A versatile and efficient approach to summarize speech into utterance-level representations

João Monteiro[1], **Jahangir Alam**[1,2], **Tiago Falk**[1].
1-INRS-EMT, University of Quebec
2-Centre de Recherche Informatique de Montreal (CRIM)
joao.monteiro@inrs.ca, jahangir.alam@crim.ca, tiago.falk@inrs.ca

Abstract

Time delay neural networks (TDNN) have become ubiquitous for voice biometrics and language recognition tasks relying on utterance-level speaker- or language-dependent representations. In this paper, we discuss directions to improve upon the conventional TDNN architecture to render it more generally applicable. More specifically, we explore the utility of performing pooling operations across different levels of the convolutional stack and further propose an approach to efficiently combine such set of representations. We show that the resulting models are more versatile, in the sense that a fixed architecture can be re-used across different tasks, and learned representations are more discriminative. Evaluations are performed across two settings: (1) two sub-tasks for spoofing attack detection, and (2) three sub-tasks for spoken language identification. Results show the proposed design yielding improvements over the original TDNN architecture, as well as other previously proposed methods.

1 Introduction

Time delay neural networks (TDNN) have been widely employed in speech processing applications, most notably within the space of voice biometrics. For example, in the x-vector setting [15], TDNNs yielded strong performance for speaker verification tasks, serving as a front-end to extract utterance-level representations. TDNNs consist of a sequence of dilated 1-dimensional convolution layers which operate across the temporal dimension. The convolutional stack is followed by a *temporal pooling layer*, which concatenates component-wise first- and second-order statistics over the time axis. The outputs of the pooling layer are finally passed through two fully-connected layers to yield outputs corresponding to conditional log-probabilities over the set of training speakers or languages. The *temporal pooling* operation is intended to enable the computation of global utterance-level representations of the audio input and yield a single vector representing an entire sequence of acoustic features from input signal. From a practical perspective, global representations can be useful for tasks where the ability to process sequences of varying lengths is required, such as speaker and/or spoken language identification. However, including a pooling operation within a feed-forward architecture brings in challenges and limitations, including: (1) *Overly specialized architectures*: lower-level (i.e., closer to inputs) or high-level (i.e., closer to outputs) learned representations might be more or less effective depending on the underlying task/data of interest. For instance, it's unlikely that the same type of representation would be useful for tasks such as speaker verification and language identification, since speaker-dependent cues are generally independent of the underlying phonetics within a signal, while determining languages does require phonetic information to be salient in learned representations. Given those variations across tasks, we argue determining the right *level of abstraction* of learned representations, mostly via deciding where to perform global pooling operations, is task-dependent and, as such, requires the design of a specific architecture for each different task. (2) *Ignoring complementary information*: TDNNs perform pooling operations only

Efficient Natural Language and Speech Processing workshop at (NeurIPS 2021).

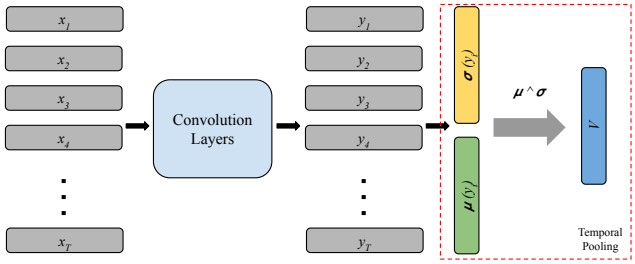

Figure 1: Conventional TDNN overview.

after the output of the final convolutional layer, thus global features from other levels of the model are not explicitly accounted for. Such operation could discard potentially discriminatory information for downstream tasks. We argue that a solution that simultaneously accounts for pooled features across different parts of the model has the potential to yield more discriminative learned representations, resulting in more generalizable models.

In order to address these limitations, we propose to modify the TDNN architecture and compute the pooling operation independently across the five convolution layers of the model. Moreover, we propose the use of a self-attentive layer [19] to give the model the ability to select, at training time, the best combination scheme between features obtained at different levels and to compute a novel sequence of global vectors as a function of the entire set of representations. Finally, a last pooling operation is applied on the resulting sequence to yield an utterance-level representation. The proposed scheme offers the following advantages over conventional TDNNs: (1) *Versatility*: as opposed to designing a new architecture for each new task, the proposed architecture introduces a data-oriented approach that allows for the model to learn which layers provide more discriminative information for global pooling. As such, a single architecture can be re-used across different tasks, as will be observed in the evaluation section. (2) *Generality*: global factors in each layer are explicitly considered, as opposed to just the last convolutional layer. As such, complementary information obtained from different layers can be leveraged. We further highlight that such a scheme defines classes of models that contain simple aggregation mechanisms as particular cases; i.e., simple schemes such as averaging pooled representations from different levels, or selecting a specific layer can all be recovered by the proposed model if those are the best solutions for the task/data at hand. (3) *Learnability*: The pooling operation across layers acts as skip/residual connections, thus yielding loss landscapes that are easier to train against [3, 5].

In this paper, we gauge the versatility of the proposed architecture, referred to as multi-level self-attentive TDNN (ML-TDNN), on two end-to-end tasks: spoken language identification and spoofing attack detection. In both cases, various sub-tasks corresponding to different types of data and conditions are considered. Across all such cases, we find evidence supporting the claim that global information from low-level layers contains complementary information that can be leveraged at later stages of the model to improve performance.

2 Background and related work

The proposed model builds upon the original TDNN architecture illustrated in Figure 1 and further detailed in Table 5 in the appendix. In details, an input sequence of length T is denoted as $x_{[1:T]}$, where each $x_i \in \mathbb{R}^d, i \in [T]$, represents a feature vector of dimension d (e.g., mel-frequency cepstral coefficients) at a given time frame. Equivalently, we denote the set of features output by the stack of convolution layers by $y_{1:T} \in \mathbb{R}^D, i \in [T]$, where D corresponds to the number of output channels of the last convolutional layer. We refer to those as local descriptors of the overall audio given that they correspond to features of a relatively short time window. D is set to 1500 in the original model and to 512 in our case since, as will be discussed, our setting requires local descriptors of matching dimensionality across the model. The temporal pooling layer, also referred to as *statistical pooling*, concatenates element-wise estimates of first- and second-order statistics of the set of local descriptors across the temporal axis. We thus define the global descriptor V, i.e., the feature vector summarizing the entire input sequence $x_{[1:T]}$, by the following:

$$V = \text{cat}[\mu(y_i), \sigma(y_i)], \tag{1}$$

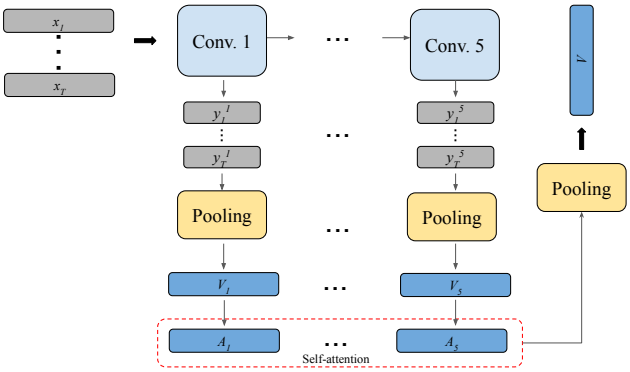

Figure 2: Proposed TDNN with multi-level self-attentive temporal pooling.

where the operator $v = \text{cat}[v_1, v_2]$ concatenates its two arguments $v_1, v_2 \in \mathbb{R}^D$ such that $v \in \mathbb{R}^{2D}$, and y_i are obtained after the last convolution layer. The global descriptor V is finally fed into a sequence of dense layers to yield the outputs corresponding to log-probabilities over the set of classes under consideration (e.g., training speakers or languages).

Past work has considered several modifications of the original TDNN architecture in order to improve performance in tasks such as speaker verification. In [22], for instance, several practical considerations are evaluated, such as the differences in performance given by applying batch normalization before or after activation functions are applied. Model variations focusing specifically in the pooling strategy were proposed, for instance, in [21], where a learnable gating mechanism is employed in order to assign more or less importance to specific frames prior to pooling. Similarly, a linear layer is used in [12] to learn how important individual frames are. Those approaches, however, are limited in that each frame is evaluated by itself, and a better informed pooling strategy should leverage the sequence structure in order to decide which frames matter more or less.

Alternatively, approaches such as the model discussed in [20] consider higher order statistics during pooling, but similarly to the base case, only representations from a given layer are used to compute the overall utterance-level representation used by top layers of the model. Closer to our work is the approach discussed in [16] where global statistics of different layers are concatenated prior to the TDNN dense layers close to the outputs. While such direction seems as an improvement compared to other approaches, given that bottom layers are considered as well, we hypothesize that simply concatenating low-level representations is sub-optimal in that the sequential nature of such set of feature vectors is not leveraged. We argue that some type of sequence processing mechanism can enrich the final pooled vector and yield lower dimensional pooled features, thus saving on parameters and computations in higher layers. While any sequence model could be used (e.g., RNNs), we focus on self-attention given its efficiency and ease of training.

3 Proposed Method

Given the conventional TDNN architecture described previously, one can observe that the time pooling operation performed in the end of the convolutional stack is the only model component able to summarize the content of the sequence into a global representation. We argue that this is: (1) *too restrictive*, as it ignores valuable information available in earlier layers of the model, and (2) *too specific*, since different tasks should require designers to search for the right layer where to apply the pooling operation. As such, we propose a multi-level pooling scheme (c.f. Figure 2 for an illustration). In this case, we extract a sequence of local descriptors $y^k_{[1:T]}$ for each convolution layer, i.e., $k \in [1, 2, ..., 5]$. The same temporal pooling operation described above is now performed across every layer k, thus yielding a sequence of global descriptors denoted $V_{[1:5]}$.

Next, we employ a self-attention layer [19] so that each V_k can be taken into account depending on how relevant they are in order to result in discriminative representations. In other words, we take advantage of the depth-wise set of global summaries of the input sequence by including a sequence

modeling component into the architecture. Such self-attention component, also referred to as *scaled dot product attention*, is given by the following:

$$\text{self-attention}(Q, K, V') = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V', \tag{2}$$

where Q, K, and V', denominated queries, keys, and values, respectively, each correspond to a linear transformation of $V_{[1:5]}$ (assumed to be a matrix of dimension $5 \times D$), i.e.: $Q = V_{[1:5]}W^Q, \quad K = V_{[1:5]}W^K, \quad V' = V_{[1:5]}W^V$.

In this case, each of the matrices W^Q, W^K, and W^V, have dimension $5 \times d_k$ and their entries are treated as learnable parameters. Intuitively, the scaled dot-products $\frac{QK^T}{\sqrt{d_k}}$ define weights indicating the importance of each element in the sequence for a specific data instance. Finally, we make use of the multi-head setting and employ the self-attention operation described above multiple times using independent sets of parameters. As usual, the final sequence is given by a linear projection of the concatenation of the outputs of each head to the space of dimension d_k. The number of self-attention heads is treated as a hyperparameter, and we empirically found the value of 16 to yield good results for the evaluation tasks considered herein. It is important to emphasize that even though the computation cost of self-attention layers scales with the square of the input length, the proposed model is favoured by the fact that the sequence length is fixed and moderate, as it corresponds to the depth of the convolutional stack (i.e., 5). Moreover, such layer defines a class of models that is rich enough to include as particular cases anything ranging from complicated non-linear relationships between elements to modeling schemes as simple as averaging/selection, depending on what is needed for the task at hand. However, any alternative sequence modeling layer can be used in this case yielding variations of the proposed model. We refer to the sequence output by the self-attentive transformation as $A_{[1:5]}$. Lastly, in order to combine the set of global descriptors $A_{[1:5]}$ into a single representation, we make use of a final statistical pooling layer that operates across the depth of the network, rather than across time. This yields the global descriptor: $V = \mu(A_k)$. We then proceed as usual and feed V into a sequence of fully-connected components leading to the output layer. Training is carried out so that the parameters of all the described components are learned jointly.

4 Evaluation and Discussion

We perform evaluations and comparisons across a number of tasks aiming to show evidence to support the following claims: **I**-A single architecture can be re-used across a set of distinct tasks while achieving high prediction performance as well as yielding effective representations for open-set tasks; **II**-Global features collected from low-level layers contain complementary information that can be leveraged to improve accuracy. We further remark that, since self-attentive aggregation includes as particular cases simple schemes such as averaging representations obtained from all layers as well as selecting specific layers to be used, we focus our computation budget on baseline systems that do not explicitly use any global summary of low-level representations. To further gauge the effectiveness of the proposed methods, results are compared with approaches recently reported in the literature.

4.1 Detecting spoofing attacks

The evaluation setting introduced for the ASVspoof 2019 challenge [18] is considered here. In particular, two independent sub-tasks relative to the detection of two different classes of attacks are used for evaluation: (1) *Logical access attacks*: Consisting of synthetic speech created using both voice conversion and text-to-speech systems, and (2) *Physical access attacks*: Consisting of simulated replays of genuine audio clips exhaustively considering varied acoustic conditions such as three different room sizes, three distances to the microphone, and three levels of reverberation. Statistics of dataset are shown in Table 1 and further detailed in [18]. We further highlight that a disjoint set of speakers is used to generate different data partitions and the algorithms make no use of speaker identity information.

For end-to-end detection of spoofing attacks, binary classifiers are trained following the strategies discussed in [7]. More specifically, audio features are extracted such that 30 linear frequency cepstral coefficients [14] (LFCC) stacked with delta and double-delta coefficients are used for logical access attack detection. For physical access attacks, in turn, product spectra (ProdSpec) [1] with

Table 1: Number of genuine and spoofing recordings in training, development, and evaluation partitions for logical and physical access attacks.

	# Speakers	# Recordings			
		Logical Access		**Physical Access**	
		Bona fide	**Spoof**	**Bona fide**	**Spoof**
Train	20	2580	22800	5400	48600
Development	20	2548	22296	5400	24300
Evaluation	67	7355	63882	18090	116640

Table 2: Detection performance on the evaluation sets of ASVspoof 2019.

		Logical access attacks		*Physical access attacks*	
	System	**EER (%)**	**min-tDCF**	**EER (%)**	**min-tDCF**
Third party baselines	CQCC-GMM [18]	9.57	0.2366	11.04	0.2454
	LFCC-GMM [18]	8.09	0.2116	13.54	0.3017
Our baselines	CQCC-GMM	8.91	0.2157	11.16	0.2478
	ivector/PLDA	16.55	0.4201	10.18	0.2687
	ResNet [10]	6.38	0.1423	1.98	0.0579
	TDNN [9]	7.00	0.1653	1.77	0.0597
Proposed	ML-TDNN	**6.07**	**0.1327**	**1.32**	**0.0470**

257 frequency bins are used. Train data is augmented offline following the approach in [9] and the sampling approach proposed in [7] to deal with unbalanced classes is further employed. The development data is used for cross-validation during model selection and hyperparameter tuning. We report performance in terms of the the equal error rate (EER) and the normalized minimum tandem detection cost function (min-tDCF) [4] (see [18] for a detailed description and motivation for these evaluation metrics). Lower values of the metrics suggest improved performance. Experimental results are reported in Table 2 for both the logical and physical access attacks. In addition to the conventional TDNN approach [9], several other baselines are considered, including the two used in the ASVspoof 2019 challenge [18]. In particular, methods based on ResNets [3] trained on top of the same features, GMM-based classifiers trained with LFCC or constant-Q cepstral coefficients (CQCC), and a system based on the i-vector/PLDA combination [2, 13] are explored. As can be seen, the proposed ML-TDNN reduced EER and min-tDCF across both attack methods, thus better separating attacks and genuine samples. This can be due to an induced more well-conditioned loss landscape and/or access to global information in early layers close to the original data.

4.2 Spoken language identification

For the spoken language identification task, we consider the data and evaluation conditions introduced for the AP18-OLR Challenge discussed in [17]. The data, whose statistics are summarized in Table 3, correspond to audio from ten languages, and the following evaluation conditions were defined: **I**-*Short-duration*: Considers only test recordings with duration lower than 1 second, **II**-*Confusing languages*: Test trials correspond to pairs of languages known to be to difficult to distinguish, i.e., Cantonese, Korean, and Mandarin, and **III**-*Unseen languages*: Test recordings correspond to languages not observed within the training sample. A total of 214560, 22071, and 404160 test trials (i.e., a pair *claimed language/test recording*) were made available for each of the evaluation conditions discussed above. We train models using the multi-task method described in [8], where a metric-learning approach is used along with a standard maximum likelihood training strategy. The training loss is minimized using stochastic gradient descent and the same learning rate schedule discussed in [19] is employed. Evaluation is performed under the end-to-end setting so that predictions are made by directly forwarding data through the model, without using any external classifier. We do so by following the approach discussed in [6] where the output unit corresponding to the claimed class in a trial is used as a verification score. Regarding data preparation, pre-processing steps follow those discussed in [6].

Results are reported in Table 4 in terms of EER and the average cost performance (C_{avg}). Details about both metrics can be found in [17]. We highlight that the reported results were computed using the official scripts released for the AP18-OLR challenge. We further remark that we report results

Table 3: Language identification dataset statistics.

Language	Train		Evaluation	
	# Speakers	Utt./Speaker	# Speakers	Utt./Speaker
Cantonese	24	320	6	300
Mandarim	24	300	6	300
Indonesian	24	320	6	300
Japanese	24	320	6	300
Russian	24	300	6	300
Korean	24	300	6	300
Vietnamese	24	300	6	300
Kazakh	86	50	86	20
Tibetan	34	330	34	50
Uyghur	353	20	353	5

Table 4: Identification performance for the three evaluation conditions considered on the AP18-OLR challenge.

	Short-duration		Confusing		Unseen	
	EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
i-vector+Cosine [11]	18.02	0.178	10.71	0.107	7.77	0.058
i-vector+PLDA [11]	17.50	0.174	10.66	0.106	7.51	0.052
ResNet (Stats.) [11]	**10.85**	0.112	3.63	0.036	4.23	0.020
ResNet (Attention) [11]	10.97	0.111	4.34	0.043	4.58	0.023
ResNet (LSTM) [11]	11.76	0.115	3.34	0.032	4.00	0.021
TDNN	13.16	0.126	4.30	0.058	4.80	0.036
ML-TDNN (A)	11.17	**0.109**	3.23	0.033	4.20	0.023
ML-TDNN (B)	14.68	0.138	**2.80**	**0.029**	**3.53**	**0.016**
ML-TDNN (A+B)	10.23	0.101	2.42	0.025	3.06	0.015

obtained by two independent models, indicated by ML-TDNN (A) and (B) in Table 4, where each model relied on different validation datasets to tune their hyperparamaters. While model (A) used only the short-duration partition of the development data, model (B) used the complete development set. As can be seen, an apparent trade-off is found regarding the performance on the short-duration condition when using model (A), i.e., while using the short-duration development data improved accuracy under this condition, it degraded the performance on the other tasks. For practical implementations, we recommend using both models (A) and (B) in parallel and mixing their scores, as indicated by (A+B). Lastly, the proposed ML-TDNN method is shown to outperform the conventional TDNN across all three evaluation conditions. In fact, with the exception of the short-duration condition, the proposed ML-TDNN outperformed all benchmarks, including those relying on ResNets with recurrent pooling [11], which considers a complex training strategy involving pre-training steps of convolutional layers, shown to be important for short-duration conditions.

5 Conclusion

We introduced a variation of the TDNN architecture in which temporal pooling operations are performed across all layers of the convolutional stack rather than only at its end. Across the different tasks and datasets considered, results are consistent in showing the ML-TDNN to outperform various benchmarks that use global features obtained from a single layer, thus highlighting that complementary speaker/language-dependent information can be efficiently leveraged from low-level layers close to inputs. Moreover, baselines in each considered task consisted of models specialized to that particular task, which suggests the ML-TDNN is general enough to perform at least as well as specialized architectures across the considered evaluation conditions. As future work, we intend to evaluate the proposed model when it is used as an embedding encoder under the speaker verification setting.

References

[1] M. J. Alam, P. Kenny, G. Bhattacharya, and T. Stafylakis. Development of crim system for the automatic speaker verification spoofing and countermeasures challenge 2015. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[2] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4): 788–798, 2011.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] T. Kinnunen et al. t-dcf: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification. *arXiv preprint arXiv:1804.09618*, 2018.

[5] H. Li, Z. Xu, G. Taylor, and T. Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[6] J. Monteiro, J. Alam, and T. H. Falk. Residual convolutional neural network with attentive feature pooling for end-to-end language identification from short-duration speech. *Computer Speech & Language*, 2019.

[7] J. Monteiro, J. Alam, and T. H. Falk. End-to-end detection of attacks to automatic speaker recognizers with time-attentive light convolutional neural networks. *IEEE International Workshop on Machine Learning for Signal Processing*, 2019.

[8] J. Monteiro, M. J. Alam, and T. H. Falk. Combining speaker recognition and metric learning for speaker-dependent representation learning. In *INTERSPEECH*, pages 4015–4019, 2019.

[9] J. Monteiro, J. Alam, and T. Falk. A multi-condition training strategy for countermeasures against spoofing attacks to speaker recognizers. In *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, pages 296–303, 2020.

[10] J. Monteiro, J. Alam, and T. H. Falk. Generalized end-to-end detection of spoofing attacks to automatic speaker recognizers. *Computer Speech & Language*, page 101096, 2020.

[11] J. Monteiro, M. J. Alam, and T. Falk. On the performance of time-pooling strategies for end-to-end spoken language identification. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3566–3572, 2020.

[12] K. Okabe, T. Koshinaka, and K. Shinoda. Attentive statistics pooling for deep speaker embedding. *Proc. Interspeech 2018*, pages 2252–2256, 2018.

[13] S. J. Prince and J. H. Elder. Probabilistic linear discriminant analysis for inferences about identity. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[14] M. Sahidullah, T. Kinnunen, and C. Hanilçi. A comparison of features for synthetic speech detection. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[15] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust DNN embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, 2018.

[16] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou. Deep speaker embedding learning with multi-level pooling for text-independent speaker verification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6116–6120. IEEE, 2019.

[17] Z. Tang, D. Wang, and Q. Chen. Ap18-olr challenge: Three tasks and their baselines. *arXiv preprint arXiv:1806.00616*, 2018.

[18] M. Todisco et al. Asvspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv preprint arXiv:1904.05441*, 2019.

[19] A. Vaswani et al. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[20] S. Wang, Y. Yang, Y. Qian, and K. Yu. Revisiting the statistics pooling layer in deep speaker embedding learning. In *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. IEEE, 2021.

[21] L. You, W. Guo, L.-R. Dai, and J. Du. Deep neural network embeddings with gating mechanisms for text-independent speaker verification. *Proc. Interspeech 2019*, pages 1168–1172, 2019.

[22] H. Zeinali, L. Burget, J. Rohdin, T. Stafylakis, and J. H. Cernocky. How to improve your speaker embeddings extractor in generic toolkits. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6141–6145. IEEE, 2019.

A TDNN Architecture details

Table 5: Standard TDNN architecture. T indicates the duration of features in number of frames and d the feature vector dimensionality. Batch normalization is further employed after each layer except temporal pooling.

Layer	***Input Dimension***	***Output dimension***
Conv1d+ReLU	d $\times$ T	512 $\times$ T
Conv1d+ReLU	512 $\times$ T	512 $\times$ T
Conv1d+ReLU	512 $\times$ T	512 $\times$ T
Conv1d+ReLU	512 $\times$ T	512 $\times$ T
Conv1d+ReLU	512 $\times$ T	1500 $\times$ T
Temporal Pooling	1500 $\times$ T	3000
Linear+ReLU	3000	512
Linear+ReLU	512	512
Linear+ReLU	512	# classes