bitfit adjusts the parameters of bias terms

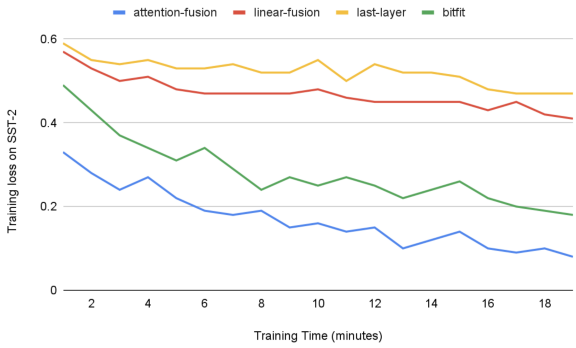

Figure 3: The loss convergence speed of attention-fusion mechanism against other early and late fusion approaches on SST-2 task.

in Query, Key, Value matrices, as well as projection and feed-forward module among all layers, for BERT-large, this amounting to 270 K trainable parameters. Prompt-tuning appends prompts to the first layer or to all layers (Liu et al., 2021), the preferred length of the prompt ranges from 20 to 100 tokens, for BERT-large, the trainable parameters is $(1024 \times prompt\ length)$ if prompt is inserted into the first layer, or $(1024 \times prompt\ length \times num\ layers)$ when inserted to all layers. In comparison, the attention-fusion mechanism is more efficient than other early-fusion based methods due to the following reasons:

- it uses a query vector of the same size as the pre-trained encoder hidden dimension (e.g., 1024 parameters for BERT-large) which adds significantly fewer parameters for training compared to early-fusion mechanisms
- the size of the fusion module does not grow with the number of layers in the pre-trained network unlike some other approaches

Table 4: Cross-lingual transfer: IC and SL results with pre-trained attention-fusion from EN on DE/ES/FR dataset. For baseline we train attention fusion with in-target language data. We report F1 scores on IC and SL tasks. Positive numbers indicate improvement over baseline.

Cross-lingual transfer	**Intent Classification**				**Slot Labeling**			
	DE	**ES**	**FR**	**AVG**	**DE**	**ES**	**FR**	**AVG**
in-target language fine-tuning (baseline)	95.96	96.68	97.18	**96.52**	95.58	91.06	94.79	94.48
in-target language fine-tuning with pre-trained fusion vector trained on EN	94.78	96.10	96.55	96.44	95.54	91.13	94.79	**94.48**

- it uses a late-fusion mechanism, in which the backward-pass and model update are performed only to the fusion module and decoder layers resulting in faster training.

In Figure 3, we report training loss over time for attention fusion and some other methods. We chose BitFit to represent an early-fusion approach due to its simplicity and effectiveness, and linear-fusion as an alternate late-fusion mechanism. We train all models with same number of GPUs and batch size, as well as adopt learning rate suggested by published paper. As shown in the plot, attention-fusion converges faster than BitFit, we hypothesis this is because late fusion does not need to back-propagate loss to all layers; hence making it faster to train and converge. last-layer is the least performant given it doesn't harness intermediate representations; on the other hand, despite using intermediate layers, linear-fusion does not perform as good as attention fusion, suggesting the effectiveness and efficiency of the task-specific attention mechanism.[3]

5.4 Analysis on Attention-fusion

In this section, we analyse the role and nature of the attention-fusion module. We visualize the distribution of attention weights after the softmax operation for different layers of mBERT in Figure 2. Along with the 12 layers of the mBERT-base encoder, we also attend to the embedding layer. Hence, the X-axis of all plots indicates layer 1-13, with 1 being the embedding layer and 13 being the 12th layer of BERT. The Y-axis denotes the attention weight associated with a layer. The attention weights across layers sum up to 1.0.

We show the attention weight distribution for IC and SL tasks in Figure 2 to investigate the learned attention patterns for different tasks. We observe that the attention-fusion module attends to mBERT layers differently for different tasks. IC focuses on mid-late layers while SL focuses on early and mid-layers. The result demonstrates that the learned attention weights vary across tasks and thus attention-fusion can improve task adaptability with its flexibility in using intermediate representations. We hypothesize that IC relies on higher-level semantic information, while SL attends to both token-level embedding input from lower layers as well as contextual information from higher layers.

We also visualize the weights learned for IC and SL tasks on four languages Figure 2 to compare the attention patterns across languages. We observed that all languages learn similar attention weight distribution for different mBERT layers for each of the tasks. This observation explains the language transfer-ability of the attention-fusion model as described in the previous section. To further demonstrate that the attention-fusion module is task-specific and language-agnostic, we take a pre-trained attention-fusion module trained on EN language, freeze it and use it for different languages such as DE, FR and ES. We then compare this with the baseline in which the attention-fusion module is trained and evaluated on the same target language. As shown in Table 4, we observe comparable results against baseline; this suggests that the attention-fusion module is task-dependent and can transfer effectively across different languages.

6 Conclusion

With the rising popularity of transfer learning in NLU, the challenge of adapting pre-trained models to NLU tasks effectively and efficiently is becoming increasingly relevant. To address this challenge, we propose a light yet efficient task-specific attention-fusion module which enables parameter sharing and efficient fine-tuning for downstream tasks. We demonstrate that our proposed late-fusion module achieves comparable performance

[3]For other approaches, our results showed attention-fusion is 4x faster than prompt-tuning v1& v2 and 1.5x faster than Adaptor when training with original released implementation. Due to differences in underlying training infrastructure, we did not report these results in Figure 3.

to other popular methods as well as the fine-tuning approach, while using less tune-able parameters per task. We also show that the task-specific attention-fusion module is transferable across languages, enabling language expansion work in NLU at a much lower cost.

References

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 535–541*.

Jin Cao, Wang Jun, Shang-Wen Li, Kelly Vanee, and Wael Hamza. 2020. Style attuned pre-training and parameter efficient fine-tuning for spoken language understanding. In *2020 INTERSPEECH*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Prakhar Ganesh, Mohammad Ali Khan Yin Yang Deming Chen Marianne Winslett Hassan Sajjad Yao Chen, Xin Lou, and Preslav Nakov. 2020. Compressing large-scale transformer-based models: A case study on bert. In *arXiv:2002.11985, 2020.*

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: studying the effects of weight pruning on transfer learning. In *CoRR, abs/2002.08307*.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning.

Song Han, Huizi Mao, and William J. Dally. 2016. Compressing deep neural networks with pruning. In *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In Proceedings of ICLR, 2016*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network.

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.

Chao-Wei Huang and Yun-Nung Chen. 2019. Adapting pretrained transformer to lattices for spoken language understanding. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. In *Proceedings of the EMNLP 2019*.

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. *arXiv preprint arXiv:1909.11942*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the EMNLP 2021*.

Xiao Liu, Kaixuan Ji1, and Zhilin Yang Jie Tang Yicheng Fu1, Zhengxiao Du. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks.

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

Saab Mansour and Haider Batool. 2020. Atis - seven languages. In *LDC2021T04, Philadelphia: Linguistic Data Consortium, 2021*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

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

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*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Ian Tenney, Alex Wang Adam Poliak R Thomas McCoy Najoung Kim Benjamin Van Durme Sam Bowman Dipanjan Das Patrick Xia, Berlin Chen, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *In Proceedings of ICLR, 2019*.

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.

Gökhan Tür, Jerry Wright, Allen Gorin, Giuseppe Riccardi, and Dilek Hakkani-Tür. 2002. Improving spoken language understanding using word confusion networks. In *Seventh International Conference on Spoken Language Processing*.

Ashish Vaswani, Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Noam Shazeer, Niki Parmar, and Illia Polosukhin. 2017. Attention is all you need. In *CoRR, abs/1706.03762*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019b. Structured pruning of large language models. In *arXiv:1910.04732, 2019b*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.