Interspeech 2024
1-5 September 2024, Kos, Greece

Prompt Tuning for Audio Deepfake Detection: Computationally Efficient Test-time Domain Adaptation with Limited Target Dataset

Hideyuki Oiso[*1], *Yuto Matsunaga*[*2], *Kazuya Kakizaki*[2], *Taiki Miyagawa*[2]

[1]University of Tsukuba, Japan
[2]NEC Corporation, Japan

hideyuki@mdl.cs.tsukuba.ac.jp, yuto-matsunaga@nec.com, kazuya1210@nec.com, miyagawataik@nec.com

Abstract

We study test-time domain adaptation for audio deepfake detection (ADD), addressing three challenges: (i) source-target domain gaps, (ii) limited target dataset size, and (iii) high computational costs. We propose an ADD method using prompt tuning in a plug-in style. It bridges domain gaps by integrating it seamlessly with state-of-the-art transformer models and/or with other fine-tuning methods, boosting their performance on target data (challenge (i)). In addition, our method can fit small target datasets because it does not require a large number of extra parameters (challenge (ii)). This feature also contributes to computational efficiency, countering the high computational costs typically associated with large-scale pre-trained models in ADD (challenge (iii)). We conclude that prompt tuning for ADD under domain gaps presents a promising avenue for enhancing accuracy with minimal target data and negligible extra computational burden.

Index Terms: anti-spoofing, audio deepfake detection, domain adaptation, prompt tuning

1. Introduction

Audio deepfake is a collection of deep learning techniques that create artificial speech [1]. It can cause significant harm, including compromising the security of automatic speaker verification systems, contributing to spreading fake news, defaming an individual's reputation, and copyright violation. Thus, developing ADD models has attracted much attention, and the ASVspoof Challenges [2, 3, 4, 5, 6, 7] have made significant progress.

In addressing the major challenges in ADD, we identify three primary areas of focus: (i) domain gaps in source and target data, (ii) limitation of the target dataset size, and (iii) high computational costs.

The challenge (i) arises from the domain gaps between training (source) and test (target) datasets for ADD models [8, 9]. They come from discrepancies in deepfake generation methods [2], recording environments (e.g., devices and surroundings) [8], and languages [9], necessitating effective domain adaptation.

Specifically, we focus on *test-time domain adaptation with a labeled target dataset*. In this context, we adapt a pre-trained model from the source domain to the labeled test dataset from the target domain, without accessing the source domain dataset during the adaptation phase. This approach markedly differs from prior studies in ADD under domain gaps, which focus on *domain generalization*, where the target dataset is *unavailable* for training [10, 11, 12, 13, 14]. In contrast, our task, emphasizing test-time domain adaptation with accessible labeled target data, remains unexplored in the context of ADD, despite its importance in practical situations. For instance, one can collect target domain data even if they are small, and one can annotate a small portion of the collected data only with a small labeling cost, which can boost the performance of ADD models in the target domain.

Regarding the challenge (ii), labeled target datasets are often small and challenging to expand due to the limited availability of additional target data. For instance, target data on new deepfake methods is hard to collect. In such situations, domain adaptation via full fine-tuning (i.e., updating the entire network) is ineffective, as it tends to overfit the scarce target data [15].

Regarding the challenge (iii), state-of-the-art (SOTA) ADD models require high computational costs because they often use large-scale foundation models as their feature extractors, such as wav2vec 2.0 [16] and Whisper [17], to improve performance in test datasets [1, 18, 19, 20, 21]. This approach has been shown to be successful, but the size of foundation models is expected to be increasing in the future, leading to prohibitive computational costs for domain adaptation.

To address these prevalent issues, we propose a plug-in style ADD method for test-time domain adaptation with a small labeled target dataset. Our core idea is based on prompt tuning [22], which has been used in natural language processing (NLP) and image processing and can be seen as a type of fine-tuning methods [23, 24]; however, its effectiveness for ADD is underexplored. In prompt tuning, several trainable parameters (prompts) are inserted into the input feature (Fig. 1) and fine-tuned on a target dataset. Our method has the following characteristics: (I) Our method can be easily combined with any transformer models (i.e., plug-in style), including SOTA models, as well as other fine-tuning methods, such as the linear probing (i.e., updating the last linear layer) and full fine-tuning (Fig. 1 & Tab. 2). (II) Our method can avoid overfitting small target datasets because the number of additional trainable parameters is small; in fact, our method improves the equal error rate (EER) even when the target sample size is as small as 10 (Tab. 4), where fully fine-tuned models immediately overfit the target dataset (Tab. 4). (III) The additional computational cost of our method is minimal (Tab. 3). In our experiment, the additional trainable parameters account for only the order of $10^{-2}\%$ or $10^{-3}\%$ compared with the number of parameters in base architectures. Moreover, our ablation study reveals a rapid performance saturation concerning prompt length, indicating that a prompt length of ~ 10 is adequate.

In summary, our contribution is threefold:

1. To bridge domain gaps in ADD, we propose a plug-in-style method based on prompt tuning. Our method can be easily combined with any SOTA transformer models as well as other

* Equal contribution. This work was mainly completed during Oiso's internship at NEC Corporation.

10.21437/Interspeech.2024-81

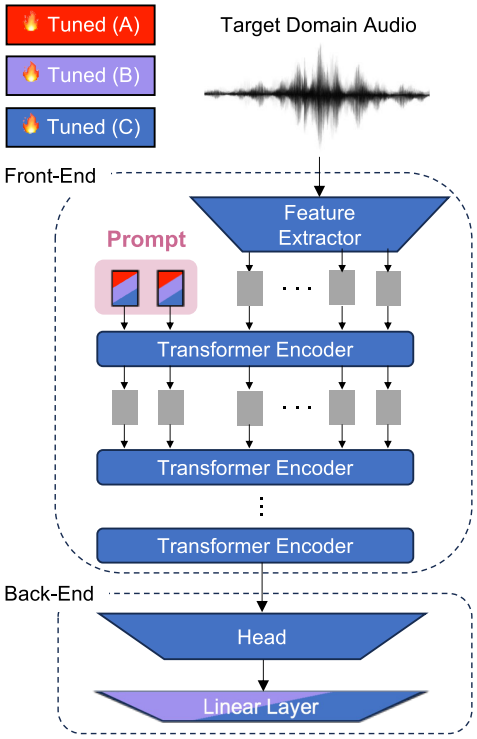

Figure 1: *Proposed prompt tuning for audio deepfake detection. (A), (B), and (C) mean tuning prompt, tuning prompt & the last linear layer, and tuning prompt & all layers, respectively.*

fine-tuning methods, reducing the EER of the target domain in many settings in our experiment (challenge (i)).

2. Our method counters overfitting to small target datasets by utilizing a limited number of additional trainable parameters, thus overcoming dataset size limitations (challenge (ii)).
3. Our method requires minimal extra computational resources, particularly when contrasted with full fine-tuning models, thereby addressing the issue of high computational costs due to base foundation models (challenge (iii)).

Code for reproduction is given at `https://github.com/Yuto-Matsunaga/Prompt_Tuning_for_Audio_Deepfake_Detection`.

2. Prompt tuning for audio deepfake detection

2.1. Problem definition: test-time domain adaptation with labeled target dataset for audio deepfake detection

We consider test-time domain adaptation with a labeled target dataset for ADD (binary classification). Let $\mathcal{X} \subseteq \mathbb{R}^{\Delta}$ be the audio waveform space, where $\Delta \geq 1$ is the number of sampling points. Let $\mathcal{Y} = \{Real, Fake\}$ be the binary label space. The pre-trained ADD model, $f : \mathcal{X} \to \mathcal{Y}$ (Front-End and Back-End in Fig. 1), parameterized by θ_f, determines whether a given waveform $x \in \mathcal{X}$ is classified as $Real$ or $Fake$. θ_f is pre-trained on a *source dataset* $D_S = \{(x_S^i, y_S^i)\}_{i=1}^{M_S}$ that is sampled from a *source distribution* $P_S(\mathcal{X}, \mathcal{Y})$. The *target dataset* is defined as $D_T = \{(x_T^j, y_T^j)\}_{j=1}^{M_T}$, sampled from a *target distribution* $P_T(\mathcal{X}, \mathcal{Y})$. We assume that there exists a discrepancy between $P_S(\mathcal{X}, \mathcal{Y})$ and $P_T(\mathcal{X}, \mathcal{Y})$. We aim to adapt the pre-trained model f to the target distribution $P_T(\mathcal{X}, \mathcal{Y})$, using post-hoc prompt tuning on D_T. This could involve introducing additional trainable parameters, or a prompt, θ_P, and tuning both θ_P and θ_f on D_T to minimize the empirical risk: $\frac{1}{M_T} \sum_{i=1}^{M_T} \mathcal{L}(f(x_T^i; \theta_f, \theta_P), y_T^i)$, where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is an arbitrary loss function. In our experiment, we use the class balanced loss [25], a class-imbalance-aware loss function. This is because ADD datasets are generally imbalanced, and we want to focus on performance gains by the prompt. Finally, it is important to note that we assume the source data D_S is *not* accessible at the adaptation stage due to privacy reasons, for example. Note also that, unlike typical test-time domain adaptation, D_T is labeled; i.e., $\{y_T^i\}_{i=1}^{M_T}$ as well as $\{x_T^i\}_{i=1}^{M_T}$ are available.

2.2. Architecture

The overall architecture of the ADD model used in our experiment is shown in Fig. 1. It is divided into two submodules: *Front-End* and *Back-End*, separated by dashed lines in Fig. 1. This Front-End-Back-End architecture is common in many SOTA models in ADD [1, 18, 19, 20, 21]. Front-End is a large-scale foundation model such as wav2vec 2.0 [16] and Whisper [17]. It consists of the first convolutional feature extractor and the following transformer encoders. The feature vector thus obtained is then input to Back-End. It consists of one non-linear module (denoted by Head) and the last linear layer and performs binary classification.

2.3. Proposed prompt tuning

The prompt θ_P is inserted to the intermediate feature vectors in Front-End (Fig. 1) during both the training phase in the target domain and the inference phase. Specifically, a prompt is $d \times N_P$ dimensional trainable parameters, where d is the dimension of a prompt, and N_P is the prompt length (the number of d-dimensional prompt vectors). The initialization algorithm of prompts is the same as [22].

We compare three types of prompt tuning, denoted by Tuned (A), (B), and (C) in Fig. 1. During the training phase in the target domain, type (A) tunes the prompt θ_P only. Type (B) tunes the prompt θ_P and the last linear layer. Type (C) tunes the prompt θ_P and all the parameters θ_f in f (i.e., Front-End and Back-End).

3. Experiments

3.1. Experimental setup

Datasets. We use ASVspoof 2019 LA [4] as the source dataset D_S for pre-training ADD models. ASVspoof 2019 LA is an audio deepfake dataset from English audio samples recorded in a hemi-anechoic chamber. For the target datasets D_T, we use In-The-Wild [8], Hamburg Adult Bilingual LAnguage (HABLA) [9], ASVspoof 2021 LA [6], and Voice Conversion Challenge (VCC) 2020 [26]. These four target datasets have multiple domain gaps, including deepfake generation methods (G), recording environments (E), and languages (L) (see Tab. 1).[1] Overall, we have seven different target domains denoted by $T_1, T_2, \ldots,$ and T_7 in Tab. 1. We perform experiments with $|D_T| = 10, 50, 100,$ and 1000 (except for VCC 2020, for which $|D_T| = 1000$ is not used); i.e., we randomly sample the target training datasets D_T from the original training datasets. Each

[1] For In-The-Wild, ASVspoof 2021 LA, and VCC 2020, which lack a predefined train-development-evaluation split, we implement random data splits, which are detailed in our code.

Table 1: *Statistics of target domain datasets and domain gaps from source dataset. #R and #F mean the numbers of real and fake samples in the training (Train), development (Dev), and evaluation (Eval) datasets. We perform experiments with* $|D_T| = 10, 50, 100,$ *and* 1000 *(except for VCC 2020, for which* $|D_T| = 1000$ *is not used); i.e., we randomly sample the target training datasets* D_T *from the original training datasets (Train). Each* D_T *maintains the original real-to-fake ratio. In the inference phase, all data in Dev or Eval are used. G, E, and L in the bottom row mean the domain gaps that come from the discrepancies in deepfake generation methods (G), recording environments (E), and languages (L), respectively. The source dataset* D_S *is ASVspoof 2019 LA.*

	In-The-Wild (T_1)	HABLA (T_2)	ASVspoof 2021 LA (T_3)	VCC 2020 English (T_4)	VCC 2020 Mandarin (T_5)	VCC 2020 German (T_6)	VCC 2020 Finnish (T_7)
	#R/#F	#R/#F	#R/#F	#R/#F	#R/#F	#R/#F	#R/#F
Train	7,935/4,776	11,041/27,750	1,676/14,788	40/250	18/194	21/189	23/189
Dev	3,970/2,386	2,718/6,980	1,960/14,966	18/127	10/96	9/96	9/97
Eval	8,058/4,654	9,057/23,270	14,816/133,360	42/249	22/190	20/190	18/194
Gaps	G, E	G, E, L	G	G, E	G, E, L	G, E, L	G, E, L

D_T maintains the original real-to-fake ratio.[2] As for the development and evaluation datasets, we use the original ones.

Pre-trained ADD models. We use two SOTA pre-trained ADD models [1, 18] for the source pre-trained models (both are pre-trained on D_S, i.e., ASVspoof 2019 LA). The first model [18], denoted by W2V hereafter, employs wav2vec 2.0 (300M parameters) [27] and Audio Anti-Spoofing using Integrated Spectro-Temporal graph attention networks (AASIST) (297k parameters) [28] as its Front-End and Back-End (Fig. 1), respectively. It is the SOTA model on ASVspoof 2021 LA, according to [7]. The second model [1], denoted by WSP, employs Whisper (39M parameters) [17] and mel-frequency cepstral coefficients (MFCC) [29] as its Front-End and MesoNet (28k parameters) [30] as its Back-End. It is the SOTA model on In-The-Wild, according to [1]. We use the official pre-trained parameters and pre-processes published by the original authors of these models.[3] Note that our prompt tuning can seamlessly integrate arbitrary transformer-based ADD models [19, 20, 21].

Hyperparameters. The hyperparameters are the prompt length N_P, learning rate η, weight decay λ, batch size B, and effective number β of the class balanced loss [25]. N_P is one of 1, 5, 10, or 100. We mainly use $N_P = 5$, and an ablation study is given in Tab. 5. The search spaces of η, λ, B, and β for W2V are $[10^{-6}, 10^{-4}]$ (log-uniform), $[5 \times 10^{-6}, 5 \times 10^{-4}]$ (log-uniform), $\{4, 8, 16\}$ (categorical), and $\{0.99, 0.999, 0.9999\}$ (categorical), respectively, and those for WSP are $[10^{-7}, 10^{-5}]$ (log-uniform), $[10^{-5}, 10^{-3}]$ (log-uniform), $\{4, 8, 16\}$ (categorical), and $\{0.99, 0.999, 0.9999\}$ (categorical), respectively. For hyperparameter tuning, the default algorithm of Optuna [31] is used. The number of training runs for hyperparameter tuning is 50 for all settings. During these runs, the EERs evaluated on the development datasets are monitored. The optimal hyperparameters are selected based on the best results from the 50 runs.

Computing infrastructure and runtimes. We use an NVIDIA Tesla V100 GPU throughout all experiments. The training durations for W2V and WSP models on the target datasets are approximately 500 and 100 seconds, respectively, under the conditions of a prompt length of 5, batch size of 16, dataset size of 50, and 100 epochs. For W2V, GPU memory consumption is observed at 8.3 GBs for (A) and (B), 4.1 GBs for (B) without prompt tuning, and 17 GBs for (C) and (C) without prompt tuning. In contrast, WSP consumes 2.3 GBs for (A) and (B), 1.7 GBs for (B) without prompt tuning, and 2.6 GBs for (C) and (C) without prompt tuning.

[2]The real-to-fake ratio for $|D_T| = 10$ is 1:1, not equal to the original ones, because the dataset size is too small and the ratio cannot be maintained.

[3]`https://github.com/TakHemlata/SSL_Anti-spoofing` (W2V) and `https://github.com/piotrkawa/deepfake-whisper-features` (WSP).

3.2. Results

We compare three fine-tuning methods illustrated in Fig. 1 ((A), (B), and (C)) with or without our prompt tuning. Tab. 2 presents the main result, which shows the EERs on the evaluation datasets ($N_P = 5$ and $|D_T| = 50$). Prompt tuning improves or maintains the EERs across many target domains despite its minimal additional trainable parameters (Tab. 3); for instance, they are as small as $\sim 10^{-3}\%$ of the number of parameters in the base pre-trained model (see the upper left of Tab. 3: (A) and W2V). This is an advantage over full fine-tuning ((C) with or without prompt tuning), which necessitates optimizing a huge number of parameters in the base pre-trained model. Consequently, options (A) and (B) emerge as viable alternatives under constrained computational budgets. Considering the recent trend towards increasing sizes of base pre-trained models, these alternative approaches could become increasingly relevant in the future.

Ablation study: target dataset sizes. Tab. 4 presents the ablation study results related to the size of the target dataset $|D_T|$. It is observed that prompt tuning reduces EERs in the majority of scenarios. This improvement is particularly notable when the target dataset size is limited to as small as 10 samples. Under this condition, (A) and (B) show superior performance compared to full fine-tuning (C) both with and without prompt tuning. This superiority is attributed to the tendency of full fine-tuning to overfit the small target dataset.

Ablation study: prompt lengths. Tab. 5 shows the ablation study on the prompt length N_P, a hyperparameter of our method. We advise limiting the number of prompts to a small scale, such as $N_P \sim O(1)$ or $N_P \sim O(10)$, because the performance saturates rapidly. This finding also shows that prompt tuning requires only minimal additional trainable parameters. Finally, note that the two ablation results above are obtained from the In-The-Wild dataset (T_1), but the same trends are also observed for the other datasets.

3.3. Discussion

This paper focuses on accuracy in the target dataset rather than the source dataset. The latter is less relevant when considering domain gaps due to changes in recording environments and

Table 2: *Equal Error Rates (EERs) [%] on various target domains (T_1, .., and T_7). Mean EERs over 12 runs with different random seeds are reported. The numbers in (...) are the standard deviations. $N_P = 5$ and $|D_T| = 50$ in this table. PT is short for prompt tuning. See Fig. 1 for the definitions of (A), (B), and (C). Bold numbers represent better results, comparing the methods with and without PT.*

Model	Method	T_1	T_2	T_3	T_4	T_5	T_6	T_7
W2V	(A) w/o PT	11.4	7.13	**0.84**	16.8	13.1	4.34	0.00
	(A)	**10.9**(0.0)	**6.47**(0.00)	2.53(0.00)	**7.19**(0.00)	**3.85**(0.00)	**0.789**(0.000)	0.00(0.000)
	(B) w/o PT	11.4(0.0)	6.64(0.02)	**0.823**(0.000)	16.8(0.0)	13.1(0.0)	4.34(0.00)	0.00(0.00)
	(B)	**10.5**(0.1)	**6.40**(0.01)	2.53(0.00)	**9.55**(0.03)	**3.85(0.00)**	**0.789**(0.00)	0.00(0.00)
	(C) w/o PT	5.48(0.49)	4.81(1.13)	**0.985**(0.021)	3.07(2.02)	**0.00**(0.00)	2.26(1.50)	0.00(0.00)
	(C)	**4.23**(0.47)	**2.55**(0.79)	1.24(0.11)	**1.08**(0.45)	0.158(0.141)	**0.632**(0.141)	0.00(0.00)
WSP	(A) w/o PT	28.7	20.2	11.7	0.00	0.00	0.00	0.00
	(A)	**28.6**(0.0)	**19.7**(0.0)	**11.3**(0.0)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
	(B) w/o PT	28.7(0.0)	20.2(0.0)	11.7(0.0)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
	(B)	**28.1**(0.1)	**19.6**(0.0)	**11.3**(0.0)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.00(0.00)
	(C) w/o PT	11.8(0.3)	17.1(0.5)	**8.46**(0.47)	0.00(0.00)	0.00(0.00)	00.00(0.00)	0.00(0.00)
	(C)	**11.4**(0.3)	**16.9**(0.5)	8.57(0.35)	0.00(0.00)	0.00(0.00)	00.00(0.00)	0.00(0.00)

Table 3: *Efficiency of our prompt-tuning methods. See Fig. 1 for the definitions of (A), (B), and (C). PT is short for prompt tuning. We use $N_P = 5$ in this table. "#Params" means the number of trainable parameters. "Ratio [%]" is defined as the number of trainable parameters divided by that of the base pre-trained model (W2V or WSP). The ratios of additional trainable parameters are minimal.*

Method	W2V		WSP	
	#Params	Ratio [%]	#Params	Ratio [%]
(A)	5,120	0.00161	1,920	0.0251
(B) w/o PT	322	0.000101	17	0.000274
(B)	5,442	0.00171	1,937	0.0253
(C) w/o PT	317,837,834	1.00	7,660,881	1.00
(C)	317,842,954	1.00161	7,662,801	1.000251

languages. Nevertheless, preserving source accuracy is crucial, particularly when the deepfake generation method in the target domain varies from that in the source domain; i.e., the adapted model should be robust against both the previous and new attack methods. To address this issue, we can easily integrate our plug-in-style method with existing techniques that maintain source accuracy while enhancing target accuracy [15, 32].

Our proposed method assumes that the base architecture includes transformer encoders. However, it could also be used for convolutional neural networks (CNNs) in light of recent studies of prompt tuning for CNNs [33]. Nevertheless, most of the SOTA models in ADD are based on transformers anyway [1, 18, 19, 20, 21]; thus, our method can be applied to a wide variety of SOTA models.

4. Conclusion

In our study on test-time domain adaptation for ADD, we tackle three prevalent challenges: (i) domain gaps between source and target dataset, (ii) limitation of target dataset size, and (iii) high computational costs. To overcome these issues, we introduce a method for ADD utilizing prompt tuning in a plug-in style. Our method can be applied to SOTA transformer models and other fine-tuning methods, to boost accuracy on target data. Additionally, our method efficiently improves accuracy even with limited amounts of labeled target data (e.g., 10). Furthermore, the computational cost of our method is low compared to full fine-tuning even when the base pre-trained model is huge. Our experiments show that the proposed method improves detection performance in most cases for two SOTA models and seven domain gaps.

Table 4: *Ablation study on training sample size $|D_T|$ (T_1, W2V, and $N_P = 5$). Mean EERs [%] over 12 runs with different random seeds are reported. The numbers in (...) are the standard deviations. See Fig. 1 for the definitions of (A), (B), and (C). PT is short for prompt tuning. Bold numbers represent better results, comparing the methods with and without PT.*

Method	Sample size		
	10	100	1000
(A) w/o PT	11.4	11.4	11.4
(A)	**10.9**(0.0)	**11.0**(0.0)	**10.2**(0.0)
(B) w/o PT	11.4(0.0)	11.3(0.0)	10.7(0.0)
(B)	**10.9**(0.0)	**10.5**(0.1)	**8.61**(0.04)
(C) w/o PT	11.4(0.8)	3.49(0.39)	**0.876**(0.158)
(C)	**11.1**(0.9)	**3.16**(0.55)	0.999(0.124)

Table 5: *Ablation study on prompt length N_P (T_1, W2V, $|D_T| = 50$). Mean EERs [%] over 12 runs with different random seeds are reported. The numbers in (...) represent the standard deviations. See Fig. 1 for the definitions of (A), (B), and (C). Note that one prompt corresponds to a 1024-dimensional vector.*

Method	Prompt length			
	1	5	10	100
(A)	12.1(0.0)	10.9(0.0)	10.0(0.0)	19.3(0.4)
(B)	12.1(0.0)	10.5(0.1)	10.2(0.0)	22.9(0.3)
(C)	5.17(0.61)	4.23(0.47)	4.68(0.40)	5.10(0.47)

5. References

[1] P. Kawa, M. Plata, M. Czuba, P. Szymański, and P. Syga, "Improved DeepFake Detection Using Whisper Features," in *Proc. INTERSPEECH 2023*, 2023, pp. 4009–4013.

[2] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Proc. Interspeech 2015*, 2015, pp. 2037–2041.

[3] H. Delgado, M. Todisco, M. Sahidullah, N. Evans, T. Kinnunen, K. A. Lee, and J. Yamagishi, "ASVspoof 2017 Version 2.0: meta-data analysis and baseline enhancements ," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2018)*, 2018, pp. 296–303.

[4] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. W. D. Evans, T. H. Kinnunen, and K.-A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," in *Interspeech*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:118646400

[5] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee *et al.*, "Asvspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *Computer Speech & Language*, vol. 64, p. 101114, 2020.

[6] J. Yamagishi, X. Wang, M. Todisco, M. Sahidullah, J. Patino, A. Nautsch, X. Liu, K. A. Lee, T. Kinnunen, N. Evans *et al.*, "Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection," *arXiv preprint arXiv:2109.00537*, 2021.

[7] X. Liu, X. Wang, M. Sahidullah, J. Patino, H. Delgado, T. Kinnunen, M. Todisco, J. Yamagishi, N. Evans, A. Nautsch *et al.*, "Asvspoof 2021: Towards spoofed and deepfake speech detection in the wild," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[8] N. Müller, P. Czempin, F. Diekmann, A. Froghyar, and K. Böttinger, "Does Audio Deepfake Detection Generalize?" in *Proc. Interspeech 2022*, 2022, pp. 2783–2787.

[9] P. A. Tamayo Flórez, R. Manrique, and B. Pereira Nunes, "HABLA: A Dataset of Latin American Spanish Accents for Voice Anti-spoofing," in *Proc. INTERSPEECH 2023*, 2023, pp. 1963–1967.

[10] H. jin Shim, J. weon Jung, and T. Kinnunen, "Multi-Dataset Co-Training with Sharpness-Aware Optimization for Audio Anti-spoofing," in *Proc. INTERSPEECH 2023*, 2023, pp. 3804–3808.

[11] S. Ding, Y. Zhang, and Z. Duan, "Samo: Speaker attractor multi-center one-class learning for voice anti-spoofing," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[12] Y. Wen, Z. Lei, Y. Yang, C. Liu, and M. Ma, "Multi-Path GMM-MobileNet Based on Attack Algorithms and Codecs for Synthetic Speech and Deepfake Detection," in *Proc. Interspeech 2022*, 2022, pp. 4795–4799.

[13] D. Salvi, P. Bestagini, and S. Tubaro, "Reliability estimation for synthetic speech detection," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[14] Y. Zang, Y. Zhang, and Z. Duan, "Phase perturbation improves channel robustness for speech spoofing countermeasures," in *Proc. INTERSPEECH 2023*, 2023, pp. 3162–3166.

[15] H. Ma, J. Yi, J. Tao, Y. Bai, Z. Tian, and C. Wang, "Continual Learning for Fake Audio Detection," in *Proc. Interspeech 2021*, 2021, pp. 886–890.

[16] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[17] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 492–28 518.

[18] H. Tak, M. Todisco, X. Wang, J. weon Jung, J. Yamagishi, and N. Evans, "Automatic Speaker Verification Spoofing and Deepfake Detection Using Wav2vec 2.0 and Data Augmentation," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, pp. 112–119.

[19] C. Wang, J. Yi, J. Tao, C. Y. Zhang, S. Zhang, and X. Chen, "Detection of Cross-Dataset Fake Audio Based on Prosodic and Pronunciation Features," in *Proc. INTERSPEECH 2023*, 2023, pp. 3844–3848.

[20] Y. Xie, H. Cheng, Y. Wang, and L. Ye, "Learning A Self-Supervised Domain-Invariant Feature Representation for Generalized Audio Deepfake Detection," in *Proc. INTERSPEECH 2023*, 2023, pp. 2808–2812.

[21] E. Rosello, A. Gomez-Alanis, A. M. Gomez, and A. Peinado, "A conformer-based classifier for variable-length utterance processing in anti-spoofing," in *Proc. INTERSPEECH 2023*, 2023, pp. 5281–5285.

[22] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3045–3059.

[23] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[24] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European Conference on Computer Vision*. Springer, 2022, pp. 709–727.

[25] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9268–9277.

[26] Y. Zhao, W.-C. Huang, X. Tian, J. Yamagishi, R. K. Das, T. Kinnunen, Z. Ling, and T. Toda, "Voice conversion challenge 2020: Intra-lingual semi-parallel and cross-lingual voice conversion," *arXiv preprint arXiv:2008.12527*, 2020.

[27] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino *et al.*, "Xls-r: Self-supervised cross-lingual speech representation learning at scale," *arXiv preprint arXiv:2111.09296*, 2021.

[28] J.-w. Jung, H.-S. Heo, H. Tak, H.-j. Shim, J. S. Chung, B.-J. Lee, H.-J. Yu, and N. Evans, "Aasist: Audio anti-spoofing using integrated spectro-temporal graph attention networks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6367–6371.

[29] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition," *Speech communication*, vol. 54, no. 4, pp. 543–565, 2012.

[30] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," in *2018 IEEE international workshop on information forensics and security (WIFS)*. IEEE, 2018, pp. 1–7.

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[32] X. Zhang, J. Yi, J. Tao, C. Wang, and C. Y. Zhang, "Do you remember? overcoming catastrophic forgetting for fake audio detection," in *International Conference on Machine Learning*. PMLR, 2023, pp. 41 819–41 831.

[33] H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola, "Exploring visual prompts for adapting large-scale models," *arXiv preprint arXiv:2203.17274*, 2022.