A Machine of Few Words
Interactive Speaker Recognition with Reinforcement Learning

Mathieu Seurin[1]*, Florian Strub*[2]*, Philippe Preux*[1]*, Olivier Pietquin*[3]

[1]Universit de Lille, CNRS, Inria - France
[2]Deepmind - France [3]Google Research, Brain Team - France

Contact author : mathieu.seurin@inria.fr

Abstract

Speaker recognition is a well known and studied task in the speech processing domain. It has many applications, either for security or speaker adaptation of personal devices. In this paper, we present a new paradigm for automatic speaker recognition that we call Interactive Speaker Recognition (ISR). In this paradigm, the recognition system aims to incrementally build a representation of the speakers by requesting personalized utterances to be spoken in contrast to the standard text-dependent or text-independent schemes. To do so, we cast the speaker recognition task into a sequential decision-making problem that we solve with Reinforcement Learning. Using a standard dataset, we show that our method achieves excellent performance while using little speech signal amounts. This method could also be applied as an utterance selection mechanism for building speech synthesis systems.

Index Terms: active speaker recognition, reinforcement learning, deep learning, iterative representation learning

1. Introduction

"Good words are worth much and cost little." - George Herbert

In many speech-based applications, the construction of a speaker representation is required [1]. Automatic Speaker Recognition (ASR) and Speech Synthesis are some examples that have recently made steady progress by leveraging large amounts of data and neural networks. Text-To-Speech (TTS) convincingly encompasses someone's voice [2], and modern Speaker Recognition systems identify a speaker [1] among thousands of possible candidates with high accuracy. Speaker recognition systems are trained to extract speaker-specific features from speech signals, and during evaluation, test speaker utterances are compared with the already existing utterances. However, dozen of test recordings are necessary, limiting usage when interacting with humans. When identifying a speaker or trying to create a convincing TTS system, only some key features might be necessary, such as certain inflexions or speech mannerisms. In this paper, we build a speaker recognition system that can identify a speaker by using a limited and personalized number of words. Instead of relying on full test utterance across all individuals, we interact with the speakers to iteratively select the most discriminative words.

Some pronunciation might be typical of certain speakers. For example, the phoneme 'r' might be pronounced differently depending on your accent. Thus starting with general phoneme and refining based on the utterances received could result in better recognition systems. More generally, a desirable feature of speaker recognition is to adapt its strategy to the current speaker as important features vary from person to person.

Here we propose to envision the problem of building a representation of the speaker as a sequential decision-making problem. The system we want to develop will select words that a speaker must utter so that it can be recognized as fast as possible. Reinforcement learning (RL) [3] is a framework to solve such sequential decision-making problems. It has been used in speech-based applications such as dialog [4, 5] but not to the problem of speaker identification. We adapt a standard RL algorithm to interact with a speaker to maximize the identification accuracy given as little data as possible. After introducing an Interactive Speaker Recognition (ISR) game based on the TIMIT dataset to simulate the speaker ASR interaction, we show that the RL agent builds an iterative strategy that achieves better recognition performance while querying only a few words.

Our contributions are thus:

1. to introduce the Interactive Speaker Recognition as an interactive game between the SR module and a human (Sec. 2);
2. to formalize ISR as a Markov Decision Process [6] so as to solve the problem with RL (Sec. 3);
3. to introduce a practical Deep RL ISR model, and train it on actual data (Sec. 4).

Finally, we test our method on the TIMIT dataset and show that ISR model successfully personalized the words it requests toward improving speaker identification, outperforming two non-interactive baselines (Sec. 5).

2. Interactive Speaker Recognition Game

In this paper, we aim to design an Interactive Speaker Recognition (ISR) module that identifies a speaker from a list of speakers only by requesting to utter a few user-specific words. To do so, we first formalize the ISR task as an interactive game involving the speaker and the ISR module. We then define the notation used to formally describe the game before detailing how we designed the ISR module.

2.1. Game Rules

To instantiate the ISR game, we first build a list of random individuals, or *guests*. Each guest is characterized by a few spoken sentences (enrolment phase), which act as their signature that we call *voice print*. In a second step, we label one of the guests as the target *speaker* that we aim to identify. Hence a game is defined by K guests characterized with K voice prints, and one of these guests is labeled as the speaker.

As the game starts, the K voice prints are provided to the ISR module, and it needs to identify the speaker among the guests. To do so, the ISR engine may interact with the speaker, but it can only request the speaker to utter T words within a predefined vocabulary list. At each turn of the game, the ISR module asks the speaker to say a word, the speaker pronounces it, and the ISR engine updates its internal speaker representation, as detailed in subsection 4.3, before asking the next word.

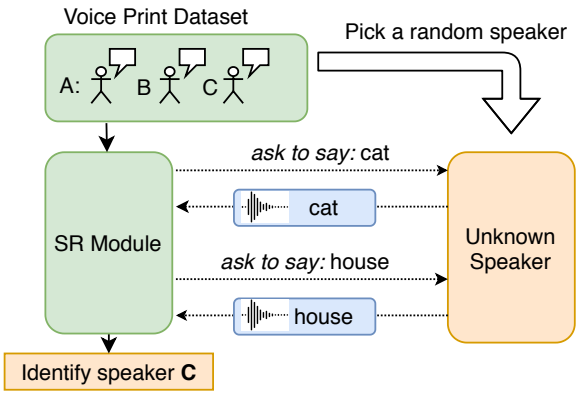


Figure 1: *Interactive Speaker Recognition game overview*

Again, the ISR module may only request T words. Thus, it needs to carefully choose them to correctly identify the speaker.

2.2. Game notation

A game is composed of a list of K guests characterized by their voice print $\boldsymbol{g} = [g^k]_{k=1}^K$ where $\boldsymbol{g}$ is a subset from a larger group of registered guests $\mathcal{G}$ of size G, and a predefined vocabulary $\mathcal{V}$ of size V. The ISR module aims at building a list of words $\boldsymbol{w} = [w_t]_{t=1}^T \in \mathcal{V}$ to be uttered by the speaker. The uttered version of $\boldsymbol{w}$ is $\boldsymbol{x} = \{x_t\}_{t=1}^T$, where x_t is the representation of word w_t pronounced by the speaker. Note that, for a given $\boldsymbol{w}$, $\boldsymbol{x}$ differs from one speaker to another.

2.3. Modelling the Speaker Recognition Module

From a machine learning perspective, we aim to design an ISR module that actively builds an internal speaker representation to perform voice print classification. As further discussed in subsection 4.2, this setting differs from standard SR methods that rely on generic but often long utterances [7]. In practice, we can split this task into two sub-modules: 1) an interactive module that queries the speaker to build the representation, and 2) a module that performs the voice print classification. In the following, we refer to these modules as *enquirer* and *guesser*.

Formally, the guesser must retrieve the speaker in a list of K guests characterized by their voice print $g^k \in \boldsymbol{g}$ and a sequence of words $\boldsymbol{x}$ uttered by the speaker $g^* \in \boldsymbol{g}$. Thus, the guesser has to link the speaker's uttered words to the speaker's voice print. The enquirer must select the next word $w_{t+1} \in \mathcal{V}$ that should be pronounced by the speaker given a list of K guests and the sequence of t previously spoken words $[x_{t'}]_{t'=1}^t$. Thus, the enquirer's goal is to pick the word that maximizes the guesser's success rate. Therefore, the ISR module first queries the speaker with the enquirer. Once the T words are collected, they are forwarded to the guesser to perform the speaker retrieval. In practice, this artificial split allows training the guesser with vanilla supervised learning, i.e., by randomly sampling words to retrieve speakers. The enquirer can hence be trained through reinforcement learning, as explained in the next section.

3. Speaker Recognition as a RL Problem

Reinforcement Learning addresses the problem of sequential decision making under uncertainty, where an agent interacts with the environment to maximize its cumulative reward [3]. In this paper, we aim at maximizing the guesser success ratio by allowing the enquirer to interact with the speaker, which makes RL a natural fit to solve the task. In this section, we thus provide the necessary RL terminology before relating the enquirer to the RL setting and defining the optimization protocol.

3.1. Markov Decision Process

In RL, the environment is modeled as a Markov Decision Process (MDP), where the MDP is formally defined as a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ [6, 8]. At each time step t, the agent is in a state $s_t \in \mathcal{S}$, where it selects an action $a_t \in \mathcal{A}$ according to its policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The agent then moves to the state s_{t+1} according to a transition kernel $\mathcal{P}$ and receives a reward $r_t = r(s_t, a_t)$ drawn from the environment's reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A}$. In this paper, we define the enquirer as a parametric policy $\pi_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta}$ is a vector of neural network weights that will be learnt with RL. At the beginning of an episode, the initial state corresponds to the list of guests: $s_0 = \{G\}$. At each time step t, the enquirer picks the action a_t by selecting the next word to utter w_t, where $w_t \sim \pi_{\boldsymbol{\theta}}(s_t)$. The speaker then pronounces the word w_t, which is processed to obtain x_t before being appended to the state $s_{t+1} = s_t \cup \{x_t\}$. After T words, the state $s_T = \{\boldsymbol{g}, \boldsymbol{x}\}$ is provided to the guesser. The enquirer is rewarded whenever the guesser identifies the speaker, i.e. $r(s_t, a_t) = 0$ if $t < T$ and $r(s_T, a_T) = \mathbb{1}_{[\text{argmax}_k p(g_k|s_T) = g^*]}$ where $\mathbb{1}$ is the indicator function.

3.2. Enquirer optimization Process

In RL, policy search aims at learning the policy $\pi_{\boldsymbol{\theta}^*}$ that maximizes the expected return by directly optimizing the policy parameters $\boldsymbol{\theta}$. More precisely, we search to maximize the mean value defined as $J(\boldsymbol{\theta}) = E_{\pi_{\boldsymbol{\theta}}}\big[\sum_{t=1}^T \gamma^{t-1} r(\boldsymbol{s}_t, a_t)\big]$. To do so, the policy parameters are updated in the direction of the gradient of $J(\boldsymbol{\theta})$. In practice, direct approximation of $\nabla J(\boldsymbol{\theta})$ may lead to destructively large policy updates, may converge to a poor deterministic policy at early training and it has a high variance. In this paper, we thus use the recent Proximal Policy Optimization approach (PPO) [9]. PPO clips the gradient estimate to have smooth policy updates, adds an entropy term to soften the policy distribution, and introduce a parametric baseline to reduce the gradient variance [9, 10].

4. Experimental Protocol

We first detail the data we used to create the ISR game before describing the speech processing phase. Finally, we present the neural training procedure.[1]

4.1. Dataset

We build the ISR game using the TIMIT corpus [11]. This dataset contains the voice recordings of 630 speakers with eight different accents. Each speaker uttered ten sentences, where two sentences are shared among all the speakers, and the eight others differ. Sentences are encoded as 16-bit, 16kHz waveforms. First, we define the ISR vocabulary by extracting the words of the two shared sentences, so the enquirer module may always request these words whatever the target speaker. In total, we obtained twenty distinct words such as *dark*, *year*, *carry* while dropping the uninformative specifier *a*. Second, we use the eight remaining sentences to build the speakers' voice print.

[1]The codebase and hyperparameters will be available at `https://github.com/Mathieu-Seurin/few_words`

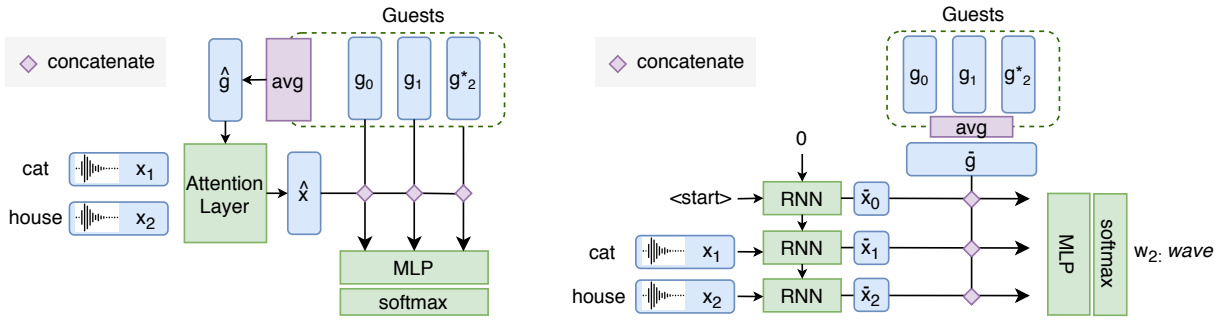


Figure 2: *(Left) The guesser must retrieve the speaker in the list of guests. (Right) The enquirer must select the next word to utter.*

4.2. Audio Processing

Following [7, 12, 13], we first down-sample the waveform to 8kHz before extracting the Mel Frequency Cepstral Coefficient (MFCC). We use MFCCs of dimension 20 with a frame-length of 25ms, mean-normalized over a sliding window of three seconds. We then process the MFCCs features through a pre-trained X-Vector network to obtain a high quality voice embedding of fixed dimension 128, where the X-Vector network is trained on augmented Switchboard [14], Mixer 6 [15], and NIST SREs [16][2]. To get the spoken word representation (word that the enquirer will query), we split the two shared sentences into individual words by following the TIMIT word timestamps. We then extract the X-Vector of each word w_t of every speaker k to obtain x_t^k. We compute the voice print by extracting the X-Vector of the eight remaining sentences before averaging them into a final vector of size 128 for each guest g^k.

4.3. Speaker Recognition Neural Modules

We here describe the ISR training details and illustrate the neural architectures in Figure 2.

Guesser. To model the guesser, we first model the guest by averaging the voice print into a single vector $\hat{g} = \frac{1}{K}\sum g^k$. We then pool the X-Vectors with an attention layer conditioned on $\hat{g}$ to get the guesser embedding $\hat{x}$ [18]:

$$e_t = MLP([x_t, \hat{g}]) \; ; \; \alpha = softmax(\boldsymbol{e}) \; ; \; \hat{x} = \sum_t \alpha_t x_t,$$

where [.,.] is the concatenation operator and MLP is a multi-layer perceptron with one hidden layer of size 256. We concatenate the guesser embedding with the guest voice print before projecting them through a MLP of size 512. Finally, we use a softmax to estimate the probability p^k of each guest to be the speaker, *i.e.* $p(g^k = g^*|\boldsymbol{x}, \boldsymbol{g}) = softmax\big(MLP([g^k, \hat{x}])\big)$. Both MLP have ReLU activations [19] with a dropout ratio of 0.5% [20]. The guesser is trained by minimizing the cross-entropy with ADAM [21], a batch size of 1024 and an initial learning rate of 3.10^{-4} over 45k games with five random guests.

Enquirer. To model the enquirer, we first represent the pseudo-sequence of words by feeding the X-Vectors into a bidirectional LSTM [22] to get the word hidden state $\bar{x}_t$ of dimension 2*128. Note that we use a start token for the first iteration. In parallel, we average the voice print into a single vector $\bar{g} = \frac{1}{K}\sum g^k$ to get the guest context. We then concatenate the word hidden state and the guest context before processing them through a one-hidden-layer MLP of size 256 with ReLU. Finally, a softmax activation estimates the probability of requesting the speaker to utter w_{t+1} as the next word: $p(w_{t+1}|x_t, \cdots, x_1, \boldsymbol{g}) = softmax\big(MLP([\bar{x}_t, \bar{g}])\big)$. The enquirer is trained by maximizing the reward encoded as the the guesser success ratio with PPO [9]. We use the ADAM optimizer [21] with a learning rate of 5e-3 and gradient clipping of 1 [23]. We performed 80k episodes of length $T = 3$ steps and $K = 5$ random guests. When applying PPO, we use an entropy coefficient of 0.01, a PPO clipping of 0.2, a discount factor of 0.9, an advantage coefficient of 0.95, and we apply four training batches of size 512 every 1024 transitions.

[2]available in kaldi library [17] at `http://www.kaldi-asr.org/models/m3`)

5. Experiments

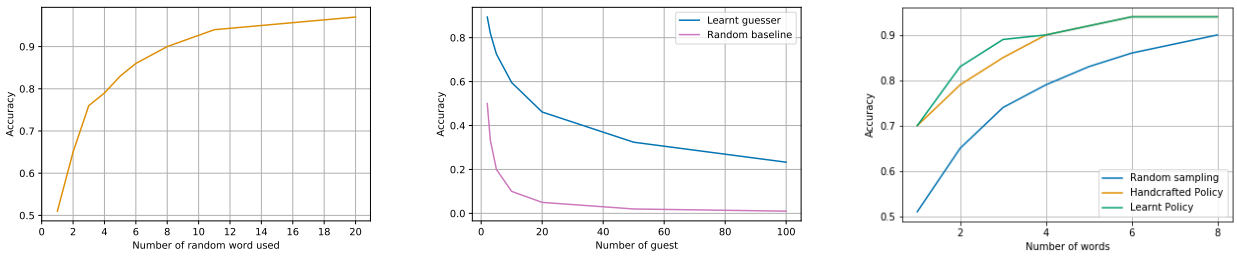
We run all experiments over five seeds, and report the mean and one-standard deviation when not specified otherwise.

5.1. Guesser Evaluation

In this section, we evaluate the guesser accuracy in different settings. As mentioned, we opt to request $T = 3$ words to identify the speaker among $K = 5$ guests. In this default setting, a random policy has a success ratio of 20%, whereas the neural model reaches $74.1\% \pm 0.2$ on the test set. As the guesser is trained on random words, these scores may be seen as an ISR lower-bound for the enquirer, which would later refine the word selection toward improving the guesser success ratio. Thus, this setting shows an excellent ratio between task difficulty and guesser initial success, allowing to train the enquirer with a relatively dense reward signal.

Word Sweep. We assess the guesser quality to successfully perform speaker recognition when increasing the number of words T in Figure 3a. We observe that a single word only gives 50% speaker retrieval, but the accuracy keeps improving when requesting more words. Noticeably, collecting the full vocabulary only scores up to 97% accuracy.

Guest Sweep. We report the impact of the number of guests K in Figure 3b. The guesser accuracy quickly collapses when increasing the number of guests with $K = 50$ having a 46% success ratio. As the number of words remains small, the guesser experiences increasing difficulty in discriminating the guests. One way to address this problem would be to use a Probabilistic Linear Discriminant Analysis (PLDA) [24] to enforce a discriminative space and explicitly separate the guests based on their class.



(a) (b) (c)

Figure 3: *(a-b) Guesser test accuracy, respectively varying the number of words (resp. guests) being used (c) Enquirer test accuracy varying the number of queried words. The RL enquirer outperforms the greedy baseline when selecting a low number of words.*

5.2. Enquirer Evaluation

Model. As previously mentioned, the enquirer aims to find the best sequence of words $\boldsymbol{w}$ that maximizes the guesser accuracy by interacting with the speaker. At each time step, we thus select the word with the highest probability $p(w_{t+1}|x_t, \cdots, x_1, \boldsymbol{g})$ according to the policy without replacement, i.e., the model never requests the same word twice.

Baseline. We compare our approach to two baselines: a non-deterministic random policy, and a deterministic greedy policy. As the name suggests, the random baseline picks T random words without replacement. On the other hand, the deterministic baseline always selects the same T words independently of the guests. To obtain a strong baseline, we preselect those words by taking advantage of the guesser model, where we value a sequence of words by computing the guesser accuracy over $\eta = 20000$ games. Optimally, we want to iterate over every tuple of words to retrieve the optimal set; yet, it is computationally intractable as it requires $\eta * \binom{V}{T}$ estimations. Therefore, we opt for a greedy policy that only requires $\eta * T * V$ guesser estimations. Given an empty tuple $\boldsymbol{\omega}_1$, we look for the word $w_t^* \in \mathcal{V}$, such as $\boldsymbol{\omega}_t \cup \{w_t^*\}$ returns the highest guesser accuracy. We repeat this operation T times to obtain the greedy deterministic policy characterized by $\boldsymbol{\omega}_T$. Note that the greedy policy may have some variance due to the guesser estimation, but it is empirically negligible.

Results. In our default setting, the random baseline reaches $74.1\% \pm 0.2$ speaker identification, and the greedy baseline scores up to 85.1%. The RL enquirer obtains up to $88.6\% \pm 0.5$, showing that it successfully leverages the guests' voice prints to refine its policy. We show the RL training in Figure 4. At early training, we observe that the ISR module still has high variance, and may behave randomly. However, RL enquirer steadily improves upon training, and it consistently outperforms the greedy baseline.

Word Diversity. To verify whether the enquirer adapts its policy to the guests, we generate a game for every speaker in the test set, and collect the requested words. We then compute the overlap Ω between the tuple of words by estimating the averaged Jaccard-index [25] of every pair of speakers as follow:

$$\Omega = \frac{1}{\sum_n^{N-1} n} \sum_{i=1}^{N-1} \sum_{j=i}^{N} J(\boldsymbol{w}^i, \boldsymbol{w}^j) \ ; \ \text{where J}(A, B) = \frac{A \cap B}{A \cup B}$$

where N is the number of speakers in the test set and $\boldsymbol{w}^i$ is the word tuple of game i. Intuitively, the lower this number, the more diverse the policy, e.g, the deterministic policy have a Jaccard-index of 1. In the default setting, the random policy has an index of 0.14 while the RL agent has an index of 0.65. Thus, the requested words are indeed diverse.

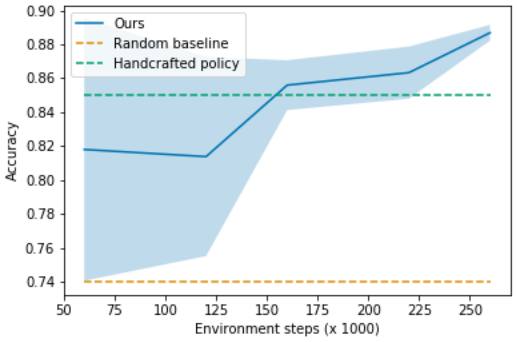


Figure 4: *Enquirer test accuracy averaged over 3 random seeds*

Requesting Additional Words We here study the impact of increasing the number of words T requested by the enquirer (see Figure 3c for results). First, we observe that the ISR module manages to outperform the greedy policy when requesting two to four words, showing that the interaction with the speaker is beneficial in the low data regime. This effect unsurprisingly diminishes when increasing the number of words. However, we noticed that the enquirer always outputs the same words when $t = 1$. It suggests that the model faces some difficulties contextualizing the guests' voice print before listening to the first speaker utterance. We assume that more advanced multimodal architecture, e.g., multimodal transformers [26, 27], may ease representation learning, further improving the ISR agent.

6. Conclusions and Future Directions

In this paper, we introduced the Interactive Speaker Recognition paradigm as an interactive game to improve speaker recognition accuracy while querying only a few words. We formalize it as a Markov Decision Process and train a neural model using Reinforcement Learning. We showed empirically that the ISR model successfully personalizes the words it requests to improve speaker identification, outperforming two non-interactive baselines. Our protocol may go beyond speaker recognition. The model can be adapted to select speech segments in the context of Text-To-Speech training. Interactive querying may also prevent malicious voice generator usage by asking complex words to the generator in a speaker verification setting.

7. References

[1] A. Irum and A. Salman, "Speaker verification using deep neural networks: A," *International Journal of Machine Learning and Computing*, vol. 9, no. 1, 2019.

[2] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] S. Chandramohan, M. Geist, and O. Pietquin, "Optimizing spoken dialogue management with fitted value iteration," in *Proc. of the Eleventh Annual Conference of the International Speech Communication Association (Interspeech)*, 2010.

[5] F. Strub, H. De Vries, J. Mary, B. Piot, A. Courvile, and O. Pietquin, "End-to-end optimization of goal-driven and visually grounded dialogue systems," in *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[6] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[7] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *Prof. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[8] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. of International Conference on Machine Learning (ICML)*, 2016.

[11] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, 11 1992.

[12] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *Proc. of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2017.

[13] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Prof. of IEEE Spoken Language Technology Workshop (SLT)*, 2016.

[14] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1992.

[15] E. Chodroff, M. Maciejewski, J. Trmal, S. Khudanpur, and J. Godfrey, "New release of mixer-6: Improved validity for phonetic study of speaker variation and identification," in *Proc. of International Conference on Language Resources and Evaluation (LREC)*, 2016.

[16] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds, "The nist speaker recognition evaluation–overview, methodology, systems, results, perspective," *Speech communication*, vol. 31, no. 2-3, pp. 225–254, 2000.

[17] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. of International Conference on Learning Representations (ICLR)*, 2014.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. of the International Conference on Machine Learning (ICML)*, 2010.

[20] 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.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. of International Conference on Machine Learning (ICML)*, 2013.

[24] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proc of. European Conference on Computer Vision (ECCV)*, 2006.

[25] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques regions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901.

[26] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[27] H. Tan and M. Bansal, "LXMERT: Learning cross-modality encoder representations from transformers," in *Proc. of Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.