NEURAL ARCHITECTURE SEARCH FOR SPEECH EMOTION RECOGNITION

Xixin Wu[1]*, *Shoukang Hu*[1], *Zhiyong Wu*[1,2], *Xunying Liu*[1], *Helen Meng*[1]

[1] The Chinese University of Hong Kong, Hong Kong SAR, China
[2] Shenzhen International Graduate School, Tsinghua University, Shenzhen, China

ABSTRACT

Deep neural networks have brought significant advancements to speech emotion recognition (SER). However, the architecture design in SER is mainly based on expert knowledge and empirical (trial-and-error) evaluations, which is time-consuming and resource intensive. In this paper, we propose to apply neural architecture search (NAS) techniques to automatically configure the SER models. To accelerate the candidate architecture optimization, we propose a uniform path dropout strategy to encourage all candidate architecture operations to be equally optimized. Experimental results of two different neural structures on IEMOCAP show that NAS can improve SER performance (54.89% to 56.28%) while maintaining model parameter sizes. The proposed dropout strategy also shows superiority over the previous approaches.

Index Terms— Speech emotion recognition, neural architecture search, uniform sampling, path dropout

1. INTRODUCTION

Speech emotion recognition (SER) is an important contributor towards graceful human-machine interaction, as machines can generate appropriate responses according to human emotions in the contexts of interaction. With the applications of deep neural networks, SER research has made great progresses in the last decade [1–6]. The architecture designing of neural models in SER is generally reliant on expert knowledge and empirical (trial-and-error) evaluations. As explicit training and evaluation of model designs are often time-consuming and resource intensive, it becomes worthwhile to explore neural architecture search (NAS) techniques to automate the neural architecture design process. Hence, the field of NAS has attracted much research attention recently [7–12]. Generally, an NAS algorithm is designed to search architectures by combining operations from a predefined space based on certain evaluation metrics. The approaches based on reinforcement learning (RL) [7, 8], evolution [9–11, 13] or Bayesian optimization [14] demonstrate outstanding performance. However, these methods demand huge computational resources for system training and evaluation (e.g., 1800 GPU days [7]). Instead of searching over a discrete architecture space, the differentiable architecture search (DARTS) approach allows the space to be continuous, i.e., the categorical network operation choice is replaced with a softmax over all possible operations [12]. The architecture search task is then transformed to the learning of the softmax function outputs with gradient descent optimization, which significantly reduces the training costs (e.g., 4 GPU days [12]). The over-parameterized network, referred to as *supernet*, is composed of the combinations of operations with the architecture weights, i.e., the softmax outputs. As the candidate operation parameters and the architecture weights are optimized on the same supernet, the sub-optimal architectures may be prematurely learned at an early stage of the training. To encourage all candidate architectures to be optimized simultaneously, various techniques are applied, e.g., operation dropout [15], uniform path sampling [16, 17]. However, the training using operation dropout is often not stable, since certain nodes in the supernet may drop all operations. The training efficiency of uniform path sampling is low because at each training step only one chosen operation is optimized.

The NAS techniques have also been successfully applied to speech synthesis [18] and speech recognition [17, 19–25]. It has been shown that the performance can be improved and the model parameter sizes can be reduced at the same time. However, limited NAS research has been conducted in the SER area. In this paper, we investigate differentiable NAS for SER based on the two effective systems with the attention mechanism [3] and capsule networks [4], respectively. To further improve the efficient training of the candidate operations, we propose a simple yet effective strategy, *uniform path dropout*. Different from the sampling strategy that selects only one single path each time, our strategy drops a group of paths and selects the rest, so that the optimization can be accelerated, as multiple paths are optimized each time. Additionally, dropping paths also destroys the supernet's reliance on sub-optimal operations and forces the supernet to optimize other operations. Experimental results show that the proposed approach can improve emotion recognition performance by maintaining a moderate model parameter size. As far as we know, this is among the first efforts to investigate the effectiveness of NAS for the SER task.

*This research is supported by the CUHK Stanley Ho Big Data Decision Analytics Research Centre, the Centre for Perceptual and Interactive Intelligence, and National Natural Science Foundation of China (62076144).

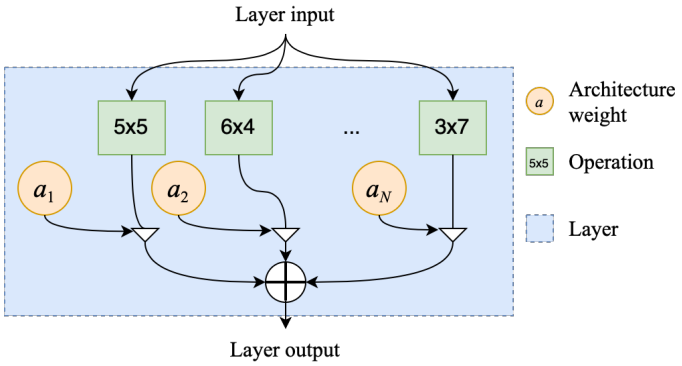


Fig. 1: A layer in the supernet is composed of candidate operations and the associated architecture weights. The operations, such as convolutional layers with different kernel sizes, are combined based on the architecture weights.

2. DIFFERENTIABLE NEURAL ARCHITECTURE SEARCH

The neural architecture search (NAS) techniques aim at automatically selecting neural architectures given a predefined search space, search algorithm and evaluation metrics [12, 26]. In this paper, we focus on the branch of differentiable architecture search (DARTS), which relaxes the search space to be continuous so that gradient descent optimization can be applied to the over-parameterized supernet. The supernet is built by connecting the layers that combine all the candidate operations, e.g., convolutional layers with different kernel sizes, in the search space, as shown in Fig 1. The operations in the layer are weighted combined as:

$$\mathbf{h}^l = \sum_{i=1}^{N^l} a_i^l \phi_i(\mathbf{h}^{l-1}; \mathbf{W}_i^l), \tag{1}$$

where $\mathbf{a}^l = \{a_1^l, a_2^l, ..., a_{N^l}^l\}$ is the architecture weights for the N^l candidate operations in the l-th layer. $\mathbf{h}^{l-1}$ and $\mathbf{h}^l$ are the layer input and output, respectively. ϕ_i is the i-th operation with the model parameters $\mathbf{W}_i^l$. Possible options for modeling the architecture weights in the supernet are using the softmax function or conducting proximal iterations [27].

2.1. Joint Optimization

The NAS can be performed by jointly optimizing the architecture weights $\mathcal{A} = \{\mathbf{a}^l\}_{l=1}^M$ and model parameters $\mathcal{W} = \{\mathbf{W}^l\}_{l=1}^M$ based on the training data:

$$\mathcal{A}^*, \mathcal{W}^* = \arg\min_{\mathcal{A},\mathcal{W}} \mathcal{L}_{\texttt{train}}(\mathbf{N}(\mathcal{A}, \mathcal{W})), \tag{2}$$

where M is the number of layers, $\mathcal{L}_{\texttt{train}}(\mathbf{N}(\mathcal{A}, \mathcal{W}))$ is training loss of the supernet $\mathbf{N}(\mathcal{A}, \mathcal{W})$. The final architecture is selected by connecting the operations corresponding to maximum architecture weights, which exhibits a connection to the network pruning techniques [28]. Jointly training the architecture weights and the model parameters saves time. However, the sub-optimal operations (e.g., simpler operations) may dominate the weights at an early stage, such that the optimal operations (e.g., operations with larger parameter sizes) may be ignored [17].

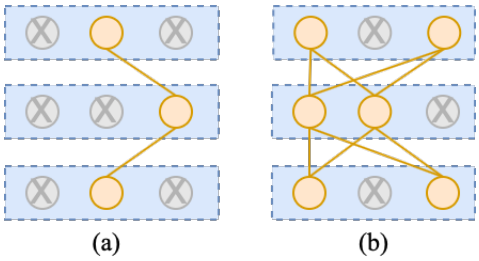


Fig. 2: Comparison of (a) uniform path sampling and (b) uniform path dropout. Only one path is selected via path sampling, while a group of paths is selected via path dropout.

2.2. Bi-level Optimization

Alternatively, a bi-level optimization can be used for NAS:

$$\mathcal{A}^* = \arg\min_{\mathcal{A}} \mathcal{L}_{\texttt{val}}(\mathbf{N}(\mathcal{A}, \mathcal{W}^*)), \tag{3}$$

$$\text{s.t. } \mathcal{W}^* = \arg\min_{\mathcal{W}} \mathcal{L}_{\texttt{train}}(\mathbf{N}(\mathcal{A}, \mathcal{W})), \tag{4}$$

where $\mathcal{L}_{\texttt{val}}$ is the validation loss. This decouples the training of the architecture weights and the model parameters. To solve this problem, [12] approximates the optimal model parameters in Eq. (3) via an one-step forward update. However, the learning rate for the one-step update needs to be carefully chosen. Alternatively, the uniform path sampling strategy is adopted for Eq. (4) by [16, 17]. The idea is to randomly select one path in the supernet (i.e. one candidate architecture) and then optimize the weights of the operations along the path. In this way, the weights of all candidate architectures are optimized by optimizing the supernet. However, since only the parameters of one architecture are optimized at each step, the training converges slowly.

To increase training efficiency, we propose to adopt a uniform path dropout strategy to randomly drop a group of paths and select the remaining paths, instead of selecting only one single path, as shown in Fig. 2. More specifically, by randomly masking a constant fraction (e.g., 1 out of 6) of operations in one layer, all the paths going through the masked operations are dropped, and the other paths are selected. The layer output is scaled up according to the mask fraction for training stability [29]. Selecting multiple paths aims at accelerating training. Dropping paths is necessary for destroying the supernet's reliance on certain sub-optimal operations and encouraging simultaneous optimization of the operation parameters (more discussion in Section 4.3).

Table 1: Configuration of the CNN component, attention layer, dense layer and the candidate operations for NAS. C, D, K, and W stand for channel number, dimension, kernel size and pooling window, respectively.

Layer	Structure	Operations
Conv2d_1 Conv2d_2	C=8, K=2×8 C=8, K=8×2	2×8,2×7,2×6, 1×9,1×10,3×5
Concat Max-pooling	Conv2d_1 + Conv2d_2 W=2×1	-
Conv2d_3	C=16, K=5×5	5×5,5×4,4×5, 4×4,4×6,6×4
Max-pooling	W=2×2	-
Conv2d_4	C=16, K=5×5	5×5,5×4,4×5, 4×4,4×6,6×4
Max-pooling Max-pooling	W=2×2 W=4×1	-
Attention (CNN_RNN_att)	C=64	32,48,64,80
Dense (CNN_SeqCap)	D=64	32,48,64,80

Different from the strategy by [28], where each operation is independently dropped, our dropout strategy is performed on the layer level and for each step a constant number of operations are dropped, so the risk of all operations are dropped can be avoided [16]. In our experiments, we found that our strategy is more stable during training because of constant number of dropout.

3. SYSTEM ARCHITECTURE

One representative among the various successful network structures for SER is convolutional neural networks (CNNs), which have demonstrated the effectiveness via learning neural hidden representations directly from spectrograms or waveforms [4, 30]. However, the convolutional layer configurations are designed mainly based on expert experience and evaluations. In this section, we describe the application of the NAS methods introduced in Section 2 to optimize the two systems of CNN_GRU_att [3] and CNN_SeqCap [4].

The CNN_GRU_att and the CNN_SeqCap both have the CNN module consisting of multiple convolutional and pooling layers to extract neural representations from spectrograms for the subsequent layers. Table 1 presents the configuration of the CNN module. Two separated convolutional layers with kernel of 2×8 and 8×2 are adopted. The outputs of these two separated layers are concatenated and passed through another two convolutional layers and three max-pooling layers. For the convolutional layers, we define three sets of candidate operations for the search space. The operations are convolutional layers with different kernel sizes, as shown in Table 1. We intend to include smaller kernel sizes in the sets as one of our goal is to reduce the model parameter sizes.

Upon the CNN module, the CNN_GRU_att system has a bi-directional gated recurrent unit (GRU) layer with 64 cells per direction. The final state of foward GRU and the first state of backward GRU are concatenated and fed to an attention layer that is composed of class-agnostic bottom-up and class-specific top-down attention maps [31]. The channel number for the attention layer adopted in [4] is 64. We define the set of $\{32, 48, 64, 80\}$ as candidate channel numbers of the attention layer for search.

For the CNN_SeqCap system, a sequential capsule layer is applied to the CNN module outputs [4]. These capsules obtained from 8 convolutional layers are routed to window-level capsule layer with 8 capsules of size 8 in each window. An utterance-level routing is conducted upon the window output vectors to produce 4 utterance-level capsules of size 16. The utterance-level capsules are then fed to two dense layers and softmax function. We define the set of $\{32, 48, 64, 80\}$ for the candidate dimensions of the first dense layer. The window used to slice the CNN module outputs is set to size of 40 input steps with shift of 20 steps. The routing iteration number is set to 3, and the number of masked operations is set to 1.

4. EXPERIMENTS AND ANALYSIS

4.1. Emotion Recognition Corpus

We conducted experiments on the benchmark corpus IEMOCAP [32], which consists of five sessions with two speakers in each session. The five-fold cross validation is adopted as [4,33]: 8 speakers from four sessions in the corpus are used as training data. One speaker from the remaining session is used as validation data, and the other one as test data. For NAS, the training data is used to train the model parameters and the validation data is used for optimizing the architecture weights. We evaluate on four emotions of the improvised data in the corpus, i.e., *Neutral*, *Angry*, *Happy* and *Sad*, following previous work [33, 34]. Spectrograms are extracted from the speech signal and split into 2-second segments, sharing the same utterance-level label. The training is conducted based on the 2-second segments, and the whole original spectrogram is used for evaluation. The spectrograms are extracted with 40-ms Hanning window, 10-ms shift and discrete Fourier transform (DFT) of length 16k, and normalized to have zero mean and unit variance.

4.2. Training Configuration

The Xavier initializer is used for the initialization of both the CNN components and the capsule layers. For the joint optimization mode, the supernet is trained for 60 epochs using the Adam algorithm ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e$-8) [35] with a dynamic learning rate scheme [4]. The learning rate is set to 0.001 in the first 3 epochs and reduced to 0.0005,

System	NAS	WA(%)	UA(%)	#param(k)
CNN_RNN_att	×	68.20	54.89	833
	Random	68.42	54.81	833
	Joint	68.61	55.59	833
	Sampling	**69.10**	54.23	835
	Dropout	68.87	**56.28**	**832**
CNN_SeqCap	×	69.86	56.71	704
	Random	68.77	54.75	700
	Joint	68.80	54.44	704
	Sampling	70.18	56.72	704
	Dropout	**70.54**	**56.94**	**700**

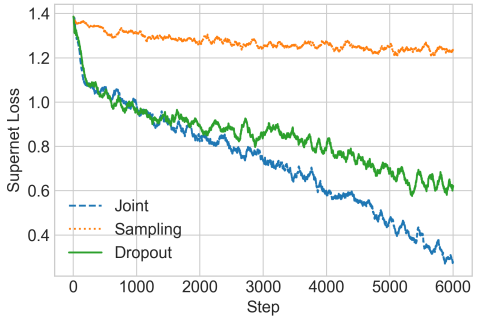
Table 2: Performance of the proposed and baseline systems.

0.0002 and 0.0001 gradually, when the average training loss is reduced by a factor of 10. The batch size is set to 16. For the bi-level optimization, the supernet is trained in the same way as the joint optimization, and the architecture weights are trained on the validation set using a constant learning rate of 0.001. The selected architecture is then trained again for 20 epochs and then optimized on the validation set with respect to the weighted accuracy.

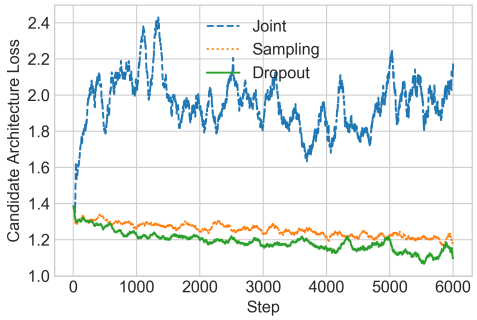
4.3. Experimental Results

We use two common evaluation metrics to evaluate system performance, i.e., weighted accuracy (WA) and unweighted accuracy (UA). WA is the accuracy of all samples in the test data, and UA is the average of class accuracies in the test set. The performance comparison of architectures found by NAS and the baseline systems are shown in Table 2. The random search, which selects the best performed architecture from 5 randomly sampled architectures, turns out to be a considerably strong baseline that achieves comparable performance with the system found by joint optimization. This coincides with the observations in previous literature [12, 17]. We need to also note that the baseline architectures are already optimized by the authors, which also provides a sufficiently good starting point for the random search. Table 2 also shows that both the uniform path sampling and dropout strategies are effective in boosting the performance to outperform the original models (without NAS) and the random search baselines. The proposed dropout strategy is better than the sampling strategy in terms of recognition performance. While the joint optimization is inferior to the two bi-level optimization strategies, it can still achieve better or comparable performance with the random search method.

To further analyze the behaviours of the sampling and dropout strategy, we plot the training loss curves of the supernet in Fig. 3 (a) and a candidate architecture randomly selected from the supernet in Fig. 3 (b). It is verified that the proposed path dropout strategy can accelerate the training via selecting multiple paths at each training step, as shown in



(a) Supernet loss



(b) Candidate architecture loss

Fig. 3: Training loss curves of (a) the supernet and (b) a randomly selected candidate architecture of CNN_SeqCap.

Fig. 3 (a) that the dropout loss curve is closer to the loss curve of joint optimization. While the sampling strategy slows down the decrease of the supernet loss, the candidate architecture loss curve is much lower than the joint optimization curve as shown in Fig. 3 (b). This implies that the sampling strategy can simultaneously optimize the parameters of various candidate operations. The dropout strategy produces an even lower loss curve than the sampling one, which demonstrates the strategy's superiority and provides insights for the performance gains achieved, as shown in Table 2.

5. CONCLUSIONS

In this paper, we investigate neural architecture search for speech emotion recognition to boost the recognition performance with reduced model parameter sizes. Experimental results suggest the effectiveness of the baseline and proposed search strategies. It is demonstrated that the uniform path dropout strategy can encourage all architecture operations to be equally optimized by randomly dropping paths, and increase the training efficiency by selecting multiple paths at each step. In the future, we will investigate more diverse search spaces and transferability of found architectures.

6. REFERENCES

[1] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Annual Conference of ISCA*, 2014.

[2] J. Lee and I. Tashev, "High-level feature representation using recurrent neural network for speech emotion recognition," in *Annual Conference of the ISCA*, 2015.

[3] P. Li, Y. Song, I. McLoughlin, W. Guo, and L. Dai, "An attention pooling based representation learning method for speech emotion recognition," in *Proc. of INTERSPEECH*, 2018, pp. 3087–3091.

[4] X. Wu, S. Liu, Y. Cao, X. Li, J. Yu, D. Dai, X. Ma, S. Hu, Z. Wu, X. Liu, and H. Meng, "Speech emotion recognition using capsule networks," in *Proc. of ICASSP*, 2019, pp. 6695–6699.

[5] T. Fujioka, T. Homma, and K. Nagamatsu, "Meta-learning for speech emotion recognition considering ambiguity of emotion labels.," in *Proc. of INTERSPEECH*, 2020, pp. 2332–2336.

[6] J. Wang, M. Xue, R. Culhane, E. Diao, J. Ding, and V. Tarokh, "Speech emotion recognition with dual-sequence LSTM architecture," in *ICASSP*, 2020, pp. 6474–6478.

[7] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," *ICLR*, 2017.

[8] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *ICML*, pp. 4095–4104, 2018.

[9] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[10] E. Real, S. Moore, A. Selle, and et al., "Large-scale evolution of image classifiers," in *ICML*, 2017, pp. 2902–2911.

[11] E. Real, A. Aggarwal, Y. Huang, and Q. V Le, "Regularized evolution for image classifier architecture search," in *Proc. of AAAI*, 2019, vol. 33, pp. 4780–4789.

[12] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. of ICLR*, 2018.

[13] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[14] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and De F., "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[15] G. Bender, P. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *ICML*. PMLR, 2018, pp. 550–559.

[16] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *Proc. of ECCV*, 2020, pp. 544–560.

[17] S. Hu, X. Xie, S. Liu, M. Cui, M. Geng, X. Liu, and H. Meng, "Neural architecture search for lf-mmi trained time delay neural networks," in *Proc. of ICASSP*. IEEE, 2021, pp. 6758–6762.

[18] R. Luo, X. Tan, R. Wang, T. Qin, J. Li, S. Zhao, E. Chen, and T. Liu, "Lightspeech: Lightweight and fast text to speech with neural architecture search," in *Proc. of ICASSP*. IEEE, 2021, pp. 5699–5703.

[19] T. Moriya, T. Tanaka, T. Shinozaki, S. Watanabe, and K. Duh, "Evolution-strategy-based automation of system development for high-performance speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 77–88, 2018.

[20] K. Jihwan, W. Jisung, K. Sangki, and et al., "Evolved speech transformer: Applying neural architecture search to end-to-end automatic speech transformer," *INTERSPEECH*, pp. 1788–1792, 2020.

[21] Y. Chen, J. Hsu, C. Lee, and H. Lee, "Darts-asr: Differentiable architecture search for multilingual speech recognition and adaptation," *INTERSPEECH*, pp. 1803–1807, 2020.

[22] L. He, D. Su, and D. Yu, "Learned transferable architectures can surpass hand-designed architectures for large scale speech recognition," in *ICASSP*, 2021, pp. 6788–6792.

[23] H. Zheng and et al., "Efficient neural architecture search for end-to-end speech recognition via straight-through gradients," in *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 60–67.

[24] Y. Liu, T. Li, P. Zhang, and Y. Yan, "Improved conformer-based end-to-end speech recognition using neural architecture search," *arXiv preprint arXiv:2104.05390*, 2021.

[25] X. Shi, P. Zhou, W. Chen, and L. Xie, "Darts-conformer: Towards efficient gradient-based neural architecture search for end-to-end asr," *arXiv preprint arXiv:2104.02868*, 2021.

[26] B. Zoph, V. Vasudevan, J. Shlens, and Q. V Le, "Learning transferable architectures for scalable image recognition," in *Proc. of CVPR*, 2018, pp. 8697–8710.

[27] Q. Yao, J. Xu, W. Tu, and Z. Zhu, "Efficient neural architecture search via proximal iterations," in *Proc. of AAAI*, 2020, vol. 34, pp. 6664–6671.

[28] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Proc. of NeurIPS*, vol. 28, 2015.

[29] 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.

[30] P. Tzirakis, J. Zhang, and B. W. Schuller, "End-to-end speech emotion recognition using deep neural networks," in *Proc. of ICASSP*. IEEE, 2018, pp. 5089–5093.

[31] R. Girdhar and D. Ramanan, "Attentional pooling for action recognition," in *Proc. of NeurIPS*, 2017, pp. 33–44.

[32] C. Busso, M. Bulut, C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N Chang, S. Lee, and S. S Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, pp. 335, 2008.

[33] A. Satt, S. Rozenberg, and R. Hoory, "Efficient emotion recognition from speech using deep learning on spectrograms.," in *Proc. of INTERSPEECH*, 2017, pp. 1089–1093.

[34] X. Ma, Z. Wu, J. Jia, M. Xu, H. Meng, and L. Cai, "Emotion recognition from variable-length speech segments using deep learning on spectrograms," in *Proc. of INTERSPEECH*, 2018, pp. 3683–3687.

[35] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proc. of ICLR*, 2015.