Sharing Encoder Representations across Languages, Domains and Tasks in Large-Scale Spoken Language Understanding

Jonathan Hueser[1,*], Judith Gaspers[1], Thomas Gueudre[1], Chandana Satya Prakash[1], Jin Cao[2], Daniil Sorokin[1], Quynh Do[1], Nicolas Anastassacos[1], Tobias Falke[1], Turan Gojayev[1], Mariusz Momotko[1], Denis Romasanta Rodriguez[1], Austin Doolittle[1], Kartik Balasubramaniam[1], Wael Hamza[1], Fabian Triefenbach[1], Patrick Lehnen[1]

[1] Amazon Alexa AI [2] Apple

Abstract

Leveraging representations from pre-trained transformer-based encoders achieves state-of-the-art performance on numerous NLP tasks. Larger encoders can improve accuracy for spoken language understanding (SLU) but are challenging to use given the inference latency constraints of online systems (especially on CPU machines). We evaluate using a larger 170M parameter BERT encoder that shares representations across languages, domains and tasks for SLU compared to using smaller 17M parameter BERT encoders with language-, domain- and task-decoupled finetuning. Running inference with a larger shared encoder on GPU is latency neutral and reduces infrastructure cost compared to running inference for decoupled smaller encoders on CPU machines. The larger shared encoder reduces semantic error rates by 4.62% for test sets representing user requests to voice-controlled devices and 5.79% on the tail of the test sets on average across four languages.

1 Introduction

Spoken Language Understanding (SLU) plays an essential role in voice-controlled devices such as Amazon Alexa, Apple Siri and Google Assistant. Two commonly studied SLU subtasks are intent classification (IC) and slot filling (SF). While IC classifies an utterance into a set of pre-defined intents, SF aims to extract relevant slot information. For example, given an utterance "play madonna" IC should determine *PlayMusic* as the intent and SF should detect "madonna" as *Artist*. The two subtasks are often modeled jointly (Do et al., 2020; Chen et al., 2019; Guo et al., 2014). In large-scale SLU systems intents can be split into separate domains allowing the same intents to exist in different domains and for domain-specific teams to work independently of each other. Consequently, multi-domain SLU models have been developed that additionally address the task of domain classification (DC). For example, relating back to the previous example DC should detect *Music* as the domain. In this paper, we focus on a multi-domain system with domain-specific IC+SF models due to its advantages for large-scale SLU, such as the support for independent development across domains and the option of updating and deploying only certain domain models instead of the whole system.

Neural network models for IC+SF and DC tasks typically leverage language representations from finetuned language modeling encoders as features. Bidirectional Encoder Representations from Transformers (BERT) pre-trained on large corpora achieve state-of-the-art performance in SLU (Devlin et al., 2019; FitzGerald et al., 2022a; Chen et al., 2019). Larger encoders can give better performance on down-stream tasks after being finetuned (Wang et al., 2020). However, increasing the number of encoder parameters for large-scale online SLU systems is challenging due to increased inference latency and infrastructure cost constraints.

In this paper we present a method for bringing the accuracy benefits of larger encoders to the users of large-scale SLU systems. We propose to share representations from a large frozen multilingual encoder as features for all of the DC and domain-decoupled IC+SF task heads across multiple languages. We compare a 170M parameter (not counting embeddings) shared encoder for four languages (German, Italian, Spanish, French) to a set of language-, domain- and task-decoupled 17M parameter encoders. For reference, the common BERT-base and BERT-large models are 87M and 306M parameters not counting the embeddings (110M and 340M parameters with embeddings). As a reference for infrastructure cost we consider inference of the language-, domain- and task-decoupled 17M parameter encoders on CPU machines as GPU inference does not meet the infrastructure cost constraints of the specific industry

*Corresponding author: hueserjh@amazon.de

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 447–456

SLU system that our baseline is based on. If the 170M parameter shared encoder inference is run on GPU and representations are cached then the inference latency is neutral compared to running the 17M parameter encoder inference on CPU. The infrastructure cost savings from not having to run a 17M parameter encoder inference for every domain and language on CPU pay for the infrastructure cost of the required GPU instances. In fact, the shared encoder architecture obtains infrastructure cost savings in practice.

To summarize, this paper contains the following contributions. We present a novel shared encoder SLU architecture that enables the use of a larger encoder to improve accuracy while staying inference latency and infrastructure cost neutral. We run large-scale experiments with both internal data of an industry SLU system (23 domains in four languages) and the public MASSIVE dataset (18 domains in four languages) (FitzGerald et al., 2022b). A 170M parameter shared encoder compared to 17M parameter decoupled encoders reduces semantic error rates by 4.62% for test sets representing user requests to voice-controlled devices and 5.79% on the tail of the test sets on average across four languages. We introduce a light-weight encoder that is trained together with the language- and domain-decoupled task heads to learn representations for training data unseen by the frozen shared encoder. We conduct empirical analyses that study the shared encoder setup in the context of feature expansion and distribution drift across a real-world SLU system release cycle.

2 Related Work

Sharing language encoder representations across tasks via multi-task learning has been a prominent research thread in recent years that has resulted in a vast literature. For example, Cer et al. (2018) evaluate a transformer encoder (Vaswani et al., 2017) as a universal sentence encoder trained across multiple natural language processing tasks without prior self-supervised pre-training. Liu et al. (2019) show that multi-task learning can be leveraged on top of self-supervised pre-training for BERT encoders to improve performance across multiple natural language understanding tasks. The multi-domain, multi-task approach to joint domain classification, intent classification and slot filling that we employ in our shared encoder pre-finetuning was already explored by Hakkani-Tür et al. (2016).

Using a "text-to-text" approach large language models (LLMs) have been demonstrated to be multi-task learners with cross-task generalization (Wang et al., 2022; Chung et al., 2022; Sanh et al., 2021) and frozen LLMs can also be adapted to different tasks via prompt tuning approaches (Lester et al., 2021).

Multilingual encoder representations for spoken language understanding are often motivated by cross-lingual transfer (Do and Gaspers, 2019; Xu et al., 2020). Zhang et al. (2021) evaluate a multi-head decoding architecture with a multilingual encoder and language-specific task heads for intent classification and slot labeling that is similar to our proposed shared encoder architecture.

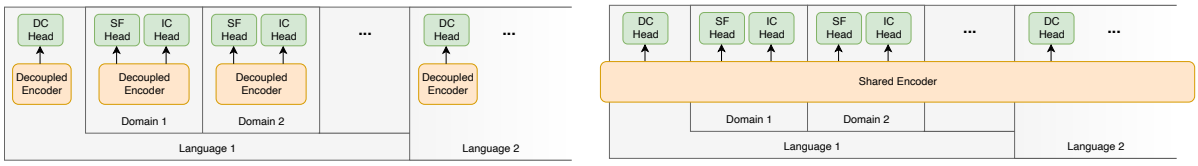
The impact of scaling the parameter count in large language models and pre-trained models in general has been explored extensively by Kaplan et al. (2020); Brown et al. (2020); Radford et al.; Abnar et al. (2021). Distillation can be used to close some of the performance gap between smaller and larger language encoders while meeting inference latency constraints (Jiao et al., 2019; Wang et al., 2020; Soltan et al., 2021). But for example FitzGerald et al. (2022a) show that using larger distilled encoders reduces the semantic error rates for spoken language understanding which motivates our exploration of a shared encoder architecture to enable encoder scaling.

3 Methodology

Our goal is to introduce a larger BERT encoder to increase accuracy for an SLU system with language, domain and task-decoupled IC+SF and DC models without regressions in inference latency and infrastructure cost. Section 3.1 provides context on practical challenges that need to be addressed by a real-world SLU system and that we consider for our proposed model architecture. We then describe how encoder representations are shared across languages, domains and tasks and explain our shared encoder model architecture in Section 3.2. In Section 3.3 we describe the impact on training, inference latency and infrastructure cost.

3.1 Challenges

In real-world SLU systems the data distribution keeps changing and new features such as intents or slots may be added over time. A common approach for feature expansion is via synthetic datasets which are then combined with the exist-



(a) Multiple different encoders for languages, domains and tasks (decoupled baseline).

(b) Single shared encoder for languages, domains and tasks.

Figure 1: Sharing encoder representations in contrast to the decoupled baseline architecture.

ing training data and subsequently used for model training. To support feature expansion and address distribution drift over time it is critical that representations available to the SLU task heads for DC and IC+SF are distinct enough to be able to learn the new data.

3.2 Model Architecture

Decoupled Baseline Our baseline is a multi-domain system with per-language multi-class domain classifiers and per-domain joint IC+SF models. Figure 1a illustrates that decoupled encoders are finetuned per-language, per-domain and per-task. The decoupling of languages and domains comes with the benefit of allowing teams to work on features for different domains and languages in parallel and we aim to keep this benefit. To meet latency constraints for CPU inference the decoupled baseline uses smaller BERT encoders with 17M parameters (not counting embeddings), 4 layers, 768 units, 1200 hidden units, and 12 attention heads (Soltan et al., 2021). Before finetuning, the 17M parameter BERT encoder is distilled from the 2.3B parameter Stage2 Alexa teacher model BERT encoder using a generic language modeling objective (FitzGerald et al., 2022a).

Shared Encoder As an alternative to smaller decoupled encoders we evaluate sharing encoder representations of a larger encoder across languages, domains and tasks while keeping the benefit of decoupled languages and domains for the task heads (see Figure 1b). For the shared encoder we use a larger BERT encoder with 170M parameters (not counting embeddings), 16 layers, 1024 units, 3072 hidden units, and 16 attention heads.

To stay inference latency neutral compared to the smaller decoupled encoders on CPU machines we perform shared encoder inference on a separate GPU machine. As illustrated in Figure 2 for the example of IC+SF the shared encoder representations are communicated to CPU machines running

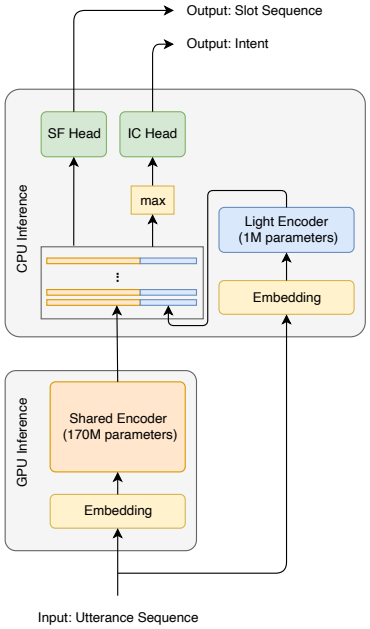


Figure 2: IC+SF model utilizing shared encoder representations. The DC model is analogous to the IC+SF model with only a classification head, i.e. the DC model has its own light encoder.

decoupled task head inference.

To address distribution drift and the feature expansion, the decoupled components are extended with a tiny one-layer BERT encoder that we call *light encoder*. If training data for a new feature is added for a feature release without training a new shared encoder then the shared encoder will not be familiar with the new feature and its representations may not be sufficient for task heads to learn the new feature. The light encoder is trained from scratch to learn useful representations for examples that the shared encoder was not trained on. The representation from the light encoder is concatenated to the representation from the shared encoder before being passed to the task heads.

The SF head is a sequence labeling model that consumes the token level representations. The IC head is a classification model that consumes the se-

quence level representation. We use max-pooling to obtain the sequence level representation from the token level representation sequence produced by concatenating the shared and light encoder representations. In Figure 2 the *max* denotes max-pooling from the stack below that denotes the concatenated representation sequences.

3.3 Training and Inference

To finetune the shared encoder for SLU we introduce an additional pre-finetuning step that uses a multi-task DC+IC+SF objective on data from all domains and languages combined. Before pre-finetuning, the larger BERT encoder is distilled from the 2.3B parameter Stage2 Alexa teacher model BERT encoder using a generic language modeling objective (FitzGerald et al., 2022a). During training of the task heads and light encoder the shared encoder stays frozen to enable decoupling. Caching the frozen encoder outputs for training and validation examples can significantly reduce training time (Liu et al., 2021). Training the task heads and light encoder on top of a cached frozen shared encoder is faster than finetuning the decoupled smaller encoders in the baseline. We provide more details about reducing training time through caching in Appendix A.3.

During online inference a cache for the encoder representations can also be used to cover a large percentage of the user traffic distribution and reduce inference latency at the corresponding percentile. For the tail of the traffic distribution that is not captured in the cached percentile GPU inference is used for the shared encoder. In practice, co-located GPU inference with a cache for the shared encoder and CPU inference for the decoupled light encoders and task heads reduces the infrastructure cost compared to the decoupled baseline.

4 Results

4.1 Experimental Setup

Dataset We report results on both an internal dataset that is representative of user requests to voice-controlled devices and on the public MASSIVE dataset (FitzGerald et al., 2022b). For the internal dataset utterances were de-identified and annotated with intent, slot and domain labels. The data spans four languages and 23 domains per language. The number of training, validation and test utterances is on the order of several million, several hundred thousand and at least several thousand, respectively. The *full* test sets are representative of the whole user requests distribution. The *tail* test sets are representative of the tail of the user requests distribution (low frequency utterances). The tail test sets were generated from the full test sets by filtering out utterances with a frequency greater than one. *Feature* test sets are representative of individual new features introduced in a release of the SLU system where for simplicity each new feature introduces a new intent. The public MASSIVE dataset contains 1M professionally labeled virtual assistant utterances for 51 languages, 18 domains, 60 intents, and 55 slots out of which we report overall and domain-wise results on four languages.

Metric For evaluation we report the relative Semantic Error Reduction Percentage ($\mathrm{SemERR\%}$) compared to the decoupled baseline model where higher is better. Semantic error rate (SemER) measures intent classification and slot filling jointly and is defined as

$$\mathrm{SemER} = \frac{\#(\text{slot+intent errors})}{\#\text{slots in reference + 1}} \,. \qquad (1)$$

If the domain prediction is incorrect then all slot and intent predictions will be incorrect except for cases where there is intent or slot overlap between domains. $\mathrm{SemERR\%}$ is computed as $\mathrm{SemERR\%} = 1 - \mathrm{SemER}_{sha}/\mathrm{SemER}_{dec}$ where SemER_{dec} is the semantic error rate of the decoupled baseline model and SemER_{sha} is the semantic error rate of a shared encoder model.

4.2 Main Results

In Table 1 we report the performance of the shared encoder architecture as described in Section 3.2 on the full and tail tests sets of the following four languages: German (DE), Italian (IT), Spanish (ES) and French (FR). The shared encoder architecture delivers consistent semantic error rate reductions across all languages. The error rate reductions on the tail test sets are larger than those on the full test sets which demonstrates the larger encoders ability to better generalize to harder user requests. In fact, SLU systems can often cover the head of the utterance distribution through deterministic rules making a performance improvement on the tail test set more relevant for neural network models.

In Table 2 we report the absolute semantic error rate performance of the decoupled and shared encoder architectures trained and evaluated on the same four languages in the public MASSIVE

Test set	DE	IT	ES	FR	Avg.
full	4.75	3.38	5.77	4.59	4.62
tail	5.98	4.85	6.43	5.92	5.79

Table 1: SemERR% (↑) for the shared encoder architecture evaluated on the full and tail test sets of four languages.

Language	dec	sha	SemERR% (↑)
DE	22.85	18.00	21.24
IT	23.42	20.07	14.31
ES	27.57	21.34	22.61
FR	25.33	19.40	23.43

Table 2: SemER for the decoupled (dec) and shared (sha) encoder architectures and corresponding SemERR% (↑) evaluated on the MASSIVE dataset for four languages.

dataset. The shared encoder architecture again delivers consistent semantic error rate reductions across all languages.

4.3 Analysis

In this section we conduct ablation studies to better understand the role of the light encoder, task-specific pre-finetuning and shared encoder size. We also explore feature expansion and the impact of encoder age, i.e. the time difference between shared encoder pre-finetuning and task head finetuning which can cause distribution drift in the training data. The experiments in this section use the German training data for light encoder and task head finetuning while the encoder pre-finetuning is always multi-lingual.

Ablations In Table 3 we report the performance of the shared encoder with light encoder (w/ LE), without light encoder (w/o LE), without pre-finetuning (w/o PFT) and using a 17M parameter encoder (17M params) instead of the 170M parameter encoder on the German full and tail test sets.

Removing the light encoder only has a slight impact on performance on the full and tail test sets. For this evaluation the shared encoder has already seen the same training data during pre-finetuning that is used for the light encoder and task head finetuning so we do not expect the light encoder to produce independent representations. The light encoder is motivated by the distribution drift and feature expansion that we analyze later in this section. Since the light encoder ablation does not control for model capacity the improved performance

Model	full	tail
w/ LE	4.75	5.98
w/o LE	4.14	5.43
w/o PFT	-2.73	-2.66
17M params	-2.12	-2.18

Table 3: SemERR% (↑) for shared encoder architecture ablations on the German full and tail test sets.

from including the light encoder may simply be due to an increased number of parameters in the domain-decoupled model component.

Removing the pre-finetuning step means that the encoder representations are not task-specific for SLU which causes a performance regression on the full and tail test sets. Without pre-finetuning, the shared encoder is only pre-trained and distilled on generic masked language modeling albeit on both public data and SLU utterances (see FitzGerald et al. (2022a) for details about Stage2 distillation).

Using a 17M parameter encoder instead of the 170M parameter encoder means that the encoder does not have enough capacity to learn good representations for all domains in all four languages which causes a performance regression on the full and tail test sets. The smaller 17M parameter encoder with pre-finetuning step only slightly outperforms the larger 170M parameter encoder without pre-finetuning step. For the 170M parameter encoder without pre-finetuning the performance regression on the tail test set is not as large as on the full test set while for the 17M parameter encoder the opposite is the case. A possible reason for this difference is that generic representations from the larger encoder without pre-finetuning can help generalization while the smaller encoder with pre-finetuning has to specialize on the seen SLU training data more and may not generalize as well.

Encoder Age To leverage caching and reduce costs, a frozen shared encoder should ideally remain deployed for several months without any updates. We pre-finetune encoders on training data of different age relative to the training data used for light encoder and task head finetuning to investigate the impact of keeping the same encoder deployed for longer time periods. In Table 4 we report the performance of the shared encoder pre-finetuned on zero (0mo), three (3mo) and six (6mo) months old training data (on the same zero months old test data in all setting). We report results for shared encoder architectures with (w/ LE) and with-

Model	Test set	0mo	3mo	6mo
w/ LE	full	4.75	4.44	4.34
	tail	5.98	5.83	5.63
w/o LE	full	4.14	4.14	3.84
	tail	5.43	5.63	5.23

Table 4: SemERR% (↑) for shared encoder pre-finetuning with older training data evaluated on the German full and tail test sets.

Model	PFT w/o feat	PFT w/ feat
w/ LE	-12.93	3.58
w/o LE	-66.68	1.16
w/ LE + upsamp	5.18	
w/o LE + upsamp	-16.40	

Table 5: SemERR% (↑) for shared encoder pre-finetuning with and without feature data evaluated on the German feature expansion test set.

out (w/o LE) light encoder to see the whether the light encoder mitigates the negative effect from distributional mismatches.

With the light encoder there is a small but consistent and monotone decline of performance improvement with encoder age for both the full and tail test sets. Without the light encoder the performance decline with encoder age is not as consistent. For example, the three months old encoder is on-par with the zero months old encoder on the full and even better on the tail test set. Such effects can be caused by seasonal changes in the training data that may not be represented in the test data. The light encoder consistently improves the performance for all encoder ages on both the full and tail test sets and, hence, seems to be able to fill in some of the gaps of the shared encoder.

Feature Expansion For our feature expansion evaluation we consider the case of adding three completely new intents. In Table 5 we report the performance of the shared encoder for the feature expansion on both a worst-case and a best-case scenario in relation to feature data availability during pre-finetuning. The reported scores are the unweighted average SemERR% of three separate feature test sets for the three new feature intents. For the best-case scenario (PFT w/ feat) the encoder pre-finetuning training data is the same as that for the finetuning step and includes the three new feature intents. For the worst-case scenario (PFT w/o feat) the encoder is pre-finetuned on training data excluding all three new feature intents. The new feature intents are added back into the training data for the light encoder and task head finetuning step. We report results for shared encoder architectures with (w/ LE) and without (w/o LE) light encoder to see whether the light encoder mitigates the negative effect of not having seen the new features during encoder pre-finetuning. For the worst-case scenario we also test upsampling (+ upsamp) one of the features by doubling its training data during the finetuning step in order to better enable the light encoder to learn this feature. We upsample data for the feature with the smallest amount of training data for the shared encoder model but not the baseline.

For the best-case scenario of a shared encoder with all feature data during pre-finetuing there is a performance improvement on the feature test sets both with and without light encoder meaning a larger shared encoder can help learn new features better. However, without feature data during pre-finetuning and without light encoder there is a large regression on the feature test sets meaning that the shared encoder representations are not conducive to learning the new feature quickly in the worst-case scenario. Both the light encoder and upsampling the smallest feature individually help reduce the regression but do not remove it completely. When combining the light encoder with the upsampling a performance improvement on the new features is obtained even for the worst-case scenario of not having seen the new features during shared encoder pre-finetuning.

5 Conclusions

We present a novel shared encoder architecture that enables the use of larger encoders in a real-world SLU system while staying inference latency and infrastructure cost neutral. By sharing representations from a larger encoder across languages, domains and tasks the semantic error rates of the SLU system can be reduced consistently across languages for test sets representing both the full user request distribution and its tail. Our empirical analyses reveal that a light-weight encoder can be used in combination with the shared encoder architecture to avoid retraining the frozen shared encoder for every new feature release.

Limitations

In this paper we compare a shared encoder architecture for SLU to a baseline architecture that was chosen based on the specific latency and cost constraints of an industry SLU system. Since encoder model sizes were chosen based on specific constraints the results may not be directly comparable to model sizes more commonly used in the literature such as BERT-large and BERT-base. We expect the general benefit and order of magnitude of accuracy improvements shown in our evaluations to transfer to comparable setups with different parameters.

The primary focus of this paper is on accuracy improvements and addressing challenges of real-world SLU systems such as distribution drift and feature expansion. We do not elaborate on the details of the computational cost and inference aspects. A detailed analysis of compute cost and benchmarks of CPU and GPU inference would better highlight the infrastructure cost benefits of a shared encoder architecture for SLU.

Regarding the multi-lingual aspect of the encoder we only tested a single grouping of similar European languages (German, French, Italian and Spanish). A more extensive analysis of different language groups would demonstrate that similar trade-offs seen in other works on multi-lingual language models also apply for the shared encoder architecture.

Ethics Statement

The shared encoder architecture proposed in this paper significantly reduces compute infrastructure cost of large-scale SLU systems in practice. A large absolute compute infrastructure cost reduction implies a positive environmental impact due to less power consumption.

References

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. 2021. Exploring the limits of large scale pre-training. *arXiv preprint arXiv:2110.02095*.

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

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174.

Q. Chen, Z. Zhuo, and W. Wang. 2019. Bert for joint intent classification and slot filling. *arXiv:1902.10909*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32.

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)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Quynh Do, Judith Gaspers, Tobias Roeding, and Melanie Bradford. 2020. To what degree can language borders be blurred in BERT-based multilingual spoken language understanding? In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Quynh Ngoc Thi Do and Judith Gaspers. 2019. Cross-lingual transfer learning for spoken language understanding. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5956–5960. IEEE.

Jack FitzGerald, Shankar Ananthakrishnan, Konstantine Arkoudas, Davide Bernardi, Abhishek Bhagia, Claudio Delli Bovi, Jin Cao, Rakesh Chada, Amit Chauhan, Luoxin Chen, et al. 2022a. Alexa teacher model: Pretraining and distilling multi-billion-parameter encoders for natural language understanding systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2893–2902.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. 2022b. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. 2021. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:2102.01386*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Saleh Soltan, Haidar Khan, and Wael Hamza. 2021. Limitations of knowledge distillation for zero-shot transfer learning. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 22–31.

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

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual nlu. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5052–5063.

Daniel Yue Zhang, Jonathan Hueser, Yao Li, and Sarah Campbell. 2021. Language-agnostic and language-aware multilingual natural language understanding for large-scale intelligent voice assistant application. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1523–1532. IEEE.

A Appendix

A.1 Implementation Details

Model Architecture The light encoders are one-layer BERT encoders with 320 units, 1200 hidden unites, and 16 attention heads. The task heads are one-layer feed-forward networks with hidden dimension 256 and dropout 0.3 for DC, two-layer feed-forward networks with hidden dimension 256 and dropout 0.5 for IC and two-layer feed-forward networks with hidden dimension 256 and dropout 0.2 for SF, and the SF head uses a CRF layer.

Training The shared encoder pre-finetuning uses a multi-task DC, IC and SF training loss with equal weights. Pre-finetuning uses Adam with Noam learning rate scheduler (Vaswani et al., 2017, Section 5.3), a learning rate multiplier of 0.4 and a mini-batch size of 128. During pre-finetuning the task heads are a two-layer feed-forward network with hidden dimension 256 with dropout 0.3 for DC, 0.5 for IC and 0.2 for SF, and the SF head does not use a CRF layer.

To mix the training data from different languages during pre-finetuning we use a temperature-based rebalancing approach with language weights as given in Conneau and Lample (2019, Section 3.1) with $\alpha = 0.5$, i.e. with language weights $q_i = p_i^\alpha / \sum_j p_j^\alpha$ with $p_i = n_i / \sum_k n_k$ where n_i is the number of utterances of language i.

The decoupled finetuning trains a single multi-class DC model and uses a per-domain multi-task IC and SF training loss with equal weights for the domain-specific joint intent/slot labelling model. Finetuning uses Adam with Noam learning rate scheduler (Vaswani et al., 2017, Section 5.3), learning rate multiplier of 0.1 for DC, 0.5 for IC+SF and a mini-batch size of 256 for the decoupled encoders. The shared encoder models use a learning rate of 0.5 across tasks and a mini-batch size of 256 for the light encoder and task head training. The baseline model with the decoupled encoder setup uses frozen embeddings and gradual unfreezing to a learning rate multiplier of 1.0 for the encoder weights for DC and a learning rate multiplier of 0.01 for the embeddings and gradual unfreezing to a learning rate multiplier of 0.1 for the encoder weights for IC+SF. The encoder dropout is set to 0.1 for the decoupled baseline. The shared encoder model is trained with frozen encoder and encoder dropout is disabled.

A.2 Domain-wise Results

In Table 6 we report domain-wise absolute semantic error rate performance of the decoupled and shared encoder architectures trained and evaluated on German (DE), Italian (IT), Spanish (ES) and French (FR) in the public MASSIVE dataset. The shared encoder architecture outperforms the decoupled baseline on 67 out of 72 domain/language pairs.

A.3 Training Cost Reduction

Building an SLU model on top of a frozen shared encoder gives the opportunity to optimize training cost and latency. Given that the shared encoder is frozen and not updated over the course of training and neural network models are trained over a fixed data set for multiple epochs, the shared encoder is redundantly engaged during the forward pass of each epoch. We eliminate latency overhead and improve training time by caching the output of the shared encoder prior to training downstream SLU components like the light encoder and task heads. By storing the encoder representations of the training and validation data sets, we are engaging the shared encoder only once and subsequent training, forward/backward pass and parameter update, is limited to the light encoder and task heads.

Figure 3 on the right shows the steps of training SLU models with a shared encoder cache. In the first step, we run inference on the shared encoder and store the encoder representations on disk. In the following step, we train the language- and domain-decoupled task heads with cached encoder representations as inputs.

Domain	DE			IT			ES			FR		
	dec	sha	Δ	dec	sha	Δ	dec	sha	Δ	dec	sha	Δ
Alarm	18.99	11.73	38.23	18.89	12.78	32.35	23.6	15.73	33.35	16	17.14	-7.13
Audio	43.24	17.57	59.37	35.14	14.86	57.71	22.97	17.57	23.51	33.78	18.92	43.99
Calendar	23.08	21.57	6.54	27.94	25.71	7.98	26.84	24.62	8.27	26.79	24.87	7.17
Cooking	27.27	22.38	17.93	27.97	25.17	10.01	25.17	20.98	16.65	28.47	25.69	9.76
Datetime	18.09	17.59	2.76	22.73	18.69	17.77	24.37	14.72	39.6	20.6	13.57	34.13
Email	17.32	13.81	20.27	18.31	14.81	19.12	25.63	17.02	33.59	18.14	13.4	26.13
General	18.22	14.13	22.45	18.59	15.99	13.99	20.82	15.99	23.2	18.66	13.81	25.99
Iot	17.3	14.05	18.79	20.54	17.3	15.77	18.75	16.85	10.13	16.22	15.68	3.33
Lists	19.72	15.49	21.45	23	22.07	4.04	24.06	26.42	-9.81	19.91	18.01	9.54
Music	29.7	14.85	50	23.76	18.81	20.83	30.69	19.8	35.48	24.75	16.83	32
News	24.89	22.36	10.16	29.11	26.16	10.13	35.02	27.85	20.47	29.87	24.68	17.38
Play	34.55	32.66	5.47	34.91	29.51	15.47	40	38.34	4.15	35.63	35.34	0.81
QA	19.07	13.18	30.89	21.3	16.02	24.79	20.12	16.02	20.38	20.45	18.18	11.1
Recomm.	32.82	28.21	14.05	33.33	31.28	6.15	46.88	29.69	36.67	47.18	30.26	35.86
Social	17.62	11.89	32.52	11.45	11.89	-3.84	18.06	13.22	26.8	17.62	11.01	37.51
Takeaway	32.8	31.2	4.88	28.57	29.37	-2.8	28.57	30.95	-8.33	26.98	25.4	5.86
Transport	18.3	15.36	16.07	21.24	16.99	20.01	21.85	17.55	19.68	18.95	14.71	22.37
Weather	15.54	14.41	7.27	18.93	15.54	17.91	22.82	15.21	33.35	19.89	18.75	5.73
Overall	22.85	18	21.24	23.42	20.07	14.31	27.57	21.34	22.61	25.33	19.4	23.43

Table 6: SemER for the decoupled (dec) and shared (sha) encoder architectures and corresponding SemERR% (↑) denoted by Δ evaluated on the MASSIVE dataset for four languages.

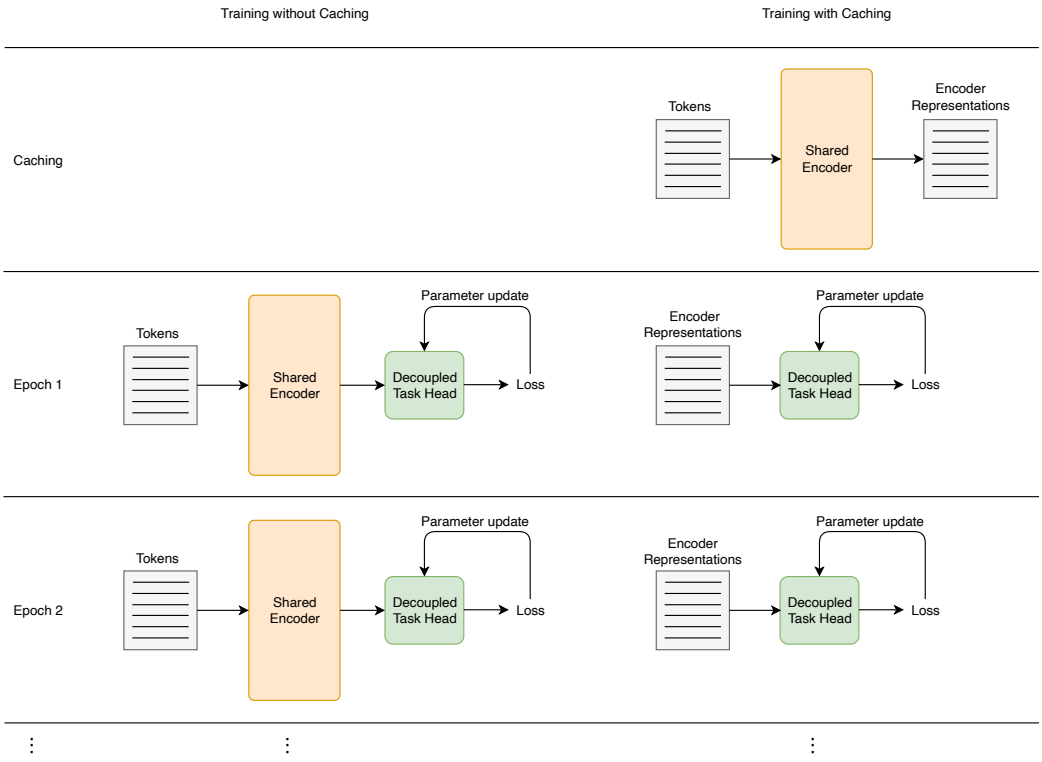


Figure 3: Training decoupled models without (left) and with (right) shared encoder caching