MULTILINGUAL END-TO-END SPOKEN LANGUAGE UNDERSTANDING FOR ULTRA-LOW FOOTPRINT APPLICATIONS

Markus Müller *Anastasios Alexandridis* *Zach Trozenski* *Joel Whiteman*
Grant Strimel *Nathan Susanj* *Athanasios Mouchtaris* *Siegfried Kunzmann*

Amazon Alexa AI, Pittsburgh, PA, USA

ABSTRACT

Tiny Signal-to-Interpretation (TinyS2I) has been recently introduced as an ultra low-footprint end-to-end spoken language understanding (SLU) model. This architecture is capable of running in ultra resource constrained environments like voice assistant devices, while at the same time reducing latency. In this work, we propose an extension to TinyS2I and train a multilingual system supporting several languages. Multilingual TinyS2I models show little to no degradation compared to their monolingual counterparts. Increasing the network size in width and depth improves the classification accuracy for mono- and multilingual setups, with the multilingual one improving beyond the monolingual accuracy. This enables users to interact with the device in the language of their choice and dynamically switch between languages without an explicit language setting or accuracy degradation.

***Index Terms*—** speech recognition, human-computer interaction, multilingual, spoken language understanding

1. INTRODUCTION

Recent research efforts in spoken language understanding (SLU) for virtual assistants have shown the potential to develop and employ models that execute on local devices instead of relying on the traditional cloud-centric system design. Apart from privacy benefits, on-device systems offer significant latency benefits since the audio does not need to be transmitted to the cloud for processing. The two main components of an SLU system are the automatic speech recognition (ASR) model that converts audio to a transcript and the natural language understanding (NLU) model that converts the transcript to a machine-readable interpretation in the form of intent, slot values, and slot tags. For ASR, the latest approaches for on-device models utilize end-to-end systems [1–5] with various optimizations for latency usually tailored for the Recurrent Neural Network Transducer (RNN-T) [1] architecture such as quantization [6, 7], network sparsification [8], and amortized network architectures [9, 10]. Similar approaches for NLU try to compress large models for on-device use [11, 12], or fuse ASR and NLU together in a single, all-neural signal-to-interpretation (S2I) architecture that enables parameter sharing and is more easily compressible [13–17].

Recently, we introduced a novel architecture, TinyS2I [18], as an ulta-low-footprint, all-neural on-device SLU model. In this design, TinyS2I acts as a command classifier for a small set of frequent command-and-control commands that we call "supported commands". All other utterances ("unsupported commands") are deferred to the cloud for SLU processing. This design leverages the specific distribution of utterances spoken to voice assistant devices and is motivated by the fact that a small set of utterances is responsible for a disproportionate large share of traffic. Using this type of dual processing, TinyS2I serves the most common high-frequency utterances on-device, thereby providing significant latency improvements for the bulk of traffic. Coverage for all other utterances is guaranteed by the cloud, thus maintaining the accuracy of a powerful cloud-based SLU model. It has been demonstrated in [18] that with high accuracy, TinyS2I can recognize supported commands and defer unsupported utterances to the cloud for processing with a model footprint that is less than 20 MiB, making it a preferable candidate for ultra resource-constrained environments.

In this paper, we investigate multilingual TinyS2I model designs that recognize supported and unsupported commands across multiple languages. Supporting multiple languages within a single model has several advantages: First, there is only a single model to maintain and deploy for multiple countries; Second, and much more critical, it also enables users to interact with devices in the language of their choice without having to explicitly choose a language setting on their device. This natural way of interacting with voice assistants is becoming more important in an increasingly globalized world.

Recently, the feasibility of training end-to-end SLU systems on multiple languages [19] was demonstrated. However, the approach relies on a transformer-based architecture [20] which is an expressive but computationally prohibitive model for deployment on resource constrained devices. In the regimen of ASR, a multitude of multilingual approaches have been proposed over the years [21–26]. While our approach is geared towards an ultra low-resource setting, [21] proposes a solution for the other end of the spectrum with 1 billion parameters. It is also possible to adapt ASR models for higher accuracy [22, 23] and increase the efficiency of weight sharing [24, 25]. For streaming end-to-end ASR, using a separate language identification component improves the performance [26]. While all these approaches are effective in improving the performance of multilingual ASR systems, they are of limited utility for our use-case as they require either additional (sub-)networks or a network of a certain size. The compute power these ASR systems require makes them unsuitable for ultra-low-footprint on-device applications.

In this work, we explore the effectiveness of multilingual training of the TinyS2I architecture. We explore two approaches of multilingual training of TinyS2I to support use cases where the language information is either implicitly passed to down-stream components or where it is omitted by the network. Our results show the feasibility to train an ultra-low footprint system to support multiple languages with little to no degradation compared to the respective monolingual models. This enables users to interact with the voice assistant in multiple languages dynamically (without explicitly selecting a language setting), while at the same time benefit from latency reductions that come from on-device processing as the audio does not need to be transmitted to the cloud for high frequent commands.

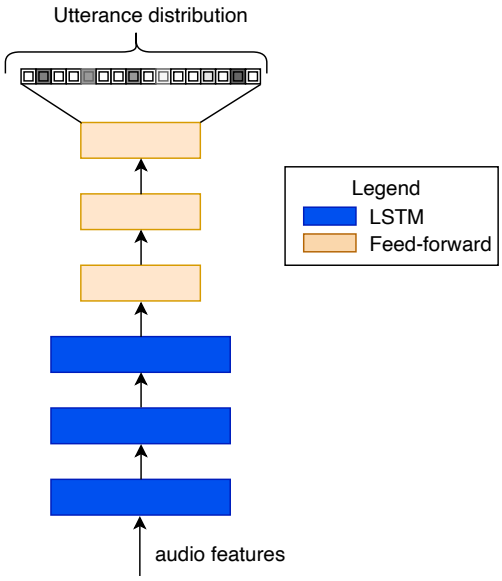

Fig. 1: TinyS2I command-and-control model architecture. It consists of three unidirectional LSTM layers with 128 units, followed by three feed-forward layers.

2. TINYS2I ARCHITECTURE

The TinyS2I [18] model architecture consists of the main command-and-control model that is responsible for recognizing supported commands or identifying when a command is unsupported, and an optional ancillary contextual model component that recognizes utterances with dynamic slot content, like video names or song names. In this work, we focus on the command-and-control component (Figure 1) and investigate its multilingual extensions.

The TinyS2I command-and-control model is an utterance classifier that classifies audio into a predetermined set of classes (i.e, commands). The output of the model is a vector of $M + 1$ probabilities, where M is the number of supported commands, plus one class for unsupported commands. The model consists of a recurrent encoder with three uni-directional LSTM layers of 128 units, followed by two feed-forward layers of 256 units with *tanh* activations, and a final feed-forward layer of $M + 1$ units with a softmax activation. The total number of parameters in the command-and-control TinyS2I model is 500K, which translates to a model footprint as low as 2 MiB.

Inference with the TinyS2I model is based on a threshold t_{unsup} that is compared against the output probability for the unsupported class ℓ_{unsup} and drives the decision of whether the command is supported or unsupported. The audio features are passed through the neural model which outputs an $M + 1$ vector $\boldsymbol{\ell}$ of probabilities that the utterance belongs to one of the M supported commands, or is unsupported. The decision of whether the utterance is supported is based on the following rule:

$$\text{decision} = \begin{cases} \text{supported, if } \ell_{\text{unsup}} < t_{\text{unsup}} \\ \text{unsupported, otherwise} \end{cases} \quad (1)$$

In the event where the utterance is recognized as supported, the utterance class is chosen based on the output probability with the maximum value. As the supported commands are fixed, the final NLU interpretation is derived by a one-to-one mapping (exact match rules) between the predicted supported command and the intent, slot values and slot tags.

The t_{unsup} threshold is used to control how aggressively we sacrifice supported utterances by sending them to the cloud in order to minimize false positives (i.e., unsupported utterances wrongly predicted as supported). If a supported utterance is falsely recognized as unsupported, it will be simply sent to the cloud for recognition. While users will not get the latency benefits of on-device processing, their request will be served by a powerful, more accurate cloud model, and the correct action will be executed. However, when an unsupported utterance is recognized as supported, a wrong action will be performed by the device, degrading user experience. Thus, for this type of systems it is important to minimize false positives, while maintaining a high true positive rate. That means supported commands are processed locally and unsupported ones are correctly deferred to the cloud for recognition.

2.1. Multilingual TinyS2I

We extend the TinyS2I model to support multiple languages. When training the model, we do not provide the language information explicitly to the network to enable use cases where users have the freedom to address the system using any of the supported languages without having to specify a language setting. When scaling to more languages, we evaluate two different approaches on how to select the targets the network is trained on. In the *merged* approach, a single class for the same command across languages is used, which results in a language independent output. This allows to scale the number of language without increasing the output classes of the network, while at the same time it requires careful preparation of the data of have matching classes across languages. The second approach *stacked* uses a separate class for each command in each language. While this increases the number of classes with each language added, it allows for more flexibility as the commands to not need to match across languages. Depending on the downstream application, either configuration may be applicable and hence we are comparing them side by side.

To demonstrate the efficacy of our approach, we chose English, French and German. This is a challenging language combination for evaluating our multilingual models: French and German originate from different language families and share little vocabulary within the target domain for our work. They also present challenges related to morphological structure for conjugations and phonetically-similar pronunciations which are not as prevalent in English. While English and German are linguistically closer, they also share few words across our target commands, e.g., “start”, “stop”.

3. EXPERIMENTAL SETUP

For our experiments, we start with a model size of 500k parameters as a baseline. When adding more languages to the mix, we are forcing the model to recognize more classes to cover the commands of these additional languages. As the task becomes more complex, the accuracy may be limited by the capacity of the model. We therefore explore increasing the number of parameters by making the layers wider, as well as adding more layers. However, even our largest models with 5 LSTM layers of 256 units per layer are still less than 10 MiB in disk footprint, which make them good candidates for heavily resource-constrained devices. With our evaluation, we aim to demonstrate the efficacy of our approach by addressing the

Table 1: Relative changes (%) in classification accuracy compared to the monolingual systems when scaling the number of languages.

Languages	*mode*	*EN*	*DE*	*FR*
Monolingual	–	–	–	–
EN DE	merged	-0.48	-0.48	–
EN DE	stacked	-0.98	-0.61	–
EN DE FR	merged	-0.79	-0.73	0.16
EN DE FR	stacked	-1.44	-1.40	-0.88

following research questions: (1) How does increasing the number of languages impact the classification accuracy? (2) Does increasing the number of parameters improve the accuracy for mono- and multilingual models in the same way? (3) What is the optimal network size for the *stacked* and *merged* configuration?

3.1. Data Preparation

For our experiments, we chose a set of 91 frequent command-and-control utterances for voice assistants, such as "turn on the lights", "volume up", "play" or "stop". For each language, we filter for frequent occurring commands and correlate them with matching ones in the other languages. We remove commands whose translated forms did not exist in all three languages. This way, we assemble a list of frequent commands that are present across all three languages. For both French and German, we add additional conjugations where necessary to cover other (grammatically different but acoustically similar) ways to speak the same command, e.g, "augmentez le volume" and "augmenter le volume" are included as conjugations of "augmente le volume" for French. These conjugations are not added as separate classes to the model, but are joined with the original command on the same class. We further select the utterances to have an equal distribution of examples per class. This includes downsampling as well as upsampling in cases where not enough examples per class are available.

For unsupported commands we use any other command that does not belong on the supported list. In order to collect enough data with a balanced distribution across the supported commands for each language, we used a semi-supervised approach where the ASR output of a powerful cloud ASR model was treated as the groundtruth. Our training and evaluation data consists of far-field de-identified data. For training each language, we use 300 hours of supported utterances with a balanced distribution of examples across the 91 supported commands, and 100 hours of unsupported utterances. Experimentally, we found that this ratio was sufficient to train robust models. For evaluation, we use 10 hours of supported utterances with a balanced distribution and 25 hours of unsupported utterances, for each language. To the best of our knowledge, we are not aware of a public multilingual end-to-end SLU data set to benchmark our system against. However, we believe that analyzing technology on real-world data provides valuable insights to the science community.

3.2. System Training

The audio features to train our models on are 64-dimensional log-filter bank energies (LFBEs), obtained by segmenting utterances with a window of 25 ms length and frame shift of 10 ms. We use a left context of 3 frames, resulting in 192-dimensional input features with a skip rate of 3 frames. The features are normalized with global mean and variance. We use a learning rate scheduler with a starting learning rate of 1e-5, which is gradually increased up to 0.001 before the decay phase, which ends in 1e-5. We further apply dropout of 0.15 to the LSTM layers.

Table 2: Relative changes (%) in classification accuracy for different encoder network sizes, monolingual. 128 neurons with 3 layers is the baseline to compare against.

Encoder size	*# of params*	*EN*	*DE*	*FR*
128 neurons, 3 layer	0.5M	–	–	–
128 neurons, 4 layer	0.7M	0.19	-1.80	1.73
128 neurons, 5 layer	0.8M	0.50	-0.06	1.77
192 neurons, 3 layer	1.0M	0.56	**0.39**	2.18
256 neurons, 3 layer	1.6M	0.46	0.01	1.91
192 neurons, 4 layer	1.3M	0.38	-0.21	**3.13**
256 neurons, 5 layer	2.7M	**0.72**	0.28	2.36

3.3. Evaluation metrics

For our analysis, we use the classification accuracy as primary metric. We report the changes in classification accuracy relative to the respective baseline, and all our baseline systems have a classification accuracy above 90%. A TinyS2I-like system must be able to accept and process on-device supported commands, giving accurate NLU interpretations, while correctly rejecting unsupported commands and deferring recognition to the cloud (for a more thorough discussion, refer to Section 2). For the final evaluation in subsection 4.3 we report two metrics: The True Positive Rate (TPR) shows the percentage of supported commands correctly recognized as supported and processed on-device by TinyS2I. The False Positive Rate (FPR) on the other hand shows the percentage of unsupported commands that were erroneously identified as supported by TinyS2I.

4. RESULTS

4.1. Scaling the number of languages

This first set of experiments is based on the network size of the original TinyS2I model (500K parameters). We train this system on multiple languages and compare the results. Thereby, we use two different modes of operation: *merged* and *stacked* (see Section 2.1). Table 1 shows the results: adding the second language (German) to English shows a minor degradation in accuracy for both languages. Comparing the two operating modes, the *stacked* condition shows a higher degradation over *merged*. When adding a third language (French) to the mix, we observe a minor additional regression regarding the classification accuracy. Here, we observe the same pattern as for the two-language case with *merged* outperforming *stacked*. It is worth noting that the accuracy for the merged model improved for French, outperforming the monolingual baseline. While merged outputs show a higher accuracy over stacked outputs, the choice may be dependent of the use case in the production environment.

The error analysis of the mis-recognitions shows, that the vast majority of wrongly classified utterances are classified to a wrong command within the same language, while there are only a few mis-recognitions where the predicted class was estimated as a command of another language. We observe that error patterns are more prominent in English and German, where the model was usually confused

Table 3: Relative changes (%) in classification accuracy for different encoder network sizes, multilingual (EN, DE, FR), using a *stacked* configuration. The baseline for each language is respective the monolingual setup with 128 neurons with 3 layers.

English (EN)	*3 layer*	*4 layer*	*5 layer*
128 neurons	–	-0.51	-0.50
192 neurons	0.09	0.83	
256 neurons	0.31		**0.95**

German (DE)	*3 layer*	*4 layer*	*5 layer*
128 neurons	–	-1.11	0.13
192 neurons	-0.34	0.44	
256 neurons	0.03		**0.50**

French (FR)	*3 layer*	*4 layer*	*5 layer*
128 neurons	–	0.64	1.77
192 neurons	1.73	**2.43**	
256 neurons	1.57		**2.43**

Table 4: Relative changes (%) in classification accuracy for different encoder network sizes, multilingual (EN, DE, FR), using a *merged* configuration. The baseline for each language is the respective monolingual setup with 128 neurons with 3 layers.

English (EN)	*3 layer*	*4 layer*	*5 layer*
128 neurons	–	-0.86	-0.89
192 neurons	-0.34	0.20	
256 neurons	**0.74**		0.72

German (DE)	*3 layer*	*4 layer*	*5 layer*
128 neurons	–	-1.25	-1.31
192 neurons	-0.84	-0.14	
256 neurons	**0.62**		0.47

French (FR)	*3 layer*	*4 layer*	*5 layer*
128 neurons	–	-0.04	-0.02
192 neurons	0.75	1.62	
256 neurons	**2.24**		**2.24**

between "turn on" and "turn off" commands. We speculate that this is because command-and-control utterances to turn an appliance on and off are acoustically very similar in English and German. Overall, the same error patterns were observed for both monolingual and multilingual TinyS2I models, which validates that the multilingual training does not introduce additional sources of errors.

4.2. Encoder size

Our experiments up to this point are based on the same network configuration as the original TinyS2I model (Section 2). When training the network on multiple languages, the complexity of the task increases 2 or 3 fold (for 2 or 3 languages) compared to training on just a single language. We therefore experiment with increasing the size of the encoder network. Starting with the baseline configuration of 128 neurons per layer and 3 LSTM layers, we increased both the width as well as the depth of the network to a maximum of 256 neurons per layer and 5 LSTM layers. Table 2 shows the results of increasing the size of the network for monolingual training. While all languages benefit from the increase in model size, the improvement is most noticeable for French. The results for the multilingual models are shown in Tables 3 and 4. They follow the same trend as the monolingual counterparts. Looking at the *merged* and *stacked* conditions separately, the optimal configuration for the *stacked* condition is at 256 neurons and 5 layers, while for the *merged* case, 256 neurons and 3 layers shows the best results. Overall, increasing the size of the network improves the performance, similar to the monolingual case. The multilingual variants benefit more from the size increase: as the accuracy increases for both mono- and multilingual models, with the exception for French, multilingual models outperform the monolingual ones. French shows a relative improvement of 3.13% in the monolingual case using 192 neurons and 4 layers, whereas the largest gain is 2.43% in the multilingual *stacked* case using the same network size. The reason for this behavior may be that this locale required the most upsampling to balance the classes.

4.3. Evaluation of TPR and FPR

TinyS2I runs in conjunction with a cloud model and if an unsupported command is recognized, it defers recognition to the cloud. Thus, apart from the classification accuracy, the TPR and FPR of the model need to be examined to show how well the model rejects unsupported commands and accepts supported ones. Different operating points (in terms of FPR and TPR) can be realized by adjusting the threshold t_{unsup} in the TinyS2I model. Figure 2 shows the TPR and FPR for different operating points, i.e., different values of t_{unsup} in terms of classification accuracy. Operating points closer to the upper left corner are the best candidates, because they represent a high TPR with a low FPR. It can be observed that the multilingual TinyS2I models outperform their monolingual counterparts for all three languages under test. French shows the largest gains over the monolingual baseline.

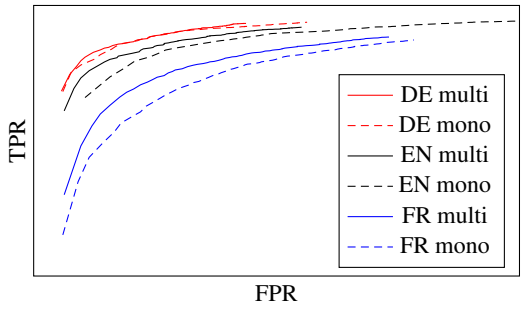

Fig. 2: Comparison of baseline monolingual models with larger multilingual model (5 layer, 256 neurons). The theoretical optimal value would be a TPR of 1.0 and a FPR of 0.0, which is the top left corner in the figure.

5. CONCLUSION

We proposed an approach to train an ultra-low-footprint end-to-end SLU system on multiple languages. Overall, we observed that all multilingual models offer little to no degradation compared to their monolingual counterparts of the same model size. This enables the deployment of multilingual models on heavily resource-constrained devices and offers latency benefits to users that can interact with the device in the language of their choice. Increasing the model size improves the performance for both mono- and multilingual systems, with multilingual systems showing an increased accuracy over the monolingual ones. Our next steps include to add the contextual component to enable support for multilingnual contextual utterances.

6. REFERENCES

[1] Alex Graves, "Sequence transduction with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2012.

[2] Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

[3] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Bejing, China, 22–24 Jun 2014, vol. 32, pp. 1764–1772.

[4] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[5] Hagen Soltau, Hank Liao, and Haşim Sak, "Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition," in *INTERSPEECH*, 2017, pp. 3707–3711.

[6] Hieu Duy Nguyen, Anastasios Alexandridis, and Athanasios Mouchtaris, "Quantization aware training with absolute-cosine regularization for automatic speech recognition," in *INTERSPEECH*, 2020.

[7] Yi Yang, Andy Chen, Xiaoming Chen, Jiang Ji, Zhenyang Chen, and Yan Dai, "Deploy large-scale deep neural networks in resource constrained iot devices with local quantization region," in *arXiv preprint arXiv:1805.09473*, 2018.

[8] Michael H. Zhu and Suyog Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," in *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018.

[9] Jonathan Macoskey, Grant P. Strimel, and Ariya Rastrow, "Bifocal neural ASR: Exploiting keyword spotting for inference optimization," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ISCASSP)*, 2021.

[10] Jonathan Macoskey, Grant P. Strimel, Jinru Su, and Ariya Rastrow, "Amortized neural networks for low-latency speech recognition," in *INTERSPEECH*, 2021.

[11] Hamidreza Saghir, Samridhi Choudhary, Sepehr Eghbali, and Clement Chung, "Factorization-aware training of transformers for natural language understanding on the edge," in *INTERSPEECH*, 2021.

[12] Kanthashree Mysore Sathyendra, Samridhi Choudhary, and Leah Nicolich-Henkin, "Extreme model compression for on-device natural language understanding," in *International Conference on Computational Linguistics (COLING)*, 2020.

[13] Yuan-Ping Chen, Ryan Price, and Srinivas Bangalore, "Spoken language understanding without speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6189–6193.

[14] Natalia Tomashenko, Antoine Caubrière, Yannick Estève, Antoine Laurent, and Emmanuel Morin, "Recent advances in end-to-end spoken language understanding," in *International Conference on Statistical Language and Speech Processing*. Springer, 2019, pp. 44–55.

[15] Marco Dinarelli, Nikita Kapoor, Bassam Jabaian, and Laurent Besacier, "A data efficient end-to-end spoken language understanding architecture," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8519–8523.

[16] Milind Rao, Anirudh Raju, Pranav Dheram, Bach Bui, and Ariya Rastrow, "Speech to semantics: Improve ASR and NLU jointly via all-neural interfaces," *arXiv preprint arXiv:2008.06173*, 2020.

[17] Martin Radfar, Athanasios Mouchtaris, and Siegfried Kunzmann, "End-to-end neural transformer based spoken language understanding," in *Interspeech 2020,th Annual Conference of the International Speech Communication Association*, 2020, pp. 500–505.

[18] Anastasios Alexandridis, Kanthashree Mysore Sathyendra, Grant P. Strimel, Pavel Kveton, Jon Webb, and Athanasios Mouchtaris, "TinyS2I: A small-footprint utterance classification model with contextual support for on-device slu," in *accepted at ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[19] Markus Müller, Samridhi Choudhary, Clement Chung, Athanasios Mouchtaris, and Siegfried Kunzmann, "In pursuit of babel - multilingual end-to-end spoken language understanding," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 1042–1049.

[20] Bhuvan Agrawal, Markus Müller, Martin Radfar, Samridhi Choudhary, Athanasios Mouchtaris, and Siegfried Kunzmann, "Tie your embeddings down: Cross-modal latent spaces for end-to-end spoken language understanding," *arXiv preprint arXiv:2011.09044*, 2020.

[21] Vineel Pratap, Anuroop Sriram, Paden Tomasello, Awni Hannun, Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert, "Massively multilingual ASR: 50 languages, 1 model, 1 billion parameters," in *Interspeech 2020,th Annual Conference of the International Speech Communication Association*, 2020.

[22] Markus Müller, Sebastian Stüker, and Alex Waibel, "Neural codes to factor language in multilingual speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, May 12 - 17 2019.

[23] Ngoc-Quan Pham, Alex Waibel, and Jan Niehues, "Adaptive multilingual speech recognition with pretrained models," *arXiv preprint arXiv:2205.12304*, 2022.

[24] Ngoc-Quan Pham, Tuan-Nam Nguyen, Sebastian Stüker, and Alex Waibel, "Efficient weight factorization for multilingual speech recognition," in *Proceedings of INTERSPEECH 2021*, Brno, Czechia, August 30 – Sep 3 2021.

[25] Mu Yang, Andros Tjandra, Chunxi Liu, David Zhang, Duc Le, John HL Hansen, and Ozlem Kalinli, "Learning asr pathways: A sparse multilingual asr model," *arXiv preprint arXiv:2209.05735*, 2022.

[26] Surabhi Punjabi, Harish Arsikere, Zeynab Raeesy, Chander Chandak, Nikhil Bhave, Ankish Bansal, Markus Müller, Sergio Murillo, Ariya Rastrow, Sri Garimella, et al., "Streaming end-to-end bilingual asr systems with joint language identification," *arXiv preprint arXiv:2007.03900*, 2020.