



BPCNN: Bi-Point Input for Convolutional Neural Networks in Speaker Spoofing Detection

Sunghyun Yoon¹ and Ha-Jin Yu^{2,*}

- ¹ Department of Artificial Intelligence, Kongju National University, Cheonan 31080, Korea; syoon@kongju.ac.kr
- ² School of Computer Science, University of Seoul, Seoul 02504, Korea
- * Correspondence: hjyu@uos.ac.kr

Abstract: We propose a method, called bi-point input, for convolutional neural networks (CNNs) that handle variable-length input features (e.g., speech utterances). Feeding input features into a CNN in a mini-batch unit requires that all features in each mini-batch have the same shape. A set of variable-length features cannot be directly fed into a CNN because they commonly have different lengths. Feature segmentation is a dominant method for CNNs to handle variable-length features, where each feature is decomposed into fixed-length segments. A CNN receives one segment as an input at one time. However, a CNN can consider only the information of one segment at one time, not the entire feature. This drawback limits the amount of information available at one time and consequently results in suboptimal solutions. Our proposed method alleviates this problem by increasing the amount of information available at one time. With the proposed method, a CNN receives a pair of two segments obtained from a feature as an input at one time. Each of the two segments generally covers different time ranges and therefore has different information. We also propose various combination methods and provide a rough guidance to set a proper segment length without evaluation. We evaluate the proposed method on the spoofing detection tasks using the ASVspoof 2019 database under various conditions. The experimental results reveal that the proposed method reduces the relative equal error rate (EER) by approximately 17.2% and 43.8% on average for the logical access (LA) and physical access (PA) tasks, respectively.

Keywords: bidirectional feature segmentation; bi-point input; convolutional neural network (CNN); spoofing detection; variable-length features

1. Introduction

Automatic speaker verification (ASV) is a technique to verify a user's identity from his or her utterances. It is convenient and has provided consistently low verification errors. However, ASV is vulnerable to various spoofing attacks that attempt to deceive the system and manipulate its results using a fake utterance. A spoofed utterance is one artificially manipulated to sound like the target speaker's utterance. Current spoofing attacks include text-to-speech (TTS) synthesis, voice conversion (VC), and replay attacks. Although the identity claim with spoofed utterance must be rejected for obviating the harms caused by the impersonation, ASV may not be able to reject these claims because it is hard to distinguish between a human being's genuine utterance and the spoofed utterance.

The development of spoofing detection systems to address the shortcomings of ASV has recently gained importance. Spoofing detection is a technique to protect ASV systems from spoofing attacks by distinguishing between genuine and spoofed utterances. Spoofing detection systems reject the identity claim with a spoofed utterance, regardless of how similar the spoofed utterance is to the target speaker's genuine utterance. For research and development on spoofing detection techniques, ASV spoofing and countermeasure (ASVspoof) challenges have been organized periodically [1–4]. In this paper, we focus on



Citation: Yoon, S.; Yu, H.-J. BPCNN: Bi-Point Input for Convolutional Neural Networks in Speaker Spoofing Detection. *Sensors* 2022, 22, 4483. https://doi.org/10.3390/ s22124483

Academic Editor: Julio Carabias-Orti

Received: 12 May 2022 Accepted: 7 June 2022 Published: 14 June 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/). ASVspoof 2019 [3], which first addresses all major types of spoofing attacks (i.e., TTS, VC, and replay attack).

It is essential for spoofing detection to capture the differences of the frequency attributes between genuine and spoofed utterances, regardless of the type of spoofing attack. The frequency attributes of genuine and spoofed utterances differ [5,6]. Furthermore, the discriminative attributes between genuine and spoofed utterances are spread across the entire time domain of an utterance and there is no strong time dependency. This locally connected information can be effectively modeled by convolutional neural networks (CNNs) [7]. Accordingly, many cutting-edge spoofing detection systems are based on CNNs [8–11].

Like other types of neural networks, a CNN is also trained in a mini-batch unit [12]. When feeding input features into a CNN in a mini-batch unit, all features in each mini-batch must have the same size. However, the lengths of speech utterances differ because speakers have different uttering styles, even if all speakers are constrained to utter only the same phrase. Therefore, it is required that all input utterances (it is not limited to 'utterance' and can correspond to any kind of sequence (e.g., a waveform signal, a sequence of acoustic feature vectors), although we use the word 'utterance' in this sentence) have the same length before we feed them into a CNN. There are several ways to make all input features in each mini-batch have the same length, which can be categorized into three methods: (1) padding only, (2) both padding and truncation, and (3) feature segmentation. Throughout this paper, the *target length* refers to the length that all features in a mini-batch must have.

The first method is to apply padding only, by repeating the feature's own frames or filling with zeros (i.e., zero-padding). In other words, there is no feature truncated. The target length is set long enough (e.g., the length of the longest feature over the dataset) so that no feature is longer than the target length, as in [13–15]. This method has the advantage of no information loss caused by truncation. However, it causes a data redundancy problem caused by padding because the padded frames are not novel but merely part of the original frames (for the repeating) or common zero values that are meaningless (for the zero-padding). Thus, padding brings the additional unnecessary information in terms of the information diversity and novelty, which may not help improve classification. Furthermore, this problem accompanies the increases in the computational burden and memory usage. This side effect worsens as the standard deviation of the utterance lengths increases.

The second method is to apply both padding and truncation. The target length is set to a moderate length, neither too long nor too short (e.g., the average length), as in [9,10,15–24]. Features shorter than the target length are padded and those longer than the target length are truncated. This method has a relatively smaller computational burden and memory usage than the first method. However, it causes the information distortion that is due to both padding and truncation. The padding leads to the additional unnecessary information, as described above, and truncation leads to the information loss because of not being able to use truncated frames.

In [25], an on-the-fly data loader was proposed to handle variable-length input features. It can be regarded as an extension of the aforementioned second method to enable batch-wise variable-length training. For each iteration, the target length (common to all the features in each mini-batch) and start points (for truncation only; different for each input feature in a mini-batch) are randomly selected. This method achieved improved performances on both text-independent speaker verification and language identification [25]. However, our findings reveal that the on-the-fly data loader was shown to be ineffective for improving performance on spoofing detection, except in some cases. This issue will be discussed in Section 5.2.

The third method is feature segmentation [8,11,26–29], which breaks each feature down into fixed-length segments. The target length corresponds to the length of a segment. A CNN receives one segment at one time rather than an entire feature. This method seems to combine the benefits of the previous two methods. It can reduce the information distortion

caused by padding and truncation. Therefore, it is a more flexible method in terms of the difference in the utterance length. In this paper, we used the feature segmentation as a baseline method.

However, segmentation has the shortcoming of the limitation on the amount of information available at one time because a CNN can consider only one segment at one time, not an entire feature. At least, but not limited in spoofing detection, this shortcoming is detrimental to the performance. It is important for better detection to capture and model as many the discriminative attributes in an input feature as possible, and then aggregate them into a fixed-size embedding. Moreover, the discriminative attributes are spread across the entire time domain, rather than being concentrated in certain time intervals, as mentioned earlier. In other words, the feature segmentation is inevitable to reach suboptimal performances on spoofing detection. This problem has made us take how to increase the amount of information available at one time into consideration.

We had proposed a method called multiple points input for CNN [27]. The proposed method increases the amount of information available at one time by feeding a pair of two segments as an input to a CNN. It exhibited substantial reductions in detection error in the ASVspoof 2019 PA scenario (i.e., replay attack detection task). Nonetheless, there are still some issues worthy of further discussions in [27] because our previous experiments in [27] were performed under restricted conditions: (1) the use of only one type of CNN-based model, squeeze-and-excitation [30] residual network [31] (SE-ResNet), (2) a fixed target length at 400 frames, and (3) evaluation only in the replay attack detection task. In this paper, therefore, we present comprehensive experimental results conducted under various conditions and in-depth analyses to verify the ability of the proposed method to reduce the detection errors in various conditions. The contributions of this paper are summarized as follows:

- Proposing a bi-point input for CNN (BPCNN). It feeds a pair of two segments, rather than one segment, into a CNN at one time. The main advantage of BPCNN is to increase the amount of the available information at one time with hardly changing the CNN structure.
- Proposing various methods for combining the two segments in various levels: embeddinglevel combination, feature map-level combination, and two-channel input.
- Evaluating the performances of the proposed method in both logical and physical access tasks in various conditions and analyzing the effect and strength of the proposed method via the ablation studies.

The structure of this paper is organized as follows. Section 2 outlines the conventional feature segmentation method. Section 3 introduces the proposed method. Section 4 describes the experimental setup, and Section 5 analyzes the obtained results. Finally, Section 6 concludes the paper.

2. Conventional Feature Segmentation

Feature segmentation is a method that decomposes all the variable-length features into fixed-length segments along the time axis. Figure 1 and Algorithm 1 illustrate the overview and pseudocode of conventional feature segmentation. Let $\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}]$ be a sequence of *T* feature frames extracted from an utterance (e.g., log power spectrogram). With the feature segmentation, \mathbf{X} is divided into *N* segments $\{\mathbf{F}_i\}_{i=0}^{N-1}$ using a sliding window of M(< T) frames length with *L* frames shift (this explanation is only for the case of M < T. The case of $M \ge T$ will be discussed later.) All segments have the same length of *M*, which is the target length. The number of segments *N* is different for each feature, which is proportional to the feature length *T*. The *i*-th segment \mathbf{F}_i corresponds to the sequence of *M* frames that start from the *iL*-th frame \mathbf{x}_{iL} to the (iL + M - 1)-th frame \mathbf{x}_{iL+M-1} , $\mathbf{F}_i = [\mathbf{x}_{iL}, \dots, \mathbf{x}_{iL+M-1}]$, where $iL + M \le T$ for all *i*.



Figure 1. The overview of the conventional feature segmentation when (left) M < T and (right) M > T, where $\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}]$ is a feature with the length of T, M is the segment length, L is the shift interval for segmentation, and \mathbf{F}_i is the *i*-th segment.

Algorithm 1 Pseudocode of Conventional Feature Segmentation	on
Input	
$\mathbf{X} = [\mathbf{x}_0, \ldots, \mathbf{x}_{T-1}]$: A feature with the length of <i>T</i>	
M: The segment length	
L: The shift interval for segmentation	
Output	
S: A list of the segments	
1. if $T > M$:	
2. $S = []$	# Empty list
3. $N = \left \frac{M}{T} \right $	# The quotient
4. $r = \operatorname{mod}(T - M, L)$	# The remainder
5. for $i = 0$ to $N - 1$:	
6. $\mathbf{F}_i = [\mathbf{x}_{iL}, \ldots, \mathbf{x}_{iL+M-1}]$	# The <i>i</i> -th segment
7. S.append(\mathbf{F}_i)	
8. if $r > 0$:	
9. $\mathbf{F}_N = [\mathbf{x}_{T-M}, \ldots, \mathbf{x}_{T-1}]$	# The last M frames of X
10. S.append(\mathbf{F}_N)	
11. return S	
12. else if $T < M$:	
13. $\mathbf{F}_0 = \operatorname{pad}(\mathbf{X}, M)$	# Set \mathbf{X} to have the length of M
14. return $\{\mathbf{F}_0\}$	
15. else :	# The case of $T == M$
16. $\mathbf{F}_0 = \mathbf{X}$	# X itself becomes a segment
17. return $\{\mathbf{F}_0\}$	

We do not consider a unified feature map approach [11,16,22] for segmentation. In the unified feature map approach, all features are first extended to have the same length that are sufficiently long (e.g., usually the length of the longest feature over the dataset) before segmentation. Therefore, all features have the same number of segments regardless of their original lengths. However, we empirically found no significant difference in detection error between with and without the unified feature map. Nonetheless, our approach (i.e., without the unified feature map) is more efficient in terms of data redundancy and computational amount and memory usage. This is why the unified feature map approach is not considered throughout this paper.

The following two issues should be considered to implement the feature segmentation. The first issue is how to handle the features shorter than or equal to the target length (i.e., $M \ge T$), where segmentation cannot be applied. As described above, M should be set to a moderate length, neither too long nor too short. In most cases, therefore, there are some features shorter than M. In our approach, we extend the features shorter than M to have the length of M by padding M - T frames from those features. If the number

of additionally required frames M - T is below the target length T (i.e., M - T < T), a padded segment \mathbf{F}_0 is obtained by padding the first M - T frames of \mathbf{X} at the end of \mathbf{X} , $\mathbf{F}_0 = \text{pad}(\mathbf{X}, M) = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}, \mathbf{x}_0, \dots, \mathbf{x}_{M-T-1}] = [\mathbf{X}, \mathbf{x}_0, \dots, \mathbf{x}_{M-T-1}]$. Otherwise (i.e., $M - T \ge T$), \mathbf{F}_0 is defined as $Q = \lfloor \frac{M}{T} \rfloor$ times repeat of \mathbf{X} followed by the first M - QT frames of \mathbf{X} , $\mathbf{F}_0 = \text{pad}(\mathbf{X}, M) = [\mathbf{X}, \dots, \mathbf{X}, \mathbf{x}_0, \dots, \mathbf{x}_{M-QT-1}]$. When M = T, \mathbf{F}_0 just corresponds to \mathbf{X} itself. In turn, there is only one segment \mathbf{F}_0 for the feature whose length is shorter than or equal to M.

The second issue is how to handle the remaining frames fewer than M after segmentation. For each feature of which the length is T(>M), there are R = mod(T - M, L) frames left after segmentation, where mod(a, b) is the modulo operation with the dividend a and divisor b. We also use the remaining R frames to exploit complete frames. When R > 0, take an additional segment corresponding to the last M frames of X, regardless of the number of remainder frames R. This segment is defined as the last segment $\mathbf{F}_N = [\mathbf{x}_{T-M}, \dots, \mathbf{x}_{T-1}]$.

In the training phase of the proposed method, each segment is treated as individual data, regardless of which feature the segment was obtained from. Even so, all segments obtained from a feature share the same class label. For instance, when the class label of a feature **X** is *c*, all the segments $\{\mathbf{F}_i\}_{i=0}^{N-1}$ from **X** also have the class label *c*. In the evaluation phase, the score (e.g., log probability of the genuine class) of each feature is calculated by aggregating (e.g., averaging) the scores of all the segments obtained from the feature.

3. The Proposed Method

The proposed method alleviates the problem of the conventional method that the available information at one time is limited because CNN takes only one segment F_i at one time. The proposed method consists of two steps: (1) bidirectional feature segmentation (cf., unidirectional feature segmentation in the conventional method) and (2) bi-point input (cf., one-point input in the conventional method).

3.1. Bidirectional Feature Segmentation

In the conventional feature segmentation described in Section 2, a feature is decomposed along only one direction: the positive time direction. In contrast, in the proposed method, a feature is decomposed along two opposite directions: the positive and negative time directions. It is inspired by the bidirectional recurrent neural network (BRNN) [32]. BRNNs are developed to increase the amount of available information. They can obtain twice as much information as standard RNNs, using the information in both the forward (responsible for the positive time direction) and backward (responsible for the negative time direction) sequences.

Figure 2 and Algorithm 2 illustrate the overview and pseudocode of bidirectional feature segmentation. Given a feature $\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}]$, bidirectional feature segmentation is processed in two steps: forward and backward steps. The forward step is identical to conventional feature segmentation. In the forward step, \mathbf{X} is divided by moving the sliding window, of which the length and shift interval are $M(\leq T)$ and L, respectively, along the positive time direction (i.e., from 0 to T - 1) to obtain N forward segments $\{\mathbf{F}_i\}_{i=0}^{N-1}$. The *i*-th forward segment \mathbf{F}_i is $[\mathbf{x}_{iL}, \dots, \mathbf{x}_{iL+M-1}]$, as explained in Section 2. In contrast, in the backward step, \mathbf{X} is divided by moving the sliding window along the negative time direction (i.e., from T - 1 to 0). After the backward step is complete, we can obtain N backward segments $\{\mathbf{B}_i\}_{i=0}^{N-1}$. Let $\tilde{\mathbf{X}} = [\mathbf{x}_{T-1}, \dots, \mathbf{x}_0]$ be the flipped order of \mathbf{X} . The *i*-th backward segment \mathbf{B}_i is $[\mathbf{x}_{T-iL-1}, \dots, \mathbf{x}_{T-iL-M}]$, where $iL + M \leq T$ for all *i*.



Figure 2. The overview of the proposed bidirectional feature segmentation when (**left**) M < T and (**right**) M > T, where $\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}]$ is a feature with the length of *T*, *M* is the segment length, and \mathbf{F}_i and \mathbf{B}_i are the *i* -th forward and backward segments, respectively.

Algorithm 2 Pseudocode of Bidirectional Feature Segmentation	
Input	
$\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}]$: A feature with the length of T	
$\tilde{\mathbf{X}} = [\mathbf{x}_{T-1}, \dots, \mathbf{x}_0]$: The flipped order of \mathbf{X}	
M: The segment length	
L: The shift interval for segmentation	
Output	
S: A list of the segment pairs	
1. if $T > M$:	
$S = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$	# Empty list
3. $N = \left\lfloor \frac{M}{T} \right\rfloor$	# The quotient
4. $r = \operatorname{mod}(T - M, L) $ #1	Гhe remainder
5. for $i = 0$ to $N - 1$:	
6. $\mathbf{F}_i = [\mathbf{x}_{iL}, \dots, \mathbf{x}_{iL+M-1}] \qquad \qquad \text{# The } i\text{-th for}$	ward segment
7. $\mathbf{B}_{i} = [\mathbf{x}_{T-iL-1}, \dots, \mathbf{x}_{T-iL-M}] $ # The <i>i</i> -th back	ward segment
8. S.append($\mathbf{F}_i, \mathbf{B}_i$)	
9. If $r > 0$: 10 E ₁ = $[r = 1, r = 1]$ #The last	M frames of V
10. $\mathbf{r}_N = [\mathbf{x}_{T-M}, \dots, \mathbf{x}_{T-1}]$ # The last	Winames of X
11. $\mathbf{D}_N = [\mathbf{x}_{M-1}, \dots, \mathbf{x}_0] \qquad \text{# The last}$	M frames of X
12. S.append($\mathbf{F}_i, \mathbf{B}_i$)	
13. Introduction 13. $14 \text{ old} \mathbf{i} f T < M$	
15. $\mathbf{F}_0 = \operatorname{pad}(\mathbf{X}, M)$ #Set X to have t	he length of M
16 P $\left(\tilde{\mathbf{x}}_{M}\right)$ #Set $\tilde{\mathbf{y}}$ to have t	he length of M
$\mathbf{b}_0 = \operatorname{pad}(\mathbf{x}, M) \qquad \qquad \text{# Set } \mathbf{x} \text{ to have the set } \mathbf{x} $	
17. return $\{\mathbf{F}_i, \mathbf{B}_i\}$	
18. else : # The ca	se of $T == M$
19. $\mathbf{F}_0 = \mathbf{X}$ # X itself become	mes a forward
U.N.C. 161	segment
20. $\mathbf{B}_0 = \tilde{\mathbf{X}}$ # X itself becom	es a backward
21. return $\{\mathbf{F}_i, \mathbf{B}_i\}$	Segment

When M > T, one forward segment \mathbf{F}_0 and one backward segment \mathbf{B}_0 can be obtained without segmentation, as in the conventional method. The method of obtaining the forward segment \mathbf{F}_0 is identical to the conventional method. The backward segment \mathbf{B}_0 can be obtained as follows. If M - T < T, where M - T corresponds to the number of additionally required frames, \mathbf{B}_0 is obtained by padding the first M - T frames of $\tilde{\mathbf{X}}$ at the end of $\tilde{\mathbf{X}}$, $\mathbf{B}_0 = \text{pad}\left(\tilde{\mathbf{X}}, M\right) = [\mathbf{x}_{T-1}, \dots, \mathbf{x}_0, \mathbf{x}_{T-1}, \dots, \mathbf{x}_{2T-M}]$ (the first M - T frames of $\tilde{\mathbf{X}}$ correspond to the last M - T frames of \mathbf{X} with the flipped order of frames, $[\mathbf{x}_{T-1}, \dots, \mathbf{x}_{2T-M}]$, where 2T - M = T - (M - T)) Otherwise (i.e., $M - T \ge T$), \mathbf{B}_0 is defined as $Q = \begin{bmatrix} M \\ T \end{bmatrix}$ times repeat of $\tilde{\mathbf{X}}$ followed by the first M - QT frames of $\tilde{\mathbf{X}}$, $\mathbf{B}_0 = \text{pad}\left(\tilde{\mathbf{X}}, M\right) = \begin{bmatrix} \tilde{\mathbf{X}}, \dots, \tilde{\mathbf{X}}, \mathbf{x}_{T-1}, \dots, \mathbf{x}_{(Q+1)T-M} \end{bmatrix}$ (the first M - QT frames of $\tilde{\mathbf{X}}$ correspond to the last M - QT frames of \mathbf{X} with the flipped order of frames, $\begin{bmatrix} \mathbf{x}_{T-1}, \dots, \mathbf{x}_{(Q+1)T-M} \end{bmatrix}$, where (Q+1)T - M = T - (M - QT)) For the case of M = T,

B₀ just corresponds to **X** itself.

If there are remaining *R* frames after bidirectional feature segmentation, the last forward segment \mathbf{F}_N is defined the same as in conventional feature segmentation. Similarly,

the last backward segment \mathbf{B}_N is defined as the last M frames of $\mathbf{X}, \mathbf{B}_N = [\mathbf{x}_{M-1}, \dots, \mathbf{x}_0]$.

All the other conditions not described are the same as those in conventional feature segmentation. For example, the class label of the feature is also shared for all backward segments.

3.2. Bi-Point Input

With bidirectional feature segmentation, we can obtain two sets of segments from an utterance: a set of forward segments $\{\mathbf{F}_i\}_{i=0}^{N-1}$ and a set of backward segments $\{\mathbf{B}_i\}_{i=0}^{N-1}$. With bi-point input, the *i*-th forward segment \mathbf{F}_i forms a pair with the *i*-th backward segment \mathbf{B}_i . Therefore, there are *N* pairs of segments. The CNN receives one pair (i.e., two segments, \mathbf{F}_i and \mathbf{B}_i) as an input at one time, rather than only one (forward) segment. Therefore, the amount of available information at one time becomes up to double. The amount of available information with the bi-point input is usually less than twice the conventional one-point input method because the time ranges of \mathbf{F}_i and \mathbf{B}_i could be partially or completely overlapped for some *i* (e.g., around the middle of the feature) under our approach. The wider the overlapped range, the less the gain in the increase in the amount of information available at one time.

It is reasonable that each pair consists of \mathbf{F}_i and \mathbf{B}_i , not \mathbf{F}_i and the (N-1-i)-th backward segment \mathbf{B}_{N-1-i} . Recall that the primary purpose of the bi-point input is to increase the amount of information available at one time. If the *i*-th pair is made up of $\mathbf{F}_i = [\mathbf{x}_{iL}, \dots, \mathbf{x}_{iL+M-1}]$ and $\mathbf{B}_{N-1-i} = [\mathbf{x}_{iL+M-1}, \dots, \mathbf{x}_{iL}]$ (for a brief explanation, we assume there is no remainder frame after segmentation, that is, R = mod(T - M, L) = 0, but even if R > 0, the increased amount of information available at one time is negligible and has little effect on reducing the detection error), both segments in each pair (i.e., \mathbf{F}_i and \mathbf{B}_{N-1-i}) always cover the exact same time range. The only difference between the two segments is the frame order. It indicates that both segments in each pair always have the same information. In this case, therefore, it is difficult to increase the amount of information available at one time. Even if the detection error is reduced, it may be due to the data augmentation based on flipping the time order of feature, rather than the increase in the available information at one time. We will discuss this issue in detail in Section 5.2.

Recall that \mathbf{F}_i and \mathbf{B}_i have the opposite frame order (explained in Section 3.1), which not only avoids meaningless operations but also has the side effect of data augmentation. Although \mathbf{F}_i and \mathbf{B}_i in a same pair generally cover the different time ranges, there could be some cases where \mathbf{F}_i and \mathbf{B}_i cover the same time range at certain *i* (e.g., around the middle of the feature), as mentioned above. When \mathbf{F}_i and \mathbf{B}_i cover the same time range, their outputs are exactly the same if they have the same frame order. In this case, feeding \mathbf{F}_i and \mathbf{B}_i together is merely a redundancy of meaningless operations, which is not the increase in the available information at one time nor data augmentation. This problem can be avoided by making the frame orders of \mathbf{F}_i and \mathbf{B}_i opposite, although the available information at one time still cannot be increased. As long as the frame orders are opposite, \mathbf{F}_i and \mathbf{B}_i can be treated as different features for CNN, even if they cover exactly the same time range. Notice that the words 'different features' do not mean that \mathbf{F}_i and \mathbf{B}_i have different information, but mean that they just have (slightly) different output values. Figure 3 illustrates the overall frameworks of (a) the conventional system (i.e., based on conventional feature segmentation and one-point input) and (b–d) the proposed systems (i.e., based on bidirectional feature segmentation and bi-point input). The differences between the conventional and proposed systems are highlighted in blue shade. In the proposed systems, both segments in a pair are fed into the same CNN, which indicates that the CNN parameters are shared across both segments. The results from each of the segments in a pair are combined before being fed into the classifier. From a pair of two segments, therefore, we obtain one score value, not two.



Figure 3. The framework of (**a**) conventional and (**b**–**d**) proposed systems, where GAP stands for global average pooling, Combine (**b**,**c**) stands for one of the appropriate combination methods (described in Sections 3.2.1 and 3.2.2, respectively), and \mathbf{F}_i and \mathbf{B}_i are the *i* -th forward and backward segments, respectively. The proposed systems consist of (**b**) embedding-level combination, (**c**) feature-map level combination, and (**d**) two-channel input. The modules for the proposed method are marked in blue.

We propose several methods to combine the results from the two segments in a pair, including the methods used in [8,27,29]. These are designed considering that most CNN-based models are composed in the following order of modules: CNN, a global average pooling (GAP) layer, and a classifier. The GAP layer transforms *C* feature maps of size ($C \times H \times W$) into a vector of size *C* by averaging along each channel, where *C* is the number of channels. The methods are categorized into four methods, depending on where the combination is performed within the networks.

3.2.1. Embedding-Level Combination

Figure 3b illustrates the embedding-level combination method. We used the output of GAP layer as embedding. In this method, two embeddings obtained from each pair of segments are combined to form one embedding. The combined embedding is then fed into the classifier. We combine the two embeddings with three approaches: concatenating (denoted as *concat*), element-wise maximum (denoted as *vmax*), and element-wise averaging (denoted as *vmean*). Let $\mathbf{u} = [u_1, \ldots, u_D]^T$ and $\mathbf{v} = [v_1, \ldots, v_D]^T$ be two embedding vectors, and $\mathbf{w} = [w_1, \ldots, w_D]^T$ be the output. *vmax* computes the element-wise maximum, so the output \mathbf{w} becomes $\mathbf{w} = \max(\mathbf{u}, \mathbf{v})$, where $w_i = \max(u_i, v_i)$. *vmean* computes the element-wise average, so the output \mathbf{w} becomes $\mathbf{w} = 0.5(\mathbf{u} + \mathbf{v})$ (i.e., just vector addition followed by scalar multiplication), where $w_i = 0.5(u_i + v_i)$. The dimensionality of the combined embedding is doubled when using *concat*, indicating that the number of parameters for the classifier has doubled. In contrast, when using *vmax* or *vmean*, the combined embedding has the same dimensionality as each original embedding, so no additional parameter is required.

3.2.2. Feature Map-Level Combination

Figure 3c illustrates the feature-map-level combination method. The output of the CNN module with the shape of $(C \times H \times W)$ corresponds to a set of *C* feature maps, where *C* is the number of output channels of the last convolution layer in the module, and *H* and *W* are the height and width of each feature map, respectively. Two sets of feature maps can be obtained from two segments in a pair. We combine these two feature maps using an element-wise maximum (denoted as *fmax*). We do not consider other operations for combination, such as concatenating and element-wise averaging per channel, because concatenating and element-wise averaging at the feature map level produce the same results as those at the embedding level (i.e., *concat* and *vmean*, respectively).

3.2.3. Two-Channel Input

Figure 3d illustrates the two-channel input method. Most acoustic features from utterances in the form of spectrograms, corresponding to a CNN input in many cases, are in 2D matrix form with the shape of $(H \times W)$. Such a 2D matrix can be viewed as a 3D tensor with the shape of $(1 \times H \times W)$, where the number of channels is 1. We propose the two-channel input method (denoted as *2ch*), where two segments in a pair are concatenated along the channel axis to form one 3D tensor with the shape of $(2 \times H \times W)$. This concatenated 3D tensor is then used as a CNN input. For receiving an input with two channels, the number of kernels of the first convolution layer of the CNN module must be doubled. Consequently, the number of parameters for the first convolution layer doubles.

The CNN designed for one-channel inputs can also receive two-channel inputs without increasing the number of kernels by sharing the kernels across all input channels (denoted as $2ch_s$). In this method, therefore, the number of parameters is not changed. However, we do not consider the $2ch_s$ method because of its lower gain in reducing detection error. The comparison of the detection errors of 2ch and $2ch_s$ will be discussed in the last part of Section 5.2.

3.2.4. Statistics-Level Combination

The statistics-level combination method is only for an x-vector network [33,34], one of the models used in our experiments. Because the x-vector network cannot receive inputs with multiple channels, the feature map-level combination and two-channel input methods cannot be used with the x-vector network.

Accordingly, we propose another statistics-level combination method (denoted as *statc*) that can be used only for the x-vector network. The statistics pooling layer originally computes the statistics (e.g., mean and standard deviation) over all the *M* output frames from one (forward) segment. In contrast, with the *statc* method, the statistics pooling layer computes the statistics over 2*M* output frames, of which some *M* frames are from a forward segment and the other *M* frames are from a backward segment.

4. Experiments

4.1. Database

We used the ASVspoof 2019 database [35]. It includes two task scenarios: logical access (LA) and physical access (PA).

The LA scenario addresses the spoofing attacks generated by TTS synthesis and VC. It is divided into the training (2580 genuine and 22,800 spoofed speeches), development (2548 genuine and 22,296 spoofed speeches), and evaluation (7355 genuine and 63,882 spoofed speeches) sets. The utterance length is approximately 3.25 s on average and ranges from 0.47 to 13.19 s with a standard deviation of 1.47. Figure 4a illustrates the histogram of the utterance lengths in the LA scenario.



Figure 4. The histogram of the utterance lengths in ASVspoof (a) LA and (b) PA scenarios.

The PA scenario addresses the spoofing attacks generated by replay attack methods. It is divided into the training (5400 genuine and 48,600 spoofed speeches), development (5400 genuine and 24,300 spoofed speeches), and evaluation (18,090 genuine and 116,640 spoofed speeches) sets. The utterance length is approximately 4.28 s on average and ranges from 1.46 to 10.32 s with a standard deviation of 1.2. Figure 4b illustrates the histogram of the utterance lengths in the PA scenario.

The training sets were used to build spoofing detection systems. The development sets were used to validate the detection errors of the systems for every epoch. The evaluation sets were used to evaluate the detection error rates of the system.

4.2. Experimental Setup

We used a 257-dimensional log power spectrogram as input feature for all types of CNNs used in our experiments. For each utterance, 25 ms frames were extracted at 10 ms intervals. The frame-level preprocessing was performed in the following order: removing the DC offset, pre-emphasis filtering with a coefficient of 0.97, and applying the Hamming window. The number of fast Fourier transform (FFT) points was 512. We performed neither voice activity detection (VAD) [36] nor cepstral mean and variance normalization (CMVN) for the spectrogram. The Kaldi toolkit [37] was used to extract the feature.

For segmentation, we set the length of segment *M* to 200 (corresponding to 2 s), 400 (corresponding to 4 s), and 600 (corresponding to 6 s). Approximately 81%, 25%, and 5% of utterances in the LA subset are longer than 2, 4, and 6 s, respectively. Approximately 99%, 56%, and 8% of utterances in the PA subset are longer than 2, 4, and 6 s, respectively. The shift interval *L* was half of *M*, which is based on the claim in [26] that segmentation with overlap generally outperforms that without overlap (i.e., L = M).

We used six types of CNN-based models to build spoofing detection systems: SE-ResNet, x-vector network, dense convolutional network (DenseNet) [38], MobileNetV2 [39], ShuffleNetV2 [40], and MNASNet [41] designed using an approach of automated mobile neural architecture search (MNAS). All models used in our experiments have a fully connected (FC) softmax classifier at the top. The number of classes is three (i.e., genuine, TTS, and VC) for the LA, and two (i.e., genuine and replayed) for the PA. AMSGrad [42], a variant of the Adam [43] optimizer, was used to minimize cross-entropy loss. The hyper-parameters for the optimizer were as follows: a learning rate of 10^{-3} , $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a weight decay of 10^{-4} . All weights were initialized from the He normal distribution [44] and no bias was used. We trained the networks for 100 epochs with a mini-batch size of 64. PyTorch [45] with the torchvision.models package was used to implement the systems.

We used the equal error rate (EER) as the evaluation metric. The lower the EER, the lower the detection error. For the LA, where the number of classes is three, the score of each segment was defined as the log probability of the genuine class log $p_g(x)$, corresponding to the log-softmax output for that class. For the PA, where the number of classes is two, the score of each segment was defined as the log ratio of the spoof probability to the genuine

probability, log $p_g(x) - \log p_s(x)$. The score of each utterance is computed by averaging the scores of all the segments in the utterance.

5. Results

5.1. Experimental Results and Discussion

Table 1 presents the EERs of the baseline and proposed systems on the development and evaluation trials of ASVspoof 2019 LA and PA tasks. For each system (corresponding to the combination of *model* and *method*), we selected the one among 100 epochs that exhibited the lowest EER in the development trials. As guidance, the lowest EERs in the evaluation trials (i.e., the oracle EERs) were additionally reported in Appendix A. The *Fusion* method indicates the score-level fusion of all the proposed systems using simply summation. Unless otherwise noted, the explanations are focused on the results of the evaluation trials.

Table 1. EERs (%) of systems on the development and evaluation trials of ASVspoof 2019 LA and PA scenarios. For each model, the lowest EER is highlighted in bold (comparison including *Fusion*) or is underlined (comparison excluding *Fusion*).

	LA						PA						
Model	Method	<i>M</i> =	= 200	<i>M</i> =	= 400	<i>M</i> =	= 600	<i>M</i> =	= 200	<i>M</i> =	= 400	<i>M</i> =	: 600
		Dev.	Eval.	Dev.	Eval.	Dev.	Eval.	Dev.	Eval.	Dev.	Eval.	Dev.	Eval.
	Baseline	0	14.099	0	10.388	0	12.551	2.152	3.197	0.811	1.255	0.519	0.985
	concat	0	15.486	0	11.148	0	<u>9.410</u>	1.261	2.641	0.370	0.945	0.612	1.007
	vmax	0	<u>13.635</u>	0	12.157	0	10.633	1.279	2.118	0.704	1.327	0.998	1.975
SE-ResNet	vmean	0	13.637	0	<u>9.081</u>	0	12.211	1.573	3.300	0.593	1.337	0.926	1.702
	fmax	0	15.812	0	9.868	0	13.353	1.388	2.875	0.534	1.105	0.484	0.934
	2ch	0	13.923	0	11.434	0	12.498	2.538	4.378	0.597	1.063	0.316	0.912
	Fusion	0	13.131	0	9.001	0	10.236	0.737	1.940	0.261	0.653	0.168	0.614
	Baseline	0.009	17.321	0	13.395	0	6.947	3.965	8.297	1.797	3.538	0.908	1.664
X-vector	concat	0.009	15.690	0	<u>7.900</u>	0	<u>6.718</u>	1.762	3.964	0.667	1.520	0.834	1.366
Network	vmax	<u>0</u>	<u>14.750</u>	0.002	8.171	0	8.811	1.296	3.936	0.721	1.072	0.721	1.056
(TDNN)	vmean	0.082	16.558	0	11.081	0	7.763	2.170	3.897	0.797	1.708	0.856	1.520
(1DININ)	statc	0.040	16.927	0	8.620	0.002	12.143	2.037	3.892	0.926	1.752	0.797	1.156
	Fusion	0.004	14.451	0	8.389	0	8.552	0.628	2.205	0.444	0.950	0.610	0.902
	Baseline	0	16.232	0	10.130	0	9.818	2.296	3.155	0.850	1.360	0.337	0.713
	concat	0	12.120	0	<u>7.139</u>	0	11.405	0.854	1.593	0.353	0.658	0.296	0.580
	vmax	0	<u>11.109</u>	0	10.744	0	<u>8.743</u>	0.700	<u>1.116</u>	0.366	0.857	0.222	0.492
DenseNet	vmean	0	12.509	0	12.628	0	12.158	1.090	1.708	<u>0.296</u>	0.564	0.238	0.476
	fmax	0	14.279	0	14.535	0	11.828	<u>0.630</u>	1.609	0.333	0.845	0.242	0.508
	2ch	0	13.963	0	9.258	0	11.951	2.164	2.798	0.370	0.956	0.226	0.542
	Fusion	0	11.027	0	7.410	0	10.023	0.444	0.885	0.148	0.425	0.129	0.303
	Baseline	0	19.511	0	11.299	0	8.292	2.667	4.666	1.076	2.233	0.409	0.785
	concat	0	13.584	0	8.307	0	10.414	1.649	3.466	0.667	1.597	0.386	0.878
	vmax	0	12.794	0	10.223	0	8.767	1.333	3.217	1.022	2.206	<u>0.279</u>	0.872
MobileNetV2	vmean	0	<u>12.387</u>	0	8.280	0	10.416	1.185	<u>2.996</u>	0.799	1.873	0.501	1.039
	fmax	0	13.433	0	9.950	0	8.768	1.076	3.228	0.756	1.592	0.388	0.928
	2ch	0	14.167	0	10.727	0	8.537	1.889	4.533	0.337	0.962	<u>0.279</u>	0.901
	Fusion	0	10.809	0	7.815	0	8.363	0.741	2.387	0.407	0.978	0.185	0.520
	Baseline	0	20.479	0	11.748	0	11.051	4.170	4.374	1.392	2.255	0.904	1.382
	concat	0	<u>13.664</u>	0	<u>9.433</u>	0	8.849	1.240	2.161	0.462	0.917	0.388	0.867
	vmax	0	19.443	0	10.265	0	10.007	1.146	2.102	0.503	0.823	0.388	0.746
ShuffleNetV2	vmean	0	17.975	0	9.705	0	9.912	1.390	2.366	0.335	0.869	0.353	0.696
	fmax	0	18.204	0	9.571	0	10.048	1.094	2.123	0.514	1.128	0.412	0.779
	2ch	0.002	19.798	0	10.472	0.038	9.764	2.168	3.737	0.739	1.260	0.593	0.951
	Fusion	0	13.923	0	8.744	0	8.620	0.760	1.581	0.279	0.603	0.207	0.492
	Baseline	0	19.050	0	10.608	0	11.205	2.263	5.052	0.834	2.232	0.353	0.995
	concat	0	<u>11.652</u>	0	10.249	0	11.624	1.207	3.570	0.255	0.912	0.370	0.614
	vmax	0	17.362	0	11.895	0	11.107	1.630	4.416	0.353	1.156	0.168	0.718
MNASNet	vmean	0	14.333	0	7.207	0	8.863	1.316	4.388	0.279	1.321	0.164	0.685
	fmax	0	15.894	0	7.570	0	11.327	1.037	3.642	0.316	0.969	0.152	0.615
	2ch	0	17.933	0	9.341	0	10.331	1.540	3.687	0.409	1.238	0.240	0.857
	Fusion	0	13.705	0	7.029	0	9.463	0.663	2.732	0.148	0.636	0.094	0.359

The experimental results in Table 1 indicate that the proposed method outperforms the conventional feature segmentation on spoofing detection, both the LA and PA tasks. In most cases, the proposed method exhibited lower EERs than the baseline with suitable combination methods, regardless of the type of model, segment length M, and task. There is an exceptional case where the proposed method failed to reduce EER for MobileNetV2 with M = 600. This result is mainly due to the limitation of the validation (especially for the LA task), rather than the proposed method's ineffectiveness at reducing the EER. This issue is explained in Appendix A. Our experiments provide evidence that the discriminative attributes between genuine and spoofed utterances are spread across the entire time domain, rather than concentrated primarily in some specific ranges, as stated in Section 1. The performance would not be improved with the proposed method if the discriminative attributes were concentrated in some specific time ranges. However, even if the attributes are spread over the entire time, we could not get lower EERs if the attributes in different time ranges do not have distinct information in terms of the detection. Therefore, the proposed method is efficient for the cases where the distinct information is spread across the entire time domain, such as spoofing detection.

We discuss the results in Table 1 with respect to the segment length *M*. The longer the *M*, the lower the EERs in most cases for both baseline and proposed systems. It is consistent with our intuition that the longer segment has more information than the shorter segment (if *M* becomes even longer (e.g., thousands of frames or more), EER could become saturated, but we do not compare with the cases where *M* is extremely long because it is rarely considered because of the limited computing power and memory usage). Some exceptional cases were observed for the LA task where the EERs were increased on average, although *M* lengthened from 400 to 600: with SE-ResNet, DenseNet (only with the proposed methods), and MNASNet. These cases are discussed in Appendix A.

We compared the baseline with a target length M_b and the proposed methods with the target length M_p shorter than M_b (i.e., $M_p < M_b$). For $M_b = 400$ and $M_p = 200$, it could be expected that the proposed systems exhibit similar EERs to the baseline systems, because the total numbers of frames fed at one time are the same. Following the same logic, it also could be expected for $M_b = 600$ and $M_p = 400$ that the proposed systems exhibit lower EERs to the baseline systems because the total number of frames fed at one time is greater with the proposed systems than the baseline systems. However, the proposed systems with $M_p = 200$ generally exhibited EERs higher than those of the baseline systems with $M_b = 400$. The proposed systems with $M_p = 400$ exhibited slightly lower but similar levels of EERs as the baseline systems with $M_b = 600$ on average. Recall that the amount of information with the proposed method is usually less than twice the amount with the baseline method because of the partially overlapped time ranges, as described in Section 3.2. Accordingly, the above results are acceptable and do not undermine our expectation that the longer M, the more available information at one time, and thus the lower the EERs.

The above results illustrated that the effectiveness of the proposed method on reducing EERs depends on *M*. We empirically found that the proposed method was more effective at reducing EER as *M* was closer to the average length. For M = 200, 33 out of the 35 proposed systems (including *Fusion*) exhibited lower EERs than the baseline in both the LA and PA tasks. The EERs were reduced on average by approximately 17.2% (-12.1% to 44.6%) for the LA task and 35.6% (-36.9% to 73.4%) for the PA task. For M = 400, 28 and 33 out of the 35 proposed systems exhibited lower EERs than the baseline in the LA and PA tasks, respectively. The EERs were reduced on average by approximately 13.4% (-43.5% to 41%) for the LA task and 43.8% (-6.5% to 73.3%) for the PA task. When *M* is longer than the average, the proposed method exhibited a lower gain in reducing EER. For M = 600, 18 out of the 35 proposed systems exhibited higher EERs than the baseline for the LA task. The EERs for the LA task increased on average by approximately 2.7% (i.e., decreased approximately -2.7%, range from -74.8% to 25%) when using the proposed method. For the PA task, 27 out of the 35 proposed systems exhibited lower EERs than the baseline. The EERs were reduced on average by approximately 2.5% to 64.4%). The proposed

method was most effective for reducing EER when M = 200 for the LA task and when M = 400 for the PA task. These results can be interpreted as follows. For the features longer than M, the proposed method enables the CNN to consider more information at one time. In contrast, the proposed method has a negligible effect for features shorter than M because the CNN already considers the information of the complete feature at one time for these short features, even without the proposed method. Thus, for the features shorter than M, there is no longer room to increase the amount of information available at one time.

We provide a rough guidance to set a proper *M* when using the proposed method without evaluation by defining the additional gain $\alpha = M\rho$, where ρ is the proportion of the number of the features longer than M in the dataset (stated in Section 4.2). For the LA dataset, where the average number of frames is 325, $\alpha = 200 \times 0.81 = 162$ when M = 200 (i.e., 81% of features have frames more than 200), $\alpha = 400 \times 0.25 = 100$ when M = 400 (i.e., 25% of features have frames more than 400), and $\alpha = 600 \times 0.05 = 30$ when M = 600 (i.e., 5% of features have frames more than 600). The average EER reductions are approximately 17.2%, 13.4%, and -2.7% (i.e., the EERs were increased by approximately 2.7% on average) when M = 200, M = 400, and M = 600, respectively. For the PA dataset, where the average number of frames is 428, $\alpha = 200 \times 0.99 = 198$ when M = 200 (i.e., 99%) of features have frames more than 200), $\alpha = 400 \times 0.56 = 224$ when M = 400 (i.e., 56% of features have frames more than 400), and $\alpha = 600 \times 0.08 = 48$ when M = 600 (i.e., 8% of features have frames more than 600). The average EER reductions are approximately 35.6%, 43.8%, and 18.2% when M = 200, M = 400, and M = 600, respectively. It is reasonable to infer that the higher the α , the higher the efficiency of the proposed method on reducing EER. Therefore, we recommend considering α to find a proper value of M when using the proposed method.

Next, we discuss the results in terms of the combination method. Contrary to our prediction that some specific combination methods would achieve lower EERs than others, there is no consistent tendency in terms of the combination method. Instead, the optimal combination method (i.e., the methods that showed the EER underlined in Table 1) differed for each case. Furthermore, in many cases, there were mismatches in the optimal combination method between the development and evaluation trials, although the other conditions (e.g., model, *M*, and task) were the same. Although *Fusion* achieved the lowest EER in most cases or EERs close to the lowest, it is inefficient because it requires multiple networks with different combination methods. Therefore, it is encouraged to develop a new combination method to achieve stable performances (e.g., consistent tendency across trials/conditions), left for future research.

5.2. Ablation Study

Throughout this paper, we have claimed that the proposed method reduces detection errors by increasing the amount of information available at one time. In this section, we support this claim through additional experiments in the following three factors: time range, flipping, and randomness of target length. We set the segment length *M* to 400 for all the experiments.

In Section 3.2, we doubted whether the proposed method still can reduce the detection errors when two segments in a pair always cover the same time range. To solve this doubt, we first compared the EERs between the proposed systems and its variations where both segments in a pair always cover the same time range (denoted as *ST*). Table 2 illustrates the oracle EERs of systems with the proposed method and *ST* for the LA and PA tasks, respectively. We focus on the results in the evaluation trials. If the detection errors are reduced even with *ST*, as with the proposed method, we no longer claim that the mechanism of the proposed method reducing the errors is to increase the amount of information available at one time. Instead, it might be because flipping the frame order has an effect of data augmentation, which led to the error reduction.

Table 2. Oracle EERs (%) of systems on the development and evaluation trials of ASVspoof 2019 LA and PA scenarios, with the proposed and *ST* methods. For each model, the lowest EER is highlighted in bold.

			L	Α		PA				
Model	Method	Prop	osed	S	5T	Prop	osed	ST		
	_	Dev.	Eval.	Dev.	Eval.	Dev.	Eval.	Dev.	Eval.	
	concat	0	7.967	0	8.484	0.370	0.708	1.329	1.835	
	vmax	0	9.517	0	8.989	0.704	1.327	0.983	1.520	
	vmean	0	9.081	0	9.012	0.593	1.156	0.945	1.414	
SE-ResNet	fmax	0	7.576	0	8.893	0.534	1.105	1.238	1.913	
	2ch	0	9.261	0	7.941	0.597	1.040	1.403	1.896	
	average	0	8.680	0	8.664	0.560	1.067	1.180	1.716	
	concat	0	6.352	0	8.267	0.667	1.493	1.760	3.108	
X-vector	vmax	0.002	6.741	0.004	8.674	0.721	1.027	1.630	2.665	
Network	vmean	0	7.804	0	10.524	0.797	1.497	1.817	3.814	
(TDNN)	statc	0	6.309	0	9.488	0.926	1.581	1.630	2.924	
	average	0.001	6.802	0.001	9.238	0.778	1.400	1.709	3.128	
	concat	0	6.868	0	6.472	0.353	0.542	0.908	1.194	
	vmax	0	6.717	0	6.499	0.366	0.719	1.133	1.414	
DoncoNot	vmean	0	7.981	0	7.108	0.296	0.564	1.148	1.587	
Denserver	fmax	0	7.628	0	5.944	0.333	0.702	0.908	1.332	
	2ch	0	7.828	0	7.548	0.370	0.796	0.737	0.961	
	average	0	7.404	0	6.714	0.344	0.665	0.967	1.298	
	concat	0	6.067	0	6.717	0.667	1.398	1.004	2.034	
	vmax	0	7.439	0	5.710	1.022	2.150	0.595	1.731	
	vmean	0	7.533	0	8.239	0.799	1.847	1.037	2.311	
MobileNetV2	fmax	0	6.186	0	6.172	0.756	1.371	0.815	1.979	
	2ch	0	6.242	0	7.738	0.337	0.935	0.980	2.217	
	average	0	6.693	0	6.915	0.716	1.540	0.886	2.054	
	concat	0	6.609	0	9.166	0.462	0.885	1.649	1.924	
	vmax	0	7.032	0	6.564	0.503	0.823	1.630	2.299	
ChuffleNetV2	vmean	0	6.405	0	7.136	0.335	0.869	1.353	2.278	
Situmervetv2	fmax	0	7.454	0	7.330	0.514	1.061	1.294	1.869	
	2ch	0	8.786	0	8.144	0.739	1.227	1.702	2.846	
	average	0	7.257	0	7.668	0.511	0.973	1.526	2.243	
	concat	0	6.080	0	6.079	0.255	0.774	0.908	1.974	
	vmax	0	5.398	0	5.452	0.353	0.818	0.651	1.951	
MNASNet	vmean	0	5.384	0	7.098	0.279	0.923	0.963	2.427	
IVII NAJINEL	fmax	0	4.840	0	4.756	0.316	0.846	0.776	2.095	
	2ch	0	6.840	0	6.349	0.409	1.039	0.926	2.869	
	average	0	5.708	0	5.947	0.322	0.880	0.845	2.263	

For the LA task, the proposed method exhibited lower EERs than ST, approximately 3.2% on average (-30.3% to 33.5%). Compared with the baseline, the proposed method exhibited lower EERs, approximately 8.2% on average, where 23 out of 29 proposed systems (except *average*) achieved the lower EERs. ST exhibited lower EERs than the baseline, approximately 2.5% on average, where 14 out of 29 systems exhibited the lower EERs with *ST*. Contrary to our prediction, there was no dominant performance gap between the proposed method and *ST*, albeit these results are still in accordance with our claim on the mechanism of the proposed method.

Unlike in the LA task, *ST* does not helpfully reduce EER at all for the PA task. The proposed method exhibited considerably lower EERs than *ST* for all cases, except the system of the MobileNetV2 with *vmax*. The EERs of the proposed methods are significantly lower than those of *ST*, approximately 46% on average (-24.2% to 64.5%). Moreover, compared with the baseline, the 27 out of 29 systems exhibited lower EERs when using the

proposed method, approximately 43.2% on average, whereas 13 out of 29 systems exhibited lower EERs when using *ST* than the baseline, approximately merely 6.3% on average.

To further analyze the effect of the flipping the frame order, we conducted additional experiments to compare the oracle EERs between the baseline, proposed, backward-only (denoted as *BO*), and data-augmented (denoted as *Augment*) systems. *BO* is a variant of the baseline where only the backward segments \mathbf{B}_i were used in both the training and evaluation phases, not the forward segments \mathbf{F}_i . *Augment* is identical to the baseline, but the total quantity of data is exactly twice the baseline in both the training and evaluation phases. These doubled amounts of data correspond to all the forward and backward segments, obtained by bidirectional feature segmentation. *Augment* treats two segments in each pair as individual data, not receiving them at one time. Therefore, the baseline, *BO*, and *Augment* have the same amount of information available at one time.

Table 3 presents the oracle EERs of the baseline, proposed, *BO*, and *Augment* systems in the development and evaluation trials of the LA and PA tasks. For the proposed systems, we presented the average of the EERs obtained with each combination method (excluding *Fusion*).

Table 3. Oracle EERs (%) of systems in the development and evaluation trials of ASVspoof 2019 LA and PA, with four different methods (i.e., baseline, *BO*, *Augment*, and proposed). For each model, the lowest EER is highlighted in bold.

Madal	Mathad	I	_A	PA		
Model	Method -	Dev.	Eval.	Dev.	Eval.	
	Baseline	0	8.211	0.811	1.255	
CE DasNat	BO	0	8.347	1.279	1.763	
SE-ResNet	Augment	0	7.940	1.316	2.443	
	Proposed	0	8.680	0.560	1.067	
V waster	Baseline	0	8.157	1.797	3.538	
A-vector Notricork	BO	0	9.557	1.464	2.858	
(TDNN)	Augment	0	10.415	1.595	4.145	
(1DMN)	Proposed	0	6.802	0.778	1.400	
	Baseline	0	9.095	0.850	1.360	
DemasNat	BO	0	6.705	1.131	1.581	
DenselNet	Augment	0	6.947	1.094	1.713	
	Proposed	0	7.404	0.344	0.776	
	Baseline	0	7.695	1.076	1.885	
M. 1. 1. N. (MO	BO	0	7.791	0.945	1.763	
MobileNetv2	Augment	0	7.043	0.926	2.521	
	Proposed	0	6.693	0.716	1.540	
	Baseline	0	7.303	1.392	2.102	
Chereff a Nati VO	BO	0	8.985	1.514	2.024	
Snumervet v 2	Augment	0	6.293	1.499	2.549	
	Proposed	0	7.257	0.511	0.973	
	Baseline	0	6.092	0.834	2.224	
MNIACNIat	BO	0	6.960	0.834	2.100	
minASinet	Augment	0	7.400	0.867	2.632	
	Proposed	0	5.708	0.322	0.880	

For the evaluation trials of the LA task, both *BO* and *Augment* exhibited higher EERs than the proposed systems on average, higher by approximately 14.9% (-9.4% to 40.5%) and 10% (-13.3% to 53.1%), respectively. Compared to the baseline systems, *BO* exhibited higher EERs, approximately 5.2% (-26.3% to 23%) on average. *Augment* exhibited almost the same EERs on average as the baseline, lower by only approximately 0.01% (-27.7 to 23.6%).

For the PA task, the proposed method always exhibited the lowest EERs by large margins. In the development trials, *BO* exhibited higher EERs on average, by approximately 11.5% (-18.5% to 57.7%) and 138.8% (32% to 228.8%) than the baseline and proposed systems, respectively. *Augment* exhibited higher EERs on average, by approximately 12.9% (-13.9% to 62.3%) and 141.7% (29.3% to 218%) than the baseline and proposed systems, respectively. In the evaluation trials, *BO* exhibited higher EERs on average, by approximately 3.6% (-19.2% to 40.5%) and 89% (14.5% to 138.6%) than the baseline and proposed systems, respectively. *Augment* exhibited higher EERs on average, by approximately 35.2% (17.2% to 94.7%) and 145.1% (63.7% to 199.1%) than the baseline and proposed systems, respectively.

As presented in Tables 2 and 3, it can be observed that the proposed method generally exhibited lower EERs than *ST*, *BO*, and *Augment* in both the LA and PA tasks. In the LA task, the average performance gains with *ST*, *BO*, *Augment*, and the proposed method compared to the baseline are 2.5%, -5.2%, 0.01%, and 8.2%, respectively. In the evaluation trials of the PA task, the average performance gains with *ST*, *BO*, *Augment*, and the proposed method compared to the baseline are 6.3%, -3.6%, -35.2%, and 46%, respectively. These results support the claim that it is important to increase the amount of available information at one time for robust spoofing detection. The proposed method can satisfy this requirement, which is difficult, by merely flipping the frame order.

The next experiment compares the EERs of 2*ch* and 2*ch_s*, introduced in Section 3.2.4. Table 4 presents the oracle EERs of systems on the development and evaluation trials of the LA and PA tasks. The x-vector network was excluded in this experiment because it cannot receive a multi-channel input, as explained in Section 3.2.3.

Model	Mathad	L	.A	PA		
	Methou	Dev.	Eval.	Dev.	Eval.	
	2ch	0	9.261	0.597	1.040	
SE-ResNet	$2ch_s$	0	8.226	0.996	1.731	
DenseNet	2ch	0	7.828	0.370	0.796	
	2ch_s	0	9.257	1.203	1.527	
M. 1. 1. N. (M2	2ch	0	6.242	0.337	0.935	
WiddlielNet V 2	2ch_s	0	8.267	0.869	1.863	
ShuffleNetV2	2ch	0	8.786	0.739	1.227	
	2ch_s	0	7.775	1.538	1.958	
MNASNet	2ch	0	6.840	0.409	1.039	
	2ch_s	0	8.620	0.869	1.885	

Table 4. Oracle EERs (%) of systems (excluding the x-vector network) in the development and evaluation trials of ASVspoof 2019 LA and PA, with the methods of *2ch* and *2ch_s*. For each model, the lowest EER is highlighted in bold.

For the LA task, $2ch_s$ exhibited higher EERs than 2ch in the evaluation trials, by approximately 10.8% on average (-11.2% to 32.4%). Three of the five models exhibited lower EERs with 2ch than with $2ch_s$.

For the PA task, *2ch_s* exhibited distinctly higher EERs than *2ch*, regardless of the model. It exhibited approximately 134.1% (66.8% to 225.1%) and 79.7% (59.6% to 99.3%) higher EERs in the development and evaluation trials, respectively. *2ch* exhibited lower EERs than *2ch_s*. Consistent with our expectation, this is due to the additional parameters that provide different modeling from the existing parameters at the input feature level.

The last experiments were conducted to demonstrate that the proposed method is better than on-the-fly data loader on spoofing detection, as introduced in Section 1. Table 5 illustrates the oracle EERs of the baseline, on-the-fly data loader (denoted as *OTF*), and proposed systems in the development and evaluation trials of the LA and PA tasks. For the on-the-fly data loader, we sampled the target length for each iteration from a uniform distribution on the interval [200, 600] to match the average number of frames that CNNs take at one time with the baseline.

Model	Mathad	L	.A	PA		
Widdel	Methou	Dev.	Eval.	Dev.	Eval.	
	Baseline	0	8.211	0.811	1.255	
SE-ResNet	OTF	0	9.626	1.292	1.787	
	Proposed	0	8.680	0.560	1.067	
X-vector	Baseline	0	8.157	1.797	3.538	
Network	OTF	0.002	8.158	1.723	3.455	
(TDNN)	Proposed	0	6.802	0.778	1.400	
	Baseline	0	9.095	0.850	1.360	
DenseNet	OTF	0	7.162	1.111	1.393	
	Proposed	0	7.404	0.344	0.776	
	Baseline	0	7.695	1.076	1.885	
MobileNetV2	OTF	0	7.163	1.111	2.498	
	Proposed	0	6.693	0.716	1.540	
	Baseline	0	7.303	1.392	2.102	
ShuffleNetV2	OTF	0	10.200	1.589	2.493	
	Proposed	0	7.257	0.511	0.973	
	Baseline	0	6.092	0.834	2.224	
MNASNet	OTF	0	5.914	0.996	2.709	
	Proposed	0	5.708	0.322	0.880	

Table 5. Oracle EERs (%) of systems in the development and evaluation trials of ASVspoof 2019 LA and PA, with three different methods (i.e., baseline, *OTF*, and proposed). For each model, the lowest EER is highlighted in bold.

For the evaluation trials of the LA task, the proposed systems showed relatively 10.3% (-3.4% to 28.9%) lower EERs than *OTF* on average. Compared to the baseline systems, *OTF* exhibited higher EERs, approximately 4.3% (-21.3% to 39.7%) on average, whereas the proposed systems exhibited lower EERs, approximately 8.2% (-5.7% to 18.6%), than the baseline systems on average.

For the PA task, the proposed method showed relatively 58.6% (35.6% to 69%) and 51.8% (38.4% to 67.5%) lower EERs than *OTF* on average in the development and evaluation trials, respectively. Compared to the baseline systems, *OTF* exhibited higher EERs, approximately 20.4% (-4.1% to 59.3%) and 19.2% (-2.3% to 42.4%) on average, in the development and evaluation trials, respectively. In contrast, the proposed systems showed significantly lower EERs, approximately 50.9% (30.9% to 63.3%) and 41.8% (15% to 60.4%) on average, in the development and evaluation trials, respectively.

6. Conclusions

We proposed a bi-point input method based on bidirectional feature segmentation. It enables CNNs that receive variable-length inputs to increase the amount of available information at one time. Each feature is decomposed into two sets of segments (i.e., forward and backward segment sets) along two directions (i.e., positive and negative time directions). Then, the CNN receives a pair of two segments (i.e., one forward segment and one backward segment) as an input at one time. We also evaluated various combination methods. The results from both segments are combined using one of the proposed combination methods before being fed into the final classifier to aggregate the information from both segments. The proposed method achieved lower EERs than the conventional method on both the LA and PA tasks of ASVspoof 2019, while minimizing changes in the network structure. Based on these results, we claim that the proposed method with suitable combination methods can function as robust spoofing detection systems by increasing the amount of information available at one time. We also provided a rough guidance to set a proper segment length without evaluation.

In the future, we will extend the proposed method to receive more than two segments at one time, for a further reduction in detection errors. To achieve a stable performance without the fusion, we will investigate how to aggregate the information in all the pairs of segments more efficiently. Moreover, we will apply our proposed method to different tasks, such as speaker verification, acoustic scene classification, and others that use different types of time-series input other than audio. We predict that the proposed method would also benefit tasks where the discriminative attributes between the classes are spread across the entire time domain, like spoofing detection.

Author Contributions: Conceptualization, S.Y.; methodology, S.Y.; formal analysis, S.Y.; investigation, S.Y.; writing—original draft preparation, S.Y.; writing—review and editing, H.-J.Y.; supervision, H.-J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a research grant from Kongju National University in 2022.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Appendix A. The Oracle Evaluation Results

Table A1 presents the oracle EERs of the baseline and proposed systems in the evaluation trials, where each system is the one among 100 epochs that exhibited the lowest EER in the evaluation trials rather than the development trials (i.e., Table 1). As already depicted in Table 1 in Section 5.1, almost all of the development EERs of the LA task were zero, even though the evaluation EERs were considerably far from zero. Furthermore, most systems easily achieved zero development EER for the LA task for most epochs. These results indicate it is easy to achieve zero EER for the development trials of the LA task, even with inadequate systems in the evaluation trials. Therefore, especially for the LA task, there is a limitation that model selection based on the development EER could be prone to failing to choose adequate systems in the evaluation trials. Consequently, even if the proposed system selected based on the development EERs presents a worse EER than the baseline in the evaluation trials, it is difficult to assert that the increase in the EER with the proposed method is due to whether the method is ineffective or a poor-performing system is chosen. Therefore, it is encouraged to interpret the results from the validation in the development trial (Table 1) with reference to the oracle results (Table A1), especially for the LA task.

In Section 5.1, we saw that there were exceptional cases for the LA task with SE-ResNet, DenseNet, and MNASNet. The first two cases are due to the limitation of the validation. Contrary to the EERs depicted in Table 1, the oracle EERs (i.e., in Table A1) were lower when M = 600 than when M = 400, like most other cases. For the case of MNASNet, however, even the oracle EERs were still higher when M = 600 than when M = 400. Further investigations should be conducted to understand this exception case. On average, the oracle evaluation EERs are relatively lower than the evaluation EERs shown in Table 1, by approximately 29.2% (18.4% to 46.2%) and 10% (2.1% to 21.6%) for the LA and PA tasks, respectively. It means that the validation based on the development trials has the limitation to select proper systems for the LA task, as our expectation. Overall, similar trends were observed between Tables 1 and A1. For example, the proposed method shows lower EERs than the baseline with suitable combination methods, but there is no consistent tendency in terms of the combination method. In addition, the proposed method is more effective to reduce EER as M is closer to the average length. These are common trends observed in both Tables 1 and A1. There is one opposite trend with Table 1 on the LA task that the proposed method showed higher oracle EERs than the baseline for SE-ResNet with M = 600, regardless of the combination method.

M. 1.1	Method -		LA		РА			
widdel	Method	M = 200	M = 400	M = 600	M = 200	M = 400	M = 600	
	Baseline	11.655	8.211	7.195	3.197	1.255	0.901	
	concat	10.156	7.967	7.956	2.127	0.708	1.007	
	vmax	<u>10.102</u>	9.517	7.480	2.118	1.327	1.503	
SE-ResNet	vmean	10.605	9.081	8.226	2.599	1.156	1.526	
	fmax	10.536	7.576	10.510	2.304	1.105	0.896	
	2ch	11.543	9.261	8.019	3.282	1.040	0.790	
	Fusion	10.335	8.038	7.313	1.643	0.619	0.553	
	Baseline	13.680	8.157	5.830	8.297	3.538	1.408	
X-vector	concat	12.618	6.352	5.820	3.964	1.493	1.166	
Notwork	vmax	<u>11.013</u>	6.741	7.396	3.936	<u>1.027</u>	1.028	
(TDNINI)	vmean	13.068	7.804	6.349	3.626	1.497	1.248	
(IDININ)	statc	12.071	6.309	7.028	3.555	1.581	1.156	
	Fusion	10.428	6.171	5.112	1.879	0.885	0.768	
DenseNet	Baseline	10.986	9.095	7.127	2.675	1.360	0.702	
	concat	10.129	6.868	6.378	1.393	0.542	0.557	
	vmax	8.484	6.717	6.841	1.116	0.719	0.492	
	vmean	10.075	7.981	7.699	1.708	0.564	0.469	
	fmax	9.762	7.628	6.881	1.470	0.702	0.493	
	2ch	9.966	7.828	6.497	2.510	0.796	0.486	
	Fusion	8.688	5.901	5.626	0.763	0.392	0.276	
	Baseline	9.939	7.695	7.507	4.223	1.885	0.785	
	concat	8.513	6.067	7.775	2.593	1.398	0.878	
	vmax	8.987	7.439	5.138	2.769	2.150	0.714	
MobileNetV2	vmean	9.029	7.533	7.599	2.587	1.847	0.967	
	fmax	9.720	6.186	5.806	2.029	1.371	0.874	
	2ch	9.669	6.242	8.279	3.344	<u>0.935</u>	0.692	
	Fusion	7.017	5.748	4.961	1.664	1.006	0.510	
	Baseline	17.919	7.303	6.744	4.374	2.102	1.283	
	concat	<u>12.101</u>	6.609	6.812	2.102	0.885	0.790	
	vmax	14.004	7.032	7.939	2.102	0.823	0.691	
ShuffleNetV2	vmean	15.892	6.405	6.637	2.366	0.869	0.696	
	fmax	12.397	7.454	8.840	2.123	1.061	0.768	
	2ch	15.309	8.786	6.706	3.444	1.227	0.951	
	Fusion	10.904	6.578	5.695	1.542	0.579	0.485	
	Baseline	9.384	6.092	8.064	4.019	2.224	0.857	
	concat	8.554	6.080	9.680	3.097	0.774	0.614	
	vmax	8.403	5.398	5.682	3.384	0.818	0.526	
MNASNet	vmean	9.027	5.384	5.970	2.935	0.923	0.525	
	fmax	8.539	4.840	7.709	3.185	0.846	0.525	
	2ch	8.403	6.840	7.575	3.582	1.039	0.452	
	Fusion	6.839	4.690	5.998	2.188	0.542	0.321	

Table A1. Oracle EERs (%) of systems on the evaluation trials of ASVspoof 2019 LA and PA scenarios. For each model, the lowest EER is highlighted in bold (comparison including *Fusion*) or is underlined (comparison excluding *Fusion*).

References

- Wu, Z.; Kinnunen, T.; Evans, N.; Yamagishi, J.; Hanilci, C.; Sahidullah, M.; Sizov, A. ASVspoof 2015: The first automatic speaker verification spoofing and countermeasures challenge. In Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association, Dresden, Germany, 6–10 September 2015; pp. 1–5.
- Kinnunen, T.; Sahidullah, M.; Delgado, H.; Todisco, M.; Evans, N.; Yamagishi, J.; Lee, K.A. The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection. In Proceedings of the 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, 20–24 August 2017; pp. 1–6.
- 3. Todisco, M.; Wang, X.; Vestman, V.; Sahidullah, M.; Delgado, H.; Nautsch, A.; Yamagishi, J.; Evans, N.; Kinnunen, T.; Lee, K.A. ASVspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv* **2019**, arXiv:1904.05441.

- Yamagishi, J.; Wang, X.; Todisco, M.; Sahidullah, M.; Patino, J.; Nautsch, A.; Liu, X.; Lee, K.A.; Kinnunen, T.; Evans, N.; et al. ASVspoof 2021: Accelerating progress in spoofed and deepfake speech detection. In Proceedings of the 2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge, Online, 16 September 2021; pp. 47–54.
- Kamble, M.R.; Patil, H.A. Novel energy separation based instantaneous frequency features for spoof speech detection. In Proceedings of the 2017 25th European Signal Processing Conference, Kos, Greece, 26 October 2017; pp. 106–110.
- Yoon, S.-H.; Koh, M.-S.; Park, J.-H.; Yu, H.-J. A new replay attack against automatic speaker verification systems. *IEEE Access* 2020, *8*, 36080–36088. [CrossRef]
- LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* 1998, 86, 2278–2324. [CrossRef]
- Yoon, S.; Yu, H.-J. Multiple-point input and time-inverted speech signal for the ASVspoof 2021 challenge. In Proceedings of the 2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge, Online, 16 September 2021; pp. 37–41.
- Tomilov, A.; Svishchev, A.; Volkova, M.; Chirkovskiy, A.; Kondratev, A.; Lavrentyeva, G. STC antispoofing systems for the ASVspoof2021 challenge. In Proceedings of the 2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge, Online, 16 September 2021; pp. 61–67.
- Chen, X.; Zhang, Y.; Zhu, G.; Duan, Z. UR channel-robust synthetic speech detection system for ASVspoof 2021. In Proceedings of the 2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge, Online, 16 September 2021; pp. 75–82.
- Benhafid, Z.; Selouani, S.A.; Yakoub, M.S.; Amrouche, A. LARIHS ASSERT reassessment for logical access ASVspoof 2021 challenge. In Proceedings of the 2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge, Online, 16 September 2021; pp. 94–99.
- LeCun, Y.; Bottou, A.; Orr, G.B.; Muller, K.R. Efficient backprop. In Neural Networks: Tricks of the Trade; Springer: Berlin/Heidelberg, Germany, 2012; pp. 9–48.
- Lai, C.; Abad, A.; Richmond, K.; Yamagishi, J.; Dehak, N.; King, S. Attentive filtering networks for audio replay attack detection. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 6316–6320.
- Wang, H.; Dinkel, H.; Wang, S.; Qian, Y.; Yu, K. Cross-domain replay spoofing attack detection using domain adversarial training. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 2938–2942.
- 15. Yoon, S.-H.; Yu, H.-J. A simple distortion-free method to handle variable length sequences for recurrent neural networks in text dependent speaker verification. *Appl. Sci.* 2020, *10*, 4092. [CrossRef]
- Cai, W.; Wu, H.; Cai, D.; Li, M. The DKU replay detection system for the ASVspoof 2019 challenge: On data augmentation, feature representation, classification, and fusion. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 1023–1027.
- Lavrentyeva, G.; Novoselov, S.; Tseren, A.; Volkova, M.; Gorlanov, A.; Kozlov, A. STC antispoofing systems for the ASVspoof2019 challenge. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 1033–1037.
- Alzantot, M.; Wang, Z.; Srivastava, M.B. Deep residual neural networks for audio spoofing detection. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 1078–1082.
- Wu, H.; Liu, S.; Meng, H.; Lee, H. Defense against adversarial attacks on spoofing countermeasures of ASV. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 6564–6568.
- von Platen, P.; Tao, F.; Tur, G. Multi-task Siamese neural network for improving replay attack detection. In Proceedings of the Interspeech, Shanghai, China, 26–29 October 2020; pp. 1076–1080.
- Parasu, P.; Epps, J.; Sriskandaraja, K.; Suthokumar, G. Investigating Light-ResNet architecture for spoofing detection under mismatched conditions. In Proceedings of the Interspeech, Shanghai, China, 26–29 October 2020; pp. 1111–1115.
- Monteiro, J.; Alam, J.; Falk, T.H. A multi-condition training strategy for countermeasures against spoofing attacks to speaker recognizers. In Proceedings of the Odyssey Speaker Language Recognition Workshop, Tokyo, Japan, 1–5 November 2020; pp. 296–303.
- Halpern, B.M.; Kelly, F.; van Son, R.; Alexander, A. Residual networks for resisting noise: Analysis of an embedding-based spoofing countermeasures. In Proceedings of the Odyssey Speaker Language Recognition Workshop, Tokyo, Japan, 1–5 November 2020; pp. 326–332.
- Chettri, B.; Kinnunen, T.; Benetos, E. Subband modeling for spoofing detection in automatic speaker verification. In Proceedings
 of the Odyssey Speaker Language Recognition Workshop, Tokyo, Japan, 1–5 November 2020; pp. 341–348.
- 25. Cai, W.; Chen, J.; Zhang, J.; Li, M. On-the-fly data loader and utterance-level aggregation for speaker and language recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 1038–1051. [CrossRef]
- Lai, C.; Chen, N.; Villalba, J.; Dehak, N. ASSERT: Anti-spoofing with squeeze-excitation and residual networks. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 1013–1017.
- Yoon, S.-H.; Yu, H.-J. Multiple points input for convolutional neural networks in replay attack detection. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 6444–6448.

- Wang, Q.; Lee, K.A.; Koshinaka, T. Using multi-resolution feature maps with convolutional neural networks for anti-spoofing in ASV. In Proceedings of the Odyssey Speaker Language Recognition Workshop, Tokyo, Japan, 1–5 November 2020; pp. 138–142.
- Yoon, S.-H.; Koh, M.-S.; Yu, H.-J. Phase spectrum of time-flipped speech signals for robust spoofing detection. In Proceedings of the Odyssey Speaker Language Recognition Workshop, Tokyo, Japan, 1–5 November 2020; pp. 319–325.
- Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
- He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 770–778.
- 32. Schuster, M.; Paliwai, K.K. Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 1997, 45, 2673–2681. [CrossRef]
- Snyder, D.; Garcia-Romeo, D.; Shell, G.; Povey, D.; Khudanpur, S. Deep neural network embeddings for text-independent speaker verification. In Proceedings of the Interspeech, 2017, Stockholm, Sweden, 20–24 August 2017; pp. 999–1003.
- Snyder, D.; Garcia-Romeo, D.; Shell, G.; Povey, D.; Khudanpur, S. X-vectors: Robust DNN embeddings for speaker recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Calgary, AL, Canada, 15–20 April 2018; pp. 5329–5333.
- Wang, X.; Yamagishi, J.; Todisco, M.; Delgado, H.; Nautsch, A.; Evans, N.; Sahidullah, M.; Vestman, V.; Kinnunen, T.; Lee, K.A.; et al. ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Comput. Speech Lang.* 2020, 64, 101114. [CrossRef]
- Sahidullah, M.; Kinnunen, T.; Hanilci, C. A comparison of features for synthetic speech detection. In Proceedings of the Interspeech, Dresden, Germany, 6–10 September 2015; pp. 1–6.
- Povey, D.; Ghoshal, A.; Boulianne, G.; Burget, L.; Glembek, O.; Goel, N.; Hannemann, M.; Motlicek, P.; Qian, Y.; Schwarz, P.; et al. The Kaldi speech recognition toolkit. In Proceedings of the IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop, Waikoloa, HI, USA, 11–15 December 2011; pp. 1–4.
- Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
- Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4510–4520.
- Ma, N.; Zhang, X.; Zheng, H.-T.; Sun, J. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
- Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. MnasNet: Platform-aware neural architectures search for mobile. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 2820–2828.
- 42. Reddi, S.J.; Kale, S.; Kumar, S. On the convergence of Adam and beyond. arXiv 2019, arXiv:1904.09237.
- 43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. arXiv 2014, arXiv:1412.6980.
- He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
- 45. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, L.; Antiga, L.; Lerer, A. Automatic differentiation in PyTorch. In Proceedings of the NIPS 2017 Workshop Autodiff Submission, Long Beach, CA, USA, 9 December 2017.