INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Towards multi-task learning of speech and speaker recognition

Nik Vaessen[1], *David A. van Leeuwen*[1]

[1]Institute for Computing and Information Sciences, Radboud University

nvaessen@science.ru.nl, dvanleeuwen@science.ru.nl

Abstract

We study multi-task learning for two orthogonal speech technology tasks: speech and speaker recognition. We use wav2vec2 as a base architecture with two task-specific output heads. We experiment with different architectural decisions to mix speaker and speech information in the output sequence as well as different optimization strategies. Our multi-task learning networks can produce a shared speaker and speech embedding, which on first glance achieve a performance comparable to separate single-task models. However, we show that the multi-task networks have strongly degraded performance on out-of-distribution evaluation data compared to the single-task models. Code and model checkpoints are available at `https://github.com/nikvaessen/disjoint-mtl`.

Index Terms: multi-task learning, speech recognition, speaker recognition, wav2vec2

1. Introduction

Speech and speaker recognition are, in a sense, orthogonal speech technology tasks. When we develop automatic speech recognition (ASR) systems, a very desirable property is speaker independence: we want the system to perform well irrespective of who uttered the words. Neural ASR models should learn to generate speech embeddings which have minimum variability when the same text is spoken by different speakers. In contrast, when developing speaker recognition (SKR) systems, a very desirable property is text independence: we want the system to perform well irrespective of what was said. Neural SKR models, then, should learn to generate speaker embeddings which have minimum variability when the same speaker utters different texts. We observe a dichotomy where ASR models should be invariant to who speaks while SKR models should be invariant to what is being said. This raises the question: is it possible to train a multi-task learning (MTL) model which can do both speaker and speech recognition, while its components respectively need to be invariant to who is speaking, and what is said?

Besides this interesting academic question, fully-fledged ASR applications often involve speaker recognition components, in order to provide, e.g., speaker-attributed transcriptions, or speaker diarization. Moreover, in the past, speaker recognition results could be used to improve the performance of ASR models [1, 2]. Therefore, bringing ASR and SKR together into a single model could reduce the complexity of ASR applications, and has some promise for increased performance. However, we observe the following obstacles in bringing these tasks together:

1. Differences in neural architectures for respective tasks, although transformers are bridging this gap.
2. Datasets for ASR lack session variability, while datasets for SKR lack transcriptions.
3. ASR training must be carried out on complete utterances. Typically, ASR datasets do not have aligned transcriptions, while SKR network training is done on short segments as training on long utterances prevents generalization.

We choose to build on top of the wav2vec2 framework, as the same architecture has been fine-tuned in a single-task learning (STL) setting to both ASR [3], and speaker recognition [4, 5], bridging the gap between neural architectures for ASR and SKR. Our proposed multi-task model is trained with LibriSpeech data for ASR and VoxCeleb for SKR. We train with disjoint steps, meaning batches only contain data from one of the two datasets. This also enables ASR training on complete utterances and SKR training on short segments. This allows us to answer the following research questions:

1. Can a transformer-based architecture perform ASR and SKR simultaneously?
2. Is it feasible to train an MTL model with state-of-the-art datasets for speech recognition and speaker recognition?
3. Can we train with the complete sentence as input for ASR while using short segments as input for SKR?

2. Background

2.1. Related MTL work

In [4] the wav2vec2 network is used for multi-task learning between the speech tasks of speaker recognition and language identification. Their MTL model did not improve on baseline STL performances. In [6] consider whether ASR systems can benefit from MTL learning of speaker recognition, or whether adversarial learning [7] (AL) is more beneficial. Using the WSJ dataset [8] and a CNN model, they find similar, but small, improvement gains with MTL and AL. Also, [9] train a MTL speech and speaker recognition network on WSJ. They use two interconnected LSTMs, one for each task. The output of each LSTM is shared in the next time step. In [10] an LSTM is trained for ASR, with SKR as auxiliary task, on the TIMIT dataset [11]. Lastly, the recent Whisper model [12] is a multi-task transformer model with impressive ASR performance, which is also capable of doing speech activity detection, language identification and speech translation, but notably, no speaker recognition.

2.2. Wav2vec2

An important aspect of the wav2vec2 framework [3] is the application of self-supervised learning to initialize the network weights based on unlabeled data, before fine-tuning the network on (a smaller amount of) labeled data. In this work, we limit ourselves to fine-tuning the network in a multi-task configuration. Further details on the self-supervised learning aspect can be found in the seminal work [3].

The wav2vec2 architecture consists of three components. First, a 1-d feature extractor CNN processes a raw audio waveform $\mathcal{X} = x_1, \ldots, x_n$ into frames of speech features $\mathcal{Z} = z_1, \ldots, z_m$, with a window size of 20 ms. These features are projected, potentially masked in the time and feature dimension to mimic SpecAugment [13] regularisation, and a relative positional embedding is added. The resulting sequence of input vectors, with a receptive field of 2.5 s, are processed by an encoder network [14] with multi-head attention transformer layers [15] to produce a sequence of output vectors $\mathcal{C}^L = c_1^L, \ldots, c_m^L$, where L specifies the output sequence of a specific transformer layer. The output sequence (of any layer, but usually the last one) can be used by a downstream task.

For ASR, the output vectors of the wav2vec2 network can represent phones or letters. A single fully-connected (FC) layer can be used to classify each vector, and with CTC loss [16] the network is trained end-to-end. For SKR, the output vectors are pooled into a fixed-length speaker embedding [4, 5]. The network is trained end-to-end by classifying speaker identities using the speaker embedding and a single FC layer.

3. Methodology

3.1. MTL network architectures

3.1.1. Two task-specific heads

Throughout the work we only use the BASE wav2vec2 network architecture with 12 transformer layers. We only make slight modifications for our multi-task purposes by adding two task-specific heads; one for speech recognition, and one for speaker recognition. The automatic speech recognition head consists of a single FC layer which predicts a softmax probability distribution over the vocabulary, for each wav2vec2 output token in the sequence $\mathcal{C}^{12}$. This is equivalent to the original ASR design [3]. The speaker recognition head consists of two components. The first part transforms the output sequence into a speaker embedding. The second part, only used during training, is a single FC layer used to classify the train speakers with the speaker embedding. We consider both heads using $\mathcal{C}^{12}$ as input, which implies $\mathcal{C}^{12}$ contains speaker and speech information. However, we also experiment with using $\mathcal{C}^n$ as input for the speaker head instead. In this configuration, the network can gradually remove speaker information from $\mathcal{C}^{n+1}$ onward. We chose layer $n = 6$ so that half of the network can be solely focused on speech recognition.

3.1.2. Speaker embeddings

We compare three strategies to extract a speaker embedding from an output sequence $\mathcal{C}^n$. The first, *mean pooling*, simply aggregates each dimension of the wav2vec2 output vectors $c_1^n, \ldots, c_m^n$ over the time-axis [4]. The second, *first pooling* [5], does not consider the actual output sequence. Instead, we simply take the first token c_1^n as a speaker embedding. As a third variant, we use the ECAPA-TDNN [17] architecture to compute a speaker embedding, with $\mathcal{C}^n$ as input to ECAPA-TDNN, similar to WavLM [18]. Note that by using mean pooling or ECAPA-TDNN, there needs to be speaker information throughout the output sequence, while for first pooling the speech and speaker information can be separated by the transformer layers.

3.2. Optimization

We want to train the network on state-of-the-art datasets for speaker and speech recognition. In this section, we suggest two methods for MTL training for speech and speaker recognition. These are based on using Librispeech [19], a well-known dataset for speech recognition, and VoxCeleb [20, 21], a well-known speaker recognition dataset.

3.2.1. Disjoint training

In order to train with LibriSpeech and VoxCeleb, we propose to optimize our network with a disjoint forward step. We assume two datasets, D_s and D_k, base network weights θ_b, speech head weights θ_s and speaker head weights θ_k. We also have a base network function N, a speech recognition head function H_s with loss function L_s as well as a speaker recognition head function H_k with loss function L_k.

Each iteration i, we sample a speech batch $(x_s^{(i)}, y_s^{(i)}) \in D_s$ and a speaker batch $(x_k^{(i)}, y_k^{(i)}) \in D_k$. We then apply two forward passes, one on the speech batch, and one on the speaker batch, where we write $p \in \{s, k\}$:

$$\begin{aligned} q_p^{(i)} &= N(x_p^{(i)}, \theta_b^{(i)}) \\ \hat{y}_p^{(i)} &= H_p(q_p^{(i)}, \theta_p^{(i)}) \\ L_p^{(i)} &= L_p(y_p, \hat{y}_p^{(i)}) \end{aligned}$$

The total loss $L^{(i)}$ is a weighted sum over speech and speaker loss

$$L^{(i)} = \lambda_s^{(i)} L_s^{(i)} + \lambda_k^{(i)} L_k^{(i)} \tag{1}$$

with $\lambda_{s,k}$ the weights for speech and speaker and $\lambda_s + \lambda_k = 1$. The gradients for the different parts of the network become

$$\begin{aligned} \nabla_{\theta_k} L^{(i)} &= \lambda_k \nabla_{\theta_k} L_k^{(i)} \\ \nabla_{\theta_s} L^{(i)} &= \lambda_s \nabla_{\theta_s} L_s^{(i)} \\ \nabla_{\theta_b} L^{(i)} &= \lambda_k \nabla_{\theta_b} L_k^{(i)} + \lambda_s \nabla_{\theta_b} L_s^{(i)} \end{aligned} \tag{2}$$

The weights for the next iteration $\theta_b^{(i+1)}$, $\theta_s^{(i+1)}$ and $\theta_k^{(i+1)}$ are obtained with an optimizer step such as Adam.

3.2.2. Joint training

Most work on MTL assumes training can be done with a 'joint' forward step, namely each sample has labels for all tasks. As a baseline, we want to see if training with joint forward steps is effective for MTL of ASR and SKR. We tried two options. The first is to use only data from LibrisSpeech, which has both labels. The second option is to use an ASR model to generate labels for the whole VoxCeleb dataset. We decided to do this with the base[1] Whisper [12] model. We skip any data labeled as non-English by the Whisper model during training, and normalize the transcript to the character vocabulary of LibriSpeech.

3.2.3. Length of audio input during training

We hypothesize that the discrepancy between audio input lengths for ASR and speaker recognition systems is a potential issue, as the encoder will observe drastically different sequence lengths for each task. We therefore suggest two strategies for cropping the speaker recognition audio segments. The first strategy follows the current paradigm [5, 17, 22, 23, 24] and uses crops of 2 s. The second strategy is to use crops of 10 s, a value closer to the average length of the audio in LibriSpeech.

4. Experiments

4.1. Data

We used the LibriSpeech [19] (LS) dataset to train and evaluate for speech recognition. The dataset consists of utterances from audio books, read by volunteers. We used all three train

[1]with `https://pypi.org/project/openai-whisper/`

Table 1: *Comparison of STL baselines versus joint and disjoint MTL training. Evaluation is done on in-distribution (LS, VOX) and out-of-distribution (HUB5, NIST) data. The second column indicates which training data was used - V2* indicates ASR labels were generated with Whisper.*

		ASR (WER %)		SKR (EER %)	
network	data	LS-to	HUB5	vox1-h	SRE08
STL					
ASR	LS	10.4	40	-	-
ASR	V2*	16.6	25	-	-
SKR (2s)	LS	-	-	33	42
SKR (2s)	V2	-	-	5.1	16
MTL (joint, full length samples)					
$\lambda_s = 0.5$	LS	15.3	48	36	40
$\lambda_s = 0.5$	LS+V2*	18.1	36	10.3	24
$\lambda_s = 0.9$	LS+V2*	17.5	36	7.2	26
MTL disjoint, 2 sec SKR samples					
$\lambda_s = 0.5$	LS	14.5	54	45	46
$\lambda_s = 0.5$	LS+V2	11.1	46	41	45
$\lambda_s = 0.9$	LS+V2	11.5	48	42	46
MTL disjoint, 10 sec SKR samples					
$\lambda_s = 0.5$	LS	13.6	49	36	44
$\lambda_s = 0.5$	LS+V2	11.1	80	4.8	39
$\lambda_s = 0.9$	LS+V2	11.2	84	4.7	27

subsets, for a total of 960 hours of training data with 2484 speakers. The training audio utterances have a mean of 12.3 seconds, and a std of 3.84 seconds. To minimize right-padding (with 0) in the speech batches, a batch was collected by sampling utterances with similar length. We used the *dev-other* subset to determine a validation word error rate (WER_{val}). Evaluation was done on the difficult *test-other* (LS-to) subset. The transcriptions were greedily decoded, we did not use a language model. We also create a trial list for dev-other and test-other for SKR. We use all possible pairs, excluding positive trials from the same session (book), and only including same-sex negative trials.

The VoxCeleb1 [20] and VoxCeleb2 [21] (V2) datasets were used to train and evaluate on speaker recognition. The datasets consist of videos of celebrities taken from YouTube. Each speaker has multiple recordings (videos), and each recording has multiple utterances. The VoxCeleb2 "dev" subset was used as training, validation, and development data. It has a total of 2305 hours of data, with 5994 speakers, and a mean utterance length of 7.79 seconds and a std of 5.22 seconds. We held-out 194 speakers (97 male/female) to create a development subset. From the remaining 5800 speakers we randomly selected at most two recordings for the validation subset to get a 98%/2% train/val split. For the development set we randomly created 100 k positive and 100 k negative trial pairs, making sure negative trials are same-sex and positive trials are from 2 different recording sources. Evaluation was done on the VoxCeleb1 dataset. We used the hard "VoxCeleb1-H (cleaned)" trial list (vox1-h). It has 1190 speakers, and each negative trial pair has the same sex and nationality. There is no speaker overlap between VoxCeleb1 and VoxCeleb2. During training and validation, all utterances are randomly cropped to either 2 or 10 seconds. During evaluation, we use the full length of the utterance unless stated otherwise. Trials are scored by computing the cosine similarity between two speaker embeddings, without any further processing.

Table 2: *Comparing three methods to extract speaker embeddings from wav2vec2. Evaluation is done on in-distribution (LS, VOX) and out-of-distribution (HUB5, NIST) data. We vary training with 2s/10s chunks and for MTL also using $\mathcal{C}^6$ or $\mathcal{C}^{12}$.*

	ASR (WER %)		SKR (EER %)	
SKR head	LS-to	HUB5	vox1-h	SRE08
STL, x/x implies training with 2s/10s SKR samples				
mean	-	-	5.1/5.1	17/13
first	-	-	5.4/5.2	19/14
ECAPA	-	-	6.3/5.8	21/13
MTL disjoint, 2 sec SKR samples, x/x implies $\mathcal{C}^6/\mathcal{C}^{12}$				
mean	13.5/13.4	53/52	21/34	40/44
first	13.6/13.9	52/53	12/34	29/40
ECAPA	13.2/13.9	45/53	9/35	25/39
MTL disjoint, 10 sec SKR samples, x/x implies $\mathcal{C}^6/\mathcal{C}^{12}$				
mean	12.9/12.8	51/79	3.9/4.0	31/33
first	13.2/13.4	46/79	3.9/4.0	15/16
ECAPA	13.2/12.7	42/83	4.2/4.7	19/16

To test on out-of-distribution (OOD) data, we also evaluate speech recognition on the English part of HUB5 2000, and speaker recognition on NIST SRE08 [25]. For HUB5, we segment the audio based on the ground truth reference to make evaluation easier. We also pre-process the text by removing all annotations and normalizing to the LibriSpeech character vocabulary. For SRE08 we use the 10 s trials for evaluation. For both datasets we resample the audio to 16Khz.

4.2. Training protocol

We use the following training protocol, unless stated otherwise, to balance between spending an equal amount of computational resources on each method, and limiting the required computational budget. Each network variant under study is initialized with available[2] self-supervised, pre-trained weights [3], with an identical random seed for all experiments. We use a batch size of up to 3.2 M audio samples ($\leq$ 200 seconds) for both tasks [3]. We use the default regularisation methods for wav2vec2, LayerDrop [26, 27] , Dropout [28], and SpecAugment masking [13]. The optimizer is Adam [29] and a tri-stage learning rate schedule [3] (10% warm up, 40% constant, 50% exponential decay). We clip gradients to $[-1, 1]$. For the first 3 k steps the whole wav2vec2 network is frozen, only the heads are updated [3]. The feature extractor CNN is always frozen [3]. We use CTC loss [16] as the speech recognition loss, and AAM softmax loss [30, 31] for the speaker recognition loss with a scale of 30 and a margin of 0.2 [17]. For each network variant we perform a grid search over the learning rates $\{1, 3\} \times 10^{-\{4,5,6\}}$ with 200 k steps. We stop early if the validation loss has not decreased for 40 k steps. We validate every 5 k steps. For the evaluation, we select an "optimal" model and learning rate based on $\frac{1}{4}\text{WER}_{\text{val}} + \frac{3}{4}\text{EER}_{\text{val}}$. Training is done on a machine with a single GPU[3], 40GB RAM and 12 CPU cores. In total 313 days of GPU time was spent on experiments.

4.3. Comparing MTL optimization strategies

The first set of experiments are focused on comparing optimization strategies and are shown in Table 1. The network architecture in these experiments is fixed; the speech and speaker

[2]The pre-trained weights were retrieved from `https://huggingface.co/facebook/wav2vec2-base`.

[3]Experiments were done on A5000, A6000 and A100 GPUs.

Table 3: *Evaluation of STL and MTL models on cross-disjoint-task and out-of-distribution data. MTL models are trained with $\lambda_s = 0.9$. V2* indicates ASR labels from Whisper during training. For ASR evaluation on Voxceleb we use the vox1-o test set, with labels from Whisper. For SKR evaluation we show results using only the first 2 seconds, or the full utterance of each audio file.*

		ASR			SKR (2 sec eval)			SKR (full sample eval)		
model	data	LS-to	vox1-o	HUB5	LS-to	vox1-h	SRE08	LS-to	vox1-h	SRE08
STL ASR	LS	10.4	35	40	-	-	-	-	-	-
STL SKR	V2	-	-	-	4.9	11.0	32	2.2	5.1	16
MTL joint	LS+V2*	17.5	27	36	13.4	21	41	8.5	7.2	26
MTL DJ 2	LS+V2	11.5	35	48	7.9	12.4	33	40	42	46
MTL DJ 10	LS+V2	11.2	100	84	44	16	41	42	4.7	27

head both use $\mathcal{C}^{12}$, and the speaker head uses mean pooling. First, observe that single-task training for SKR with LS achieves much worse performance compared to training with V2. It follows that the SKR performance with both joint and disjoint MTL optimization using only LS data is similar. When we do joint optimization with LS and whisper-transcribed V2, the speaker recognition performance drastically improves. Note that MTL training with LS data is worse than STL training with LS data for both SKR and ASR. Looking at disjoint MTL training, we see that using 2 s SKR chunks during training seemingly leads to no speaker recognition capabilities (discussed further in Section 4.5). Using 10 s SKR chunks however, makes the MTL outperform the STL baseline on the vox1-h test set, with slightly degraded ASR performance on the LS test set. We also see that the choice of $\lambda_s = 0.9$ versus $\lambda_s = 0.5$ trades-off SKR and ASR performance. Lastly, we observe that all MTL models have drastically degraded performance on out-of-distribution test data (Hub5, NIST) compared to the STL baselines.

4.4. Varying architectures

In the second set of experiments we focus on different strategies for extracting speaker information for SKR, and effectively combining it with the speech information for ASR. For all MTL experiments we use $\lambda_s = 0.5$ and only train with disjoint steps. We train with either 2 or 10 s SKR chunks, and place the speaker head at either $\mathcal{C}^6$ or $\mathcal{C}^{12}$. The speech head is always at $\mathcal{C}^{12}$. We also apply gradient clipping after summing the gradients instead of before. Table 2 shows the results. The first observation is that STL speaker recognition actually has better performance when using 10 s chunks during training, more noticeably on NIST data. Secondly, the specific variant of the speaker head has only a minor effect on the ASR performance. However, using ECAPA-TDNN on $\mathcal{C}^6$ seems very effective compared to mean or first pooling. Noticeably, when training with 2 s second chunks, using $\mathcal{C}^6$ instead of $\mathcal{C}^{12}$ seems to result in some SKR capabilities. ASR performance on LS is also worse compared to Table 1 with equivalent architectures, likely due to changing the clipping strategy.

4.5. Different evaluation conditions

In this section we further analyze the results described in Table 1. As we observed decreased performance on out-of-distribution data for the MTL models, we also wanted to observe the performance on cross-disjoint-task data, namely, can we do SKR on LS data, and ASR on V2 data? To evaluate for ASR on V2 we use the transcribed whisper output as ground truth. In Table 3 we see that this is not always the case. Noticeably, disjoint MTL with 10 s chunks has a 100% WER on VoxCeleb data and a 42% EER on LibriSpeech data. Furthermore, we observed that MTL disjoint training with 2 s SKR chunks and mean pooling did not show any SKR capabilities. Therefore, perhaps counter-intuitively, we also evaluate on SKR by

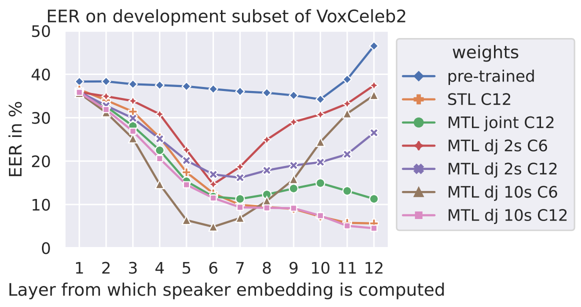

Figure 1: *The performance of each wav2vec2 transformer layer by mean pooling the output sequence before (pre-trained, in blue) and after fine-tuning wav2vec2 (STL and 5 MTL variants). Fine-tuning used speaker head with mean-pooling at $\mathcal{C}^6$ or $\mathcal{C}^{12}$.*

only using the *first* 2 s of the utterance, instead of the whole utterance. We observe that for STL SKR, MTL joint, and MTL disjoint with 10 s chunks, the SKR performance is worse when using only the first 2 s of the audio compared to using the full sample. However, MTL disjoint training with 2 s chunks has decent performance when also evaluating with 2 s of audio. This compares to no capabilities when evaluating on the full sample. Lastly, in Figure 1 we show how the speaker information is distributed over the network layers. We see that MTL models with a speaker head using $\mathcal{C}^{12}$ actually lose speaker information after $\mathcal{C}^6$, indicating the models attempt to separate speech and speaker information.

5. Conclusion

We have shown that creating an MTL model for speech and speaker recognition is challenging. First, we need multi-labelled data with session variability, LibriSpeech is not sufficient for creating a good SKR model. Our mitigation strategies with either automatic labels, or disjoint training, have drawbacks. Optimizing a model with disjoint steps doesn't generalize to OOD data. We further saw that MTL models have increased SKR performance, at the cost of decreased ASR performance. It is hard to include speaker information without harming ASR performance. This might be inherent to the MTL loss function, which always needs to trade-off the CTC loss versus the AAM-softmax loss. We believe that future work could focus on integrating speaker information into the CTC loss, by adding e.g., speaker-related targets, and foregoing the need to use two loss functions and two output heads.

6. Acknowledgements

This work was sponsored by NWO - Domain Science for the use of supercomputer facilities (Snellius, `www.surf.nl`).

7. References

[1] M. F. BenZeghiba and H. Bourlard, "On the combination of speech and speaker recognition," in *Proc. 8th European Conference on Speech Communication and Technology (Eurospeech 2003)*, 2003, pp. 1361–1364.

[2] V. Peddinti, G. Chen, V. Manohar, T. Ko, D. Povey, and S. Khudanpur, "Jhu aspire system: Robust lvcsr with tdnns, ivector adaptation and rnn-lms," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 539–546.

[3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[4] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on Speaker Verification and Language Identification," in *Proc. Interspeech 2021*, 2021, pp. 1509–1513.

[5] N. Vaessen and D. A. Van Leeuwen, "Fine-tuning wav2vec2 for speaker recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7967–7971.

[6] Y. Adi, N. Zeghidour, R. Collobert, N. Usunier, V. Liptchinsky, and G. Synnaeve, "To reverse the gradient or not: An empirical comparison of adversarial and multi-task learning in speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3742–3746.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[8] D. B. Paul and J. Baker, "The design for the wall street journal-based csr corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[9] Z. Tang, L. Li, and D. Wang, "Multi-task recurrent model for speech and speaker recognition," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2016, pp. 1–4.

[10] G. Pironkov, S. Dupont, and T. Dutoit, "Speaker-aware long short-term memory multi-task learning for speech recognition," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1911–1915.

[11] J. S. Garofolo, "Timit acoustic phonetic continuous speech corpus," *Linguistic Data Consortium, 1993*, 1993.

[12] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[13] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. ACL 2019*. Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[16] A. Graves, "Connectionist temporal classification," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, pp. 61–93.

[17] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in *Proc. Interspeech 2020*, 2020, pp. 3830–3834.

[18] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[20] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A Large-Scale Speaker Identification Dataset," in *Proc. Interspeech 2017*, 2017, pp. 2616–2620.

[21] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep Speaker Recognition," in *Proc. Interspeech 2018*, 2018, pp. 1086–1090.

[22] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *ICASSP*, 2018.

[23] W.-W. Lin and M.-W. Mak, "Wav2spk: A simple DNN architecture for learning speaker embeddings from waveforms." in *Proc. Interspeech*, 2020, pp. 3211–3215.

[24] J. S. Chung, J. Huh, S. Mun, M. Lee, H. Heo, S. Choe, C. Ham, S. Jung, B. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 2977–2981.

[25] A. F. Martin and C. S. Greenberg, "Nist 2008 speaker recognition evaluation: Performance across telephone and room microphone channels," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[26] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*. Springer, 2016, pp. 646–661.

[27] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *arXiv preprint arXiv:1909.11556*, 2019.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[30] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[31] Y. Liu, L. He, and J. Liu, "Large Margin Softmax Loss for Speaker Verification," in *Proc. Interspeech 2019*, 2019, pp. 2873–2877.