DISTILL-QUANTIZE-TUNE - LEVERAGING LARGE TEACHERS FOR LOW-FOOTPRINT EFFICIENT MULTILINGUAL NLU ON EDGE

Pegah Kharazmi *Zhewei Zhao*
Clement Chung *Samridhi Choudhary*

Amazon Alexa AI

ABSTRACT

This paper describes Distill-Quantize-Tune (DQT), a pipeline to create viable small-footprint multilingual models that can perform NLU directly on extremely resource-constrained Edge devices. We distill semantic knowledge from a large-sized teacher (transformer-based), that has been trained on huge amount of public and private data, into our Edge candidate (student) model (Bi-LSTM based) and further compress the student model using a lossy quantization method. We show that unlike monolingual models, in a multilingual scenario post-compression finetuning on downstream tasks is not enough to recover the performance loss caused by compression. We design a fine-tuning pipeline to recover the lost performance using a compounded loss function consisting of NLU, distillation and compression losses. We show that pre-biasing the encoder with semantics learned on a language modeling task can further improve the performance when used in conjunction with DQT pipeline. Our best performing multilingual model achieves a size reduction of 85% and 99.2% when compared to uncompressed student and teacher models respectively. It outperforms the uncompressed monolingual models (by >30% on average) across all languages on our in-house data. We further validate our approach and see similar trends on the public MultiATIS++ dataset.

1. INTRODUCTION

Natural Language Understanding (NLU) is the task of inferring semantics (domain, intent and entities) from the transcription of an utterance and is the key technology behind commercial voice assistants (VAs). The rise in the adoption of deep learning in NLU [1, 2] has made VAs more reliable and efficient with an estimate of up to 8.4 billion VA devices being available by 2024[1]. With growing number of VA users, privacy, fast and accurate response and availability without internet connection, across different languages becomes critical for good experience. 'On-device' or 'Edge' processing has been an important enabler, with increasing number of VAs moving to edge [3, 4, 5, 4]. While they have been a practical success, edge-compatible NLU models are still largely trained to support a single language. In an increasingly globalized world, it should be possible for users to address their VAs in the language(s) of their choice. Creating multilingual systems is the key to achieving this goal. These systems not only understand multiple languages but also demonstrate better performance per language due to cross-lingual knowledge transfer [6]. A possible solution is to combine multiple monolingual systems with a component for language identification. While this would work for some applications, it increases the footprint and requires maintaining multiple models, making it a bottleneck for resource-constrained devices.

[1] https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/

Recent development of large Transformer-based models as in [7, 8, 9] and many more, have pushed the frontier for multilinguality in NLU, resulting in state of the art performances, especially for low-resource locales [10, 11, 12, 13]. However, due to their size, inference latency and resource requirements they are problematic to deploy on-device [14]. Driven by such limitations, the last few years have seen the rise of transfer learning approaches in NLU, where knowledge from large transformer models are distilled into smaller footprint models, without much loss in performance [14, 15, 16, 17, 18]. Many distilled versions of these models have pioneered research in efficient transfer learning [19, 20, 21, 22]. These studies achieved parameter-size reductions of $> 80\%$ with $< 10\%$ loss in performance, making on-device inference on certain devices possible. However, strict hardware constraints for VAs require orders of magnitude further reduction. A principled study in [3], lays out various limitations to consider when selecting a suitable on-device candidate for resource-constrained devices. This includes strict memory budget requiring an architecture that is amenable to compression without much loss in performance, rigorous latency targets requiring fast inference, and compatibility with on-device inference engine that often lacks support for sophisticated layers such as self-attention. Driven by these constraints, one of the most successful on-device NLU architectures sofar have been simple LSTM based models, with compression and quantization applied to different parts of the network.

In this work, we design a pipeline to build robust multilingual on-device models. Although our approach is architecture agnostic, we focus our experiments on an on-device compatible Bi-LSTM based multi-task NLU architecture and a quantization schema similar to [3]. Given the limited capacity of our shallow LSTM encoder especially as we increase the number of languages, similar to [23], we propose a simple yet effective approach that transfers knowledge from a large multilingual teacher to our shallow student model. We show that in contrast to the findings in [3], task-specific fine-tuning post compression is not sufficient to regain performance lost due to quantization, as more languages are added to the model. We propose a training and finetuning framework to not only regain the performance but also get a consolidated semantic understanding across languages in a way that our final compressed multilingual model has the best performance per language compared to its compressed/uncompressed monolingual/multilingual counterparts. The main contributions of this work are: (1) We propose a framework to **D**istill-**Q**unatize-**T**une (**DQT**) multilingual on-device models even when the architecture of the teacher (transformer-based) and the student (Bi-LSTM based) is different; (2) We design a unique way to recover the lost performance of the compressed model by fine-tuning the compressed model on a compounded loss; (3) We investigate different viable approaches for cross-lingual knowledge transfer and analyze their effectiveness in an ablation setup. The best performing model created by our pipeline, is the first viable on-device multilingual candidate for our in-house VA.

2. METHODOLOGY

2.1. Edge NLU Architecture

Similar to [3], we design our on-device model as a multi-task model with a shared encoder and three task-specific heads to perform sentence-level domain and intent classification (DC and IC) and sequence-level NER tagging. The shared encoder is a word-piece based Bi-LSTM, while the DC, IC and NER heads are fully-connected (FC) layers. The sentence representation, formed by concatenating the final states of the forward and backward LSTM of the last encoder layer, is passed to the DC and IC FCs as input. The word representation, formed by concatenating the bi-directional hidden states at each time-step is passed to the NER FC to predict the slot for each word. The model is trained to minimize the weighted cross entropy (CE), where λ_i controls importance of a task-specific CE:

$$L_{NLU} = \lambda_1 L_{DC} + \lambda_2 L_{IC} + \lambda_3 L_{NER} \quad (1)$$

2.2. Distill-Quantize-Tune (DQT)

Our DQT pipeline consists of three steps - (1) *Distill* knowledge from a multilingual teacher into a student; (2) *Quantize* the model post teacher-supervised training; and (3) *Tune* the model to recover the performance lost in step 2.

Distill - *Cross-lingual Distillation from Large Teachers* We use the teacher model from [24] for performing knowledge distillation (KD) into our Edge student. The teacher is first fine-tuned on downstream tasks. To perform KD, we calculate multi-component KD loss for DC, IC and NER tasks as:

$$L_{Distill} = \lambda_1 L_{DC}^{KD} + \lambda_2 L_{IC}^{KD} + \lambda_3 L_{NER}^{KD} \quad (2)$$

where L^{KD} is the CE loss between teacher and student output probabilities. At inference the model only outputs student predictions.

Quantize - *Additive Quantization using Deep Compositional Code Learning* As $> 90\%$ of our model size is due to the word embeddings, similar to [3], we use additive quantization (AQ), also referred to as Deep Compositional Code Learning (DCCL), first proposed by [25] and popularly used for word-embedding compression [3, 26, 27]. AQ compresses the original embedding matrix W into a set of K integer codes, which in turn create a combination of a set of M codebooks. Each codebook contains K basis vectors referred to as codewords of size D (embedding dimension). DCCL learns the codes and codebooks using an autoencoder with a Gumbel-Softmax, details of which can be found in [26].

Tune - *Post Compression Fine-tuning* DCCL, like most of the post-processing compression approaches, is a lossy quantization scheme as the compression model has no supervision from the downstream task. In order to recover this lost performance we employ a suite of approaches, where we jointly fine-tune the DCCL compression layers of the student model on the downstream tasks. Details of the proposed approaches are given below.

- **Fine-Tuning with NLU Loss** Similar to [3], here the compressed model is fine-tuned on the L_{NLU} from equation 1. The task-aware embeddings are reconstructed by passing the input embeddings through the compression layers. Using Gumbel-Softmax allows the gradients to backpropagate all the way, thereby fine-tuning the compression layers jointly with the remaining layers in the model. We call this approach the vanilla DQT pipeline.

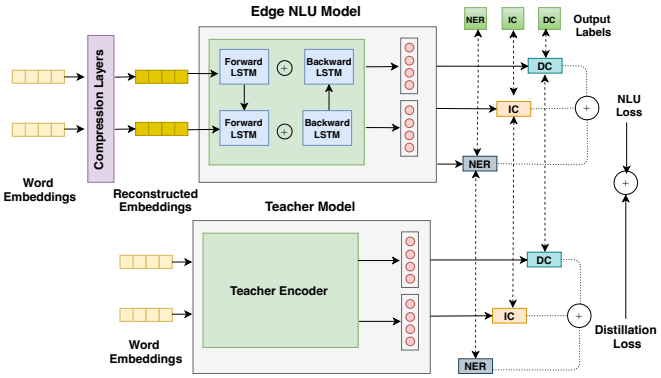

Fig. 1: DQT with KD Loss Setup.

- **Fine-Tuning with Distillation Loss** As shown in figure 1, we jointly fine-tune L_{NLU} (equation 1) and $L_{Distill}$ (equation 2) by defining a joint loss:

$$L_{joint} = \lambda_1 L_{NLU} + \lambda_2 L_{Distill} \quad (3)$$

  We hypothesize that providing supervision signals from a semantically robust teacher, after a lossy compression, should help both the quantized and un-quantized parameters of the model better adjust to the final task, compared to just providing the supervision from the NLU task from student (as in the previous technique).

- **Fine-Tuning with Embedding Loss** In this setup, we add the embedding reconstruction loss (L_e) to the L_{joint} (equation 3). L_e is defined as the mean squared error between the reconstructed (w_i') and original embeddings (w_i). We hypothesize that gradients from this joint loss, especially those from L_e, provide a stronger signal to the compression layers, improving the downstream performance. The joint loss is defined in equation 4, where α_i determines each term's weight.

$$L_{joint} = \alpha_1 L_{NLU} + \alpha_2 L_{Distill} + \alpha_3 \sum_{i=1}^{N} (w_i - w_i')^2 \quad (4)$$

- **DQT with Encoder Pre-Biasing** Similar to [28], we leverage cross-lingual representations learned from multilingual language models (LM) [29]. The LM encoder uses the same architecture as our edge model (word-piece embeddings, Bi-LSTM encoder), stacked with a dense layer to perform next word prediction and is trained on the same dataset. Once trained, learned encoder and embedding weights are used to initialize the corresponding components of the edge model, which is then used as a student in our DQT pipeline with L_{joint} (equation 4). We hypothesize that pre-biasing the encoder with semantics learned from LM task can help the student model before entering the DQT pipeline.

3. EXPERIMENTAL SETUP

3.1. Datasets

We evaluate our approach on two datasets.

Internal NLU Dataset - We use a random snapshot of proprietary, de-identified VA utterances for four languages: English (En), Spanish (Es), French (Fr) and German (De). Each instance consists of an utterance text, along with domain, intent and slot tags. For each

No.	Experiment	IRER Change (%)			
		EN	ES	FR	DE
1	Multilingual Teacher [24]	-49.57	-38.69	-40.60	-50.87
2	Distilled BERT	-44.12	-35.08	-35.61	-42.01
3	Distilled TinyBERT	-34.81	-30.01	-30.96	-34.32
4	Monolingual - uncompressed (B1)	-	-	-	-
5	Monolingual - compressed (B1) [3]	-1.65	-1.22	-0.29	-0.18
6	Monolingual - DQT (B2)	-17.36	-13.99	-13.57	-16.47
7	2L DQT	-18.32	-13.58	N/A	N/A
8	3L DQT	-18.96	-14.98	-12.75	N/A
9	4L D_only (uncompressed)	-27.92	-21.68	-24.97	-26.08
10	4L DQT	-22.87	-16.78	-19.57	-20.04
11	4L DQT + KD Loss	-29.66	-22.67	-25.09	-27.45
12	4L DQT + KD Loss + Emb Loss	-32.05	-25.00	-27.38	30.10
13	**4L DQT Full**	**-33.47**	**-26.11**	**-28.73**	**-32.11**

Table 1: Experiments on Internal Dataset. 2L, 3L and 4L refer to EnEs, EnEsFr and EnEsFrDe respectively. IRER is reported as relative % change. Negative numbers show improvement over baseline.

No.	Experiment	IRER (%)			
		EN	ES	FR	DE
1	Multilingual Teacher [24]	12.21	29.68	26.76	13.45
2	Distilled BERT	13.08	32.12	28.25	16.76
3	Distilled TinyBERT	15.22	36.65	30.12	18.26
4	Monolingual - uncompressed (B1)	20.49	59.35	40.43	23.09
5	Monolingual - compressed (B1) [3]	21.35	60.55	41.22	24.76
6	Monolingual DQT (B2)	20.49	48.35	40.23	22.87
7	2L DQT	20.12	46.06	N/A	N/A
8	3L DQT	19.93	46.69	32.67	N/A
9	4L D_only (uncompressed)	18.03	42.55	31.80	21.41
10	4L DQT	20.38	44.34	32.59	22.98
11	4L DQT + KD Loss	17.91	41.09	31.57	18.95
12	**4L DQT + KD Loss + Emb loss**	**16.69**	**40.54**	**30.46**	**18.61**
13	4L DQT Full	16.91	42.55	31.47	19.96

Table 2: Experiments on MultiATIS++. 2L, 3L and 4L refer to EnEs, EnEsFr and EnEsFrDe. IRER is reported as absolute % values.

language, we have $\approx$ 1600 hours worth of training data, with $>$ 75 intents and $>$ 220 slots across 7 domains. We curate similar datasets ($<$ 200h) for validation and test.

MultiATIS++ - Publicly available MultiATIS++ (MA++) extends the Multilingual ATIS [30] to nine languages across four language families. It contains intent and slot annotations on queries of a flight reservation system. Similar to our Internal Dataset we use a subset of MultiATIS++ for 4 languages (En, Es, Fr, De).

3.2. Baselines

Our target candidate is a 4-language (4L) model. To show the efficacy of DQT, we conduct experiments in the following stages -

Monolingual Models: As our first baseline (B1), we build monolingual models per language, using the same approach as [3], wherein we train, compress and finetune the compressed model on L_{NLU} post-compression (equation 1), without distillation. For a fairer comparison, we train a second monolingual baseline (B2) using vanilla DQT pipeline, where we use the same teacher for distillation and perform fine-tuning on L_{NLU}.

Incremental Multilingual Models: To examine if cross-lingual knowledge transfer has additive effect on top of distillation, we incrementally add languages to the EN monolingual model from B2, creating 2L (EnEs) and 3L (EnEsFr) models. We use vanilla DQT pipeline for each incremental model.

BERT-based Models: In these experiments, we compare our technique with a number of state-of-the-art BERT-based models. We compare our results with 1. the multilingual teacher, a 16-layer 327M parameters BERT-based multilingual model distilled from [24]; 2. A 135M parameters BERT-based model distilled from [24] and 3. A 54M parameters TinyBERT model [15] distilled from [24]. All baselines are further finetuned on the downstream NLU tasks.

DQT Multilingual Models: In these experiments, we train our target 4L Bi-LSTM based edge model using our DQT pipeline. We experiment with all our proposed ways of fine-tuning (section 2.2) and compare the results on each language against the monolingual (B1 and B2), the incremental multilingual and the BERT-based baselines.

3.3. Evaluation and Model Specifics

We use Interpretation Recognition Error Rate (IRER) for evaluating performance. It is the strictest NLU metric and is calculated as the ratio of incorrect interpretations to total number of utterances. An incorrect interpretation is when either the predicted domain, intent or any of the slots is wrong.

Our student is a 2-layer, 256 dimensional Bi-LSTM encoder and 100 dimensional word-piece embeddings (18M parameters). The teacher is a 327M (170M encoder, 157M embeddings) multilingual BERT-based model distilled from the base teacher in [24] which is trained on over 3.84B spoken language examples with 115.2B tokens.

We perform a grid search over learning rate and dropout. In the fine-tuning stage, the joint loss (equation 4) consists of NLU loss, Distillation loss and Embedding loss with weights of 0.425, 0.425 and 0.15 respectively. Both NLU loss and Distillation loss come from the compound loss of IC, DC and NER with weights 0.2, 0.2 and 0.6.

4. RESULTS AND ANALYSIS

We present the experiment results for the internal and MA++ datasets in tables 1 and 2. Owing to the private nature of internal data, we report relative improvements over the uncompressed monolingual baseline (B1) in table 1, whereas absolute IRER numbers and additional intent-wise and slot-wise metrics are reported in table 2 and table 3 on MA++ dataset. For both datasets, our target 4L model has the best performance across all languages, compared to both monolingual (B1) and distilled monolingual (B2) baselines. Among multilingual models using vanilla DQT pipeline (rows 7,8 and 10 in both tables), the 4L model has the best performance, most notably for our internal dataset with at least 4% extra improvement for En compared to 2L and 3L models. Adding languages to the distillation schema results in consistent improvement across both datasets (rows 6,7,8, 10 in both tables). Distillation has an additive effect on both multilingual and monolingual models. B2 models outperform B1 models by at least 14% for internal dataset and $\approx 2\%$ for non-Es and a significant 12% for Es for MA++. To compare the training computational cost of the teacher vs student, we report

Model	Metrics	EN	ES	FR	DE
Multilingual Teacher [24]	ICER	2.69	2.02	2.58	2.80
	SER	10.30	28.00	25.08	11.43
Monolingual - uncompressed (B1)	ICER	6.27	7.05	9.97	7.62
	SER	16.01	58.34	25.16	18.16
4L DQT + KD Loss + Emb loss	ICER	4.03	5.49	5.04	5.72
	SER	13.99	36.95	27.21	14.35

Table 3: Intent classification error rate (ICER) and slot error rate (SER) on MultiATIS++. Metrics are reported as absolute % values.

Model	Relative Size
Multilingual Teacher [24]	1
Distilled BERT	0.41
Distilled TinyBERT	0.17
4L DQT Full (Bi-LSTM)	0.008

Table 4: Comparison of total model size w.r.t the teacher in [24]

the virtual cpu (vcpu) hour required for each. To train the teacher 4,431,872 vcpu/hour is used whereas training the student using the DQT pipeline, requires 35,840 vpcu/hours. The student computation cost is orders of magnitude smaller than that of the teacher.

Through distillation, our uncompressed student achieves a size reduction of 94.5% compared to the teacher (18M vs 327M parameters). Through quantization, we further compress the student, achieving a size reduction of 85%. This puts our compressed candidate model at a size reduction of *99.2%* compared to the teacher. Looking at the effects of quantization, we notice that while for the internal dataset, post-compression finetuning on L_{NLU} can recover the lost performance for B1 models (row 4 vs 5 in table 1), this is not the case for MA++ (rows 4 vs 5 in table 2). While distillation improves the compressed monolingual performance in B2 for the internal dataset, it becomes an essential component to recover the lost performance for MA++. Similar behavior is seen for multilingual models, where the compressed multilingual model shows an average 5% and 1.5% degradation per language in tables 1 and 2 respectively (row 9 vs 10).

Using DQT, we are able to recover and beat the performance of the uncompressed multilingual models (rows 11-13 vs 9-10, both tables). For the internal dataset, the best performance is obtained with Full DQT, i.e. using L_{joint} (equation 4) and LM pre-biasing, with a relative improvement of 30.1% over B1 (rows 13 vs 4 table 1) and 5% over the uncompressed multilingual model (rows 13 vs 9 table 1), on average. The best setting for MA++ is DQT with L_{joint} *without* LM pre-biasing, with an average relative improvement of 9% over B1 (row 12 vs 4 table 2). This might be because MA++ dataset is not large enough for LM-biasing to add any extra semantic knowledge, on top of that transferred through distillation. Training the LM on larger datasets might help get the best performance for MA++.

We also compare the results from our best performing model with state-of-the-art BERT-based baselines in terms of IRER performance (row 13 vs rows 1-3 in table 1 and row 12 vs rows 1-3 in table 2) and model size. Table 4 demonstrates a comparison between the total model size of our best performing model and BERT-based baselines. We use the size of the multilingual teacher as the baseline and calculate the relative size of other models accordingly. It is worth noting that compared to the teacher, distilled BERT and distilled TinyBERT, our shallow Bi-LSTM edge-compatible architecture demonstrates only 2.4%, 1.5%, 0.5% IRER degredation on internal dataset and 5.8%, 4.1%, 1.6% degredation on MA++, while being at 99.2%, 98% and 95.2% size reduction compared to these three baselines, respectively. This further validates that DQT can effectively help boost the performance of simple architecture low foot-print multilingual models, making them viable candidates for edge use cases.

Error Analysis - To further investigate the effects of DQT on increasing semantic robustness, we analyze the most confused intent pairs of our internal dataset with our best performing 4L model. To select the hardest intents, we compute the jaccard similarity index (JSI) between all intent pairs, using a unigram overlap between the intent utterances. We also compute the cross-intent confusion matrix and pick the intent pairs that have a higher than median JSI value and greater than average cross-intent confusion. We categorize top-10 such pairs into 'semantically similar' (ex. to browse notifications vs reminders) and 'semantically opposite' (ex. to ignore vs accept a call). We create TSNE plots of the shared encoder outputs for the compressed B1 and the best performing 4L model. Figure 2 shows such plot, for a number of intent pairs. It can be seen that the encoder outputs from our best performing 4L model (right plots) show a much better separation compared to that of the B1 model (left plots). These observations indicate the importance of the extended DQT pipeline to learn a better semantic space, especially for classes with high word overlap but different semantic labels.

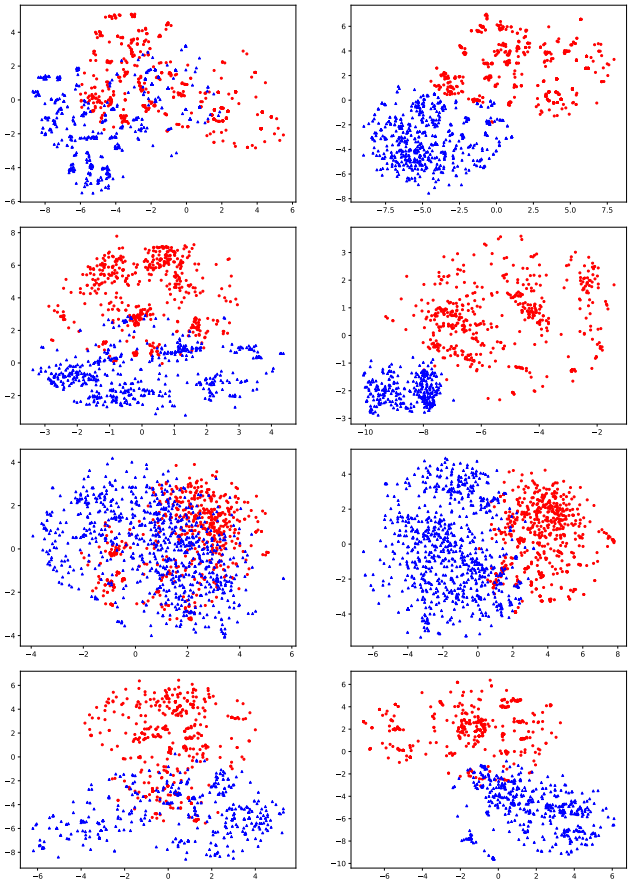

Fig. 2: TSNE visualization of outputs on high jaccard index intent pairs for baseline (left) and the proposed (right) model. From Top to bottom, intents are semantically opposite pairs: to accept vs ignore a call, to extend vs shorten notifications; and semantically similar pairs: to get the time vs the distance of a travel, to browse notifications vs reminders. On these pairs, the proposed model is able to correct 90% of the samples that were misclassified by the baseline model.

5. CONCLUSION

We present a pipeline to build low-footprint multilingual models for edge use cases and show results across four languages and two datasets. We demonstrate that as more languages are added, the model faces a higher performance degradation due to compression. As a solution, we propose a distillation-led, post-compression finetuning setup. We show that incorporating a compounded loss with LM pre-baising helps recover and beat the uncompressed models' performance, while achieving a compression rate of 85% and 99.2% compared to uncompressed student and teacher models respectively.

6. REFERENCES

[1] S. Ravuri and A. Stolcke, "Recurrent neural network and lstm models for lexical utterance classification," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. 1

[2] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *ACL 2019*, June 2019. 1

[3] K. Mysore Sathyendra, S. Choudhary, and L. Nicolich-Henkin, "Extreme model compression for on-device natural language understanding," in *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, Online, Dec. 2020, pp. 160–171, International Committee on Computational Linguistics. 1, 2, 3

[4] A. Saade, A. Coucke, A. Caulier, J. Dureau, A. Ball, Th. Bluche, D. Leroy, C. Doumouro, Th. Gisselbrecht, F. Caltagirone, et al., "Spoken language understanding on the edge," *arXiv preprint arXiv:1810.12735*, 2018. 1

[5] Zh. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020. 1

[6] D. Mueller, N. Andrews, and M. Dredze, "Sources of transfer in multilingual named entity recognition," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020, pp. 8093–8104, Association for Computational Linguistics. 1

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 1

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, et al., "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. 1

[9] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R Salakhutdinov, and Q. V Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763. 1

[10] O. Cattan, Ch. Servan, and S. Rosset, "On the cross-lingual transferability of multilingual prototypical models across nlu tasks," *arXiv preprint arXiv:2207.09157*, 2022. 1

[11] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," *arXiv preprint arXiv:2003.00104*, 2020. 1

[12] S. Louvan and B. Magnini, "Simple data augmentation for multilingual nlu in task oriented dialogue systems.," in *CLiC-it*, 2020. 1

[13] Z. Liu, G. I. Winata, Zh. Lin, P. Xu, and P. Fung, "Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 8433–8440. 1

[14] V. Sanh, L. Debut, J. Chaumond, and Th. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019. 1

[15] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019. 1, 3

[16] S. Sun, Y. Cheng, Zh. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," *arXiv preprint arXiv:1908.09355*, 2019. 1

[17] M. A Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," *arXiv preprint arXiv:2002.08307*, 2020. 1

[18] Zh. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020. 1

[19] P. Ganesh, Y. Chen, X. Lou, M. A Khan, Y. Yang, D. Chen, M. Winslett, H. Sajjad, and P. Nakov, "Compressing large-scale transformer-based models: A case study on bert," *arXiv preprint arXiv:2002.11985*, 2020. 1

[20] H. Saghir, S. Choudhary, S. Eghbali, and C. Chung, "Factorization-Aware Training of Transformers for Natural Language Understanding on the Edge," in *Proc. Interspeech 2021*, 2021, pp. 4733–4737. 1

[21] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert.," *arXiv preprint arXiv:1910.06188*, 2019. 1

[22] Sh. Shen, Zh. Dong, J. Ye, L. Ma, Zh. Yao, A. Gholami, M. W Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 8815–8821. 1

[23] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin, "Distilling task-specific knowledge from bert into simple neural networks," *arXiv preprint arXiv:1903.12136*, 2019. 1

[24] J. FitzGerald, S. Ananthakrishnan, K. Arkoudas, D. Bernardi, A. Bhagia, C. Delli Bovi, et al., "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*, New York, NY, USA, 2022, KDD '22, p. 2893–2902, Association for Computing Machinery. 2, 3, 4

[25] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 931–938. 2

[26] R. Shu and H. Nakayama, "Compressing word embeddings via deep compositional code learning," *CoRR*, vol. abs/1711.01068, 2017. 2

[27] T. Chen, M. Renqiang Min, and Y. Sun, "Learning k-way d-dimensional discrete codes for compact embedding representations," in *ICML*, 2018, pp. 853–862. 2

[28] H. Tu, S. Choudhary, H. Saghir, and R. McGowan, "Leveraging multilingual neural language models for on-device natural language understanding," in *The Web Conference 2021 Workshop on Multilingual Search*, 2021. 2

[29] T. Wada, T. Iwata, and Y. Matsumoto, "Unsupervised multilingual word embedding with limited resources using neural language models," in *ACL*, 2019. 2

[30] G. Tur D. Hakkani-Tur L. Heck Sh. Upadhyay, M. Faruqui, "(almost) zero-shot cross-lingual spoken language understanding," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018. 3