HOT-FIXING WAKE WORD RECOGNITION FOR END-TO-END ASR VIA NEURAL MODEL REPROGRAMMING

Pin-Jui Ku[*†], *I-Fan Chen*[†], *Chao-Han Huck Yang, Anirudh Raju, Pranav Dheram, Pegah Ghahremani, Brian King, Jing Liu, Roger Ren, Phani Sankar Nidadavolu*

[*]Georgia Institute of Technology, USA
Amazon, USA

ABSTRACT

This paper proposes two novel variants of neural reprogramming to enhance wake word recognition in streaming end-to-end ASR models without updating model weights. The first, "trigger-frame reprogramming", prepends the input speech feature sequence with the learned trigger-frames of the target wake word to adjust ASR model's hidden states for improved wake word recognition. The second, "predictor-state initialization", trains only the initial state vectors (cell and hidden states) of the LSTMs in the prediction network. When applying to a baseline LibriSpeech Emformer RNN-T model with a 98% wake word verification false rejection rate (FRR) on unseen wake words, the proposed approaches achieve 76% and 97% relative FRR reductions with no increase on false acceptance rate. In-depth characteristic analyses of the proposed approaches are also conducted to provide deeper insights. These approaches offer an effective hot-fixing methods to improve wake word recognition performance in deployed production ASR models without the need for model updates.

***Index Terms*—** Neural model reprogramming, Hot-fixing, End-to-end ASR, Wake word verification

1. INTRODUCTION

Automatic Speech Recognition (ASR) technology, driven by neural end-to-end models [1], has become a vital user interface in ubiquitous voice assistants like Alexa, Cortana, Google Home, and Siri. These systems are invoked by a wake word such as "Alexa", and detection of these words is central to a good user experience. To enhance customer privacy, these assistants often employ wake word verification using the ASR hypothesis to validate detection results from on-device wake word models [2, 3], reducing the risk of false acceptance. However, when an ASR system fails to recognize a wake word which is actually present, it results in false rejection, preventing customers from activating their devices.

When such errors are detected, it is imperative to deploy fixes to the production ASR system promptly to minimize customer impact. These "hot-fixing" techniques have distinct requirements compared to model adaptation or training, such as prior knowledge of the ASR errors, a quick deployment process enabling dynamic activation and deactivation without touching the already deployed ASR model, and ensuring minimal impact on runtime latency.

There are three places in a pretrained end-to-end ASR model where we can apply the hot-fixing techniques: 1) ASR model output, 2) within ASR model, and 3) ASR model input. For hot-fixing the ASR model output, label mapping techniques correct known errors in the output hypotheses, using methods like word or regular expression replacements, machine translation [4], or query rewriting [5]. These approaches are fast, but are limited by the mapping design as it lacks knowledge of the input acoustic information. Hot-fixing within the ASR model involves updating model parameters or adding components for fine-tuning [6–10], such as adapters [11] or LoRA [12]. While effective, direct ASR model updates are more computationally expensive and slower for deployment compared to the other two techniques. Examples for ASR model input-based hot-fixing include techniques such as model reprogramming [13–15], input prompting [16, 17]. The approaches aim to learn input transformations to adapt existing models for specific tasks. While model reprogramming [13, 18] is akin to speech prompting in goal [19], the latter requires a particular model architecture and possesses restricted utility in the context of existing ASR systems employing different architectures.

The model reprogramming mentioned above [13–15] primarily focuses on reprogramming at acoustic embedding levels. The input-sequence-based adaptation is still largely unexplored in the model input-based hot-fixing research for ASR. In this paper, we delve into the potential of input-sequence-based adaptation for streaming ASR and propose two variants of the approach. The first "trigger-frame reprogramming" adds a trainable input-frame-sequence at the beginning of the speech feature sequence to initialize the encoder in the RNN-T ASR model for better recognition of the target word for hot-

[*]The author performed the work during an internship at Amazon.
[†]Equal contribution

fixing. The second "predictor-state initialization" approach is equivalent to learning trigger-frames at token level for prediction network initialization to improve recognition on the target word. We conducted experiments and analyses on wake word verification tasks for streaming ASR systems to showcase the effectiveness of our proposed methods and offer insights into their limitations. While the proposed idea aligns with the concepts of P-tuning [20, 21] and word-level adversarial reprogramming (WARP) [22] in the NLP community, it is worth noting that P-tuning's training process takes place in the embedding space and requires tokenization. In contrast, WARP provides an alternative approach for input-text-based adaptation, but its training process requires a trainable output mapping. In summary, both P-tuning and WARP are designed for NLP tasks and differ from our proposed approach, where we aim to explore leveraging trainable parameters outside the ASR model for adaptation.

2. PROPOSED APPROACH

2.1. Problem Formulation

ASR is a sequence-mapping process converting an input speech feature vector sequence $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where N is the number of input feature frames, to a word label sequence $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_i, ..., \boldsymbol{y}_U\}$, where U is the number of words in the ASR hypothesis. In the context of streaming ASR, the transformation occurs in an autoregressive fashion, i.e. the prediction of output label $\boldsymbol{y}_i$ depends on the input speech features $\boldsymbol{X}_t$ and the output labels $\boldsymbol{Y}_{i-1}$ observed so far as shown in Eq. 1.

$$\boldsymbol{y}_i = f(\boldsymbol{X}_t, \boldsymbol{Y}_{i-1}) \tag{1}$$

The ASR model $f(.)$ may sometimes predict some of the output word labels, $\boldsymbol{y}_i$, incorrectly. For the incorrect output labels, in this paper, we annotated them with a superscript ^{e}, e.g.,

$$\boldsymbol{y}_i^e = f(\boldsymbol{X}_t, \boldsymbol{Y}_{i-1}). \tag{2}$$

The goal of our neural model reprogramming is to find a reprogramming function $g(.)$, where $\tilde{\boldsymbol{X}}_t, \tilde{\boldsymbol{Y}}_{i-1} = g(\boldsymbol{X}_t, \boldsymbol{Y}_{i-1})$, that fixes consistent recognition errors without updating the ASR model $f(.)$, i.e. Eq. 2 becomes $\boldsymbol{y}_i = f(\tilde{\boldsymbol{X}}_t, \tilde{\boldsymbol{Y}}_{i-1})$. The reprogramming function $g(.)$ can be trained using the standard gradient descent with back-propagation from regular ASR losses, e.g., the RNN-T loss, while freezing the ASR model $f(.)$ during training. In this paper, we specifically address ASR errors occurring at position $i = 1$ for wake word recognition. However, the approach should ideally be applicable to any position i in the utterance.

2.2. Trigger-Frame Reprogramming

The trigger-frame reprogramming is realized by choosing the transformation function $g(.)$ in the following form:

$$g(\boldsymbol{X}_t, \boldsymbol{Y}_{i-1}) = [\boldsymbol{T}; \boldsymbol{X}_t], \boldsymbol{Y}_{i-1} \tag{3}$$

, where $\boldsymbol{T} = (\boldsymbol{t}_1, ..., \boldsymbol{t}_M)$ represents a sequence of M trainable vectors with the same dimension of feature frames $\boldsymbol{x}_i$. M is a tunable hyperparameter and is empirically set to 40 in this paper, where we experimented with different Ms and found that $M = 40$ worked the best on our validation data. Note that the trigger-frames are prepended to the original feature sequence $\boldsymbol{X}_t$ so that they will be fed into the streaming ASR model before the real feature inputs, which provides the opportunity for the trigger-frames to adjust the model hidden states for better wake word recognition.

2.3. Predictor-State Initialization

In addition to applying the transformation at the input speech feature, we may also choose $g(.)$ to transform the output label $\boldsymbol{Y}_{i-1}$, namely, $g(\boldsymbol{X}_t, \boldsymbol{Y}_{i-1}) = \boldsymbol{X}_t, [\boldsymbol{T}; \boldsymbol{Y}_{i-1}]$. The advantage of applying $g(.)$ on $\boldsymbol{Y}_{i-1}$ over $\boldsymbol{X}_t$ is that the output label has less frequent refresh rate, which means the effect of the prepended vector may last longer while new observation vectors come in. Also note that the effect of prepending vectors to label sequence is to some extent equivalent to initializing the hidden states of the prediction network in the RNN-T-based ASR model, which is the implementation used in this paper. In essence, rather than training the trigger-frame at the token level, we directly learn the hidden-state vectors for the prediction network initialization. This approach is referred to as "predictor-state initialization."

3. EXPERIMENTS

3.1. Experimental Setup

3.1.1. Data

Experiments are conducted on Text-to-Speech (TTS) synthesized SLURP [23] utterances with five different wake words: *Alexa*, *Cortana*, *Disney*, *Google*, and *Siri*. We used TTS from the ESPNet toolkit [24] to generate the audio for the SLURP-based sentences with speaker profiles from the LibriTTS dataset [25]. To ensure the correctness of the TTS audio generation, we carefully removed all the SLURP sentences where the ESPNet TTS system has pronunciation errors. This is done by decoding all the TTS generated data with the HuBERT XLarge [26] model from the TorchAudio Wav2Vec2ASRBundle and manually examining all the utterances where ASR hypothesis differs from the original SLURP sentence. We adopt SLURP dataset training, validation, and evaluation partitions and selected approximately half of the original SLURP sentences from each partition for experiments.

For the validation and evaluation data, we selected 1,049 and 1,524 SLURP sentences for data generation. Since SLURP sentences are voice commands without wake words,

Table 1. Example of the original SLURP sentence (w/o WW) and the simulated "WW-*Cortana*" sentence.

w/o WW	Send a request to Martin
WW	*Cortana* send a request to Martin

we simply add the wake word to the beginning of the sentences to create voice commands with wake words. Table 1 shows examples of an original SLURP sentence and its corresponding sentence with the *Cortana* wake word. For each wake word, we create the validation and evaluation sentences for the wake word using all the 1,049 and 1,524 original sentences. For TTS audio, two speaker profiles (one male and one female) were used for the audio data generation for each sentence. There is no speaker profile overlap among train / validation / evaluation partition. The final validation and evaluation data for each wake word are 2,098 and 3,048 utterances. In addition to the TTS data, we also include the LibriSpeech [27] test-clean set in the model evaluation to measure the impact of the adaptation approaches on the model performance on its original task.

For training data, we only select 60 sentences from the SLURP training partition to synthesize the 120 training utterances for each wake word following the same process. The 120 training utterances for each wake word are used for training WW specific hot-fixing systems. We limited the amount to only 120 utterances for the purpose of investigating the effectiveness of proposed approaches with limited training data.

In this paper, data sets for a specific wake word are denoted "WW-<*wake_word_name*>", e.g., "WW-*Alexa*", and simply "WW" when referring to all of them collectively. The data sets based on the original SLURP sentences without wake words are denoted as "w/o WW". For the LibriSpeech test-clean set, we denoted it as "Libri".

3.1.2. Systems

Our baseline model is the LibriSpeech [27] pretrained Emformer RNN-T model [28] from TorchAudio (denoted as system **B1**). The model has 76M parameters including 20 transformer layers in the transcription network and 3 LSTM layers in the prediction network. The model has a high averaged False Rejection Rate (FRR) at 98.1% for the five experimental wake words (see Table 3) as those wake words are under-represented in LibriSpeech training data.

Three hot-fixing approaches are compared: fine-tuning (**B2**), trigger-frame reprogramming (**E1**), and predictor-state initialization (**E2**). For each wake word, we train a dedicated hot-fixed system using each approach, utilizing the synthesized training data for the wake word with RNN-T loss. For the fine-tuning system, the whole ASR model is updated; while for the trigger-frame and the predictor-state initialization systems, the ASR model is frozen and only the trigger-frames and the predictor initialization states are updated in the back-propagation process. We tune the hyper-parameters, e.g., learning rate, for each wake word using the validation data. The final checkpoints are selected based on validation loss and the Word Error Rate (WER) performance on the LibriSpeech dev-clean set. Table 2 show the trainable parameter count for each system inside and outside the ASR model. We evaluate general ASR performance using WER and the WW recognition performance using the FRR, and False Acceptance Rate (FAR).

Table 2. Number of trainable parameter inside and outside the ASR model for each systems.

System	# Trainable Param inside ASR Model	# Trainable Param outside ASR Model
B2: Finetuning	76 M	0
E1: Trigger-Frame	0	3,200
E2: Pred-State-Init	0	3,072

Table 3. The averaged FRR and WER over the five wake words for the four systems in the experiment.

System	FRR (%)	WER (%)		
		WW	w/o WW	Libri
B1: pretrained	98.1	27.0	7.8	4.6
B2: finetuning	0.1	4.0	4.8	4.7
E1: trigger-frame	22.9	11.5	8.9	4.8
E2: pred-state init	2.8	7.0	8.7	4.7

3.2. Experiment Results

Table 3 shows the performance of the four systems. The LibriSpeech pretrained baseline **B1** has very high FRR and WER on the wake word test sets at 98.1% and 27.0%, respectively. Fine-tuning the model (**B2**) with WW specific training data reduces the FRR and WER to 0.1% and 4.0%, respectively. However, this approach requires redeployment of the ASR model. Both **E1** and **E2** reduce the FRR of the baseline **B1** significantly as well with only 3k trainable parameters outside the pretrained ASR model **B1**. The **E2** system has much better FRR and WER on wake word test data, confirming our hypothesis in section 2.3 that neural reprogramming applied at output token level is more effective than reprogramming at input feature level. All the hot-fixing systems have minor WER degradation on Librispeech test-clean because of the checkpoint selection criterion (section 3.1.2). The FAR for all the systems is 0% for the same reason. Note that the WER numbers for both **E1** and **E2** on "w/o WW" and "Libri" in Table 3 are the decoding results with the trigger-frame and predictor-state initialization applied ignoring the fact that we already know there is no wake word presents in the utterance.

In the real wake word verification task, where the on-device wake word result is available, the WER numbers will be much closer to the baseline **B1** as we only apply the hot-fixing when on-device wake word result suggests so.

3.3. Utterance Position Limitation of the Approach

To gain a deeper understanding of whether the proposed approach can be applied beyond the initial utterance position, we conduct an utterance position analysis. This is achieved by adding different number of tokens before the wake word, such as, *"call" Cortana* for position 2, *"tell me" Cortana* for position 3, and so on. We then compare the miss recognition rate for the wake word at the each position. Table 4 shows the text templates we used for each utterance position analysis. Note that token sequences we add before the wake word are popular phrases in LibriSpeech. This is designed to ensure the pretrained LibriSpeech ASR model can correctly recognize them so our miss recognition rate for each wake word purely reflecting the error on the wake word. Figure 1 shows the result, where position 1 is the wake word result. Both trigger-frame (**E1**) and predictor-state initialization (**E2**) show improvement over the baseline (**B1**) in all positions; however the magnitude of improvement reduces as the target word moves further away from the beginning of the utterance. This reduction is more significant for the trigger-frame approach possibly because it operates at the acoustic feature level, and the effect is diminished by the new incoming feature frames. We plan to address this utterance position limitation in our future investigation.

Table 4. Text templates used for each utterance position analysis.

Position	Template
1	<wake_word_name> ...
2	Call <wake_word_name> ...
3	Tell me <wake_word_name> ...
4	How are you <wake_word_name> ...
5	Do me a favor <wake_word_name> ...

3.4. Understand how trigger-frames impact ASR output

Figure 2 provides visualizations of the learned trigger-frames for hot-fixing the "Alexa" wake word at three stages during the training process, along with their impact on the output probabilities of the pretrained ASR model. Comparing the input features and the token probability curves confirms that the trigger-frame does not just replicate the wake word feature sequence. Instead, it learns to utilize the frame content to adjust the pretrained ASR model hypothesis, enabling it to improve recognition of the target wake word.

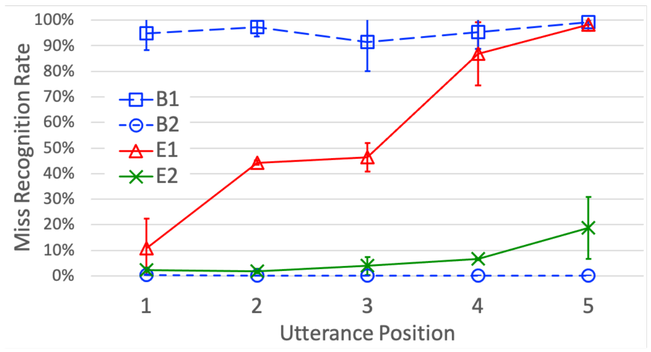

Fig. 1. The miss recognition rate for words at different utterance position from the beginning of the sentence.

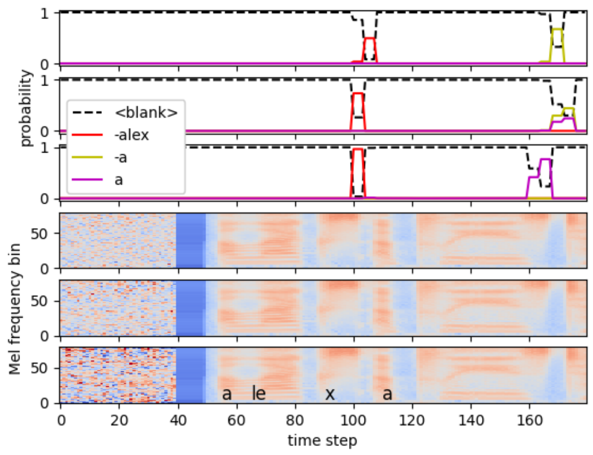

Fig. 2. Visualization of the trigger-frames for "Alexa" and the token probabilities for an "Alexa" wake word utterance in the trigger-frame training process at step 0 (top), 200 (middle), and 1,000 (bottom). "-" in token indicates word boundary.

4. CONCLUSION

In this paper, we introduce two innovative neural reprogramming methods, namely, "trigger-frame reprogramming" and "predictor-state initialization," designed for streaming ASR hot-fixing. Our experimental results on wake word verification tasks highlight the effectiveness of both approaches, showcasing a substantial reduction in FRR that is comparable with whole-model fine-tuning, while utilizing a mere 3k trainable parameters outside the ASR model. In-depth analysis shows predictor-state initialization out-performs trigger-frame reprogramming in FRR at all utterance positions, despite both constrained by utterance position limitations. Addressing these limitations is our future research focus.

5. REFERENCES

[1] Bo Li, Shuo-yiin Chang, et al., "Towards fast and accurate streaming end-to-end ASR," in *Proc. ICASSP*, 2020.

[2] Rajath Kumar, Mike Rodehorst, Joe Wang, Jiacheng Gu, and Brian Kulis, "Building a robust word-level wakeword verification network," in *Proc. Interspeech*, 2020.

[3] "Implement Wake Word Verification — Alexa Voice Service," `https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/avs-ww-overview.html`.

[4] Anirudh Mani et al., "ASR error correction and domain adaptation using machine translation," in *Proc. ICASSP*, 2020.

[5] Xing Fan et al., "Search based self-learning query rewrite system in conversational AI," in *KDD Workshop on Data-Efficient Machine Learning*, 2021.

[6] Ding Zhao, Tara N Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang, "Shallow-fusion end-to-end contextual biasing," in *Proc. Interspeech*, 2019.

[7] Mahaveer Jain, Gil Keren, Jay Mahadeokar, and Yatharth Saraf, "Contextual RNN-T for open domain ASR," in *Proc. Interspeech*, 2020.

[8] Rongqing Huang, Ossama Abdel-Hamid, Xinwei Li, and Gunnar Evermann, "Class LM and word mapping for contextual biasing in end-to-end ASR," in *Proc. Interspeech*, 2020.

[9] Feng-Ju Chang, Jing Liu, Martin Radfar, Athanasios Mouchtaris, Maurizio Omologo, Ariya Rastrow, and Siegfried Kunzmann, "Context-aware transformer transducer for speech recognition," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021.

[10] Kanthashree Mysore Sathyendra, Thejaswi Muniyappa, Feng-Ju Chang, Jing Liu, Jinru Su, Grant P Strimel, Athanasios Mouchtaris, and Siegfried Kunzmann, "Contextual adapters for personalized speech recognition in neural transducers," in *Proc. ICASSP*, 2022.

[11] Sungjun Han, Deepak Baby, and Valentin Mendelev, "Residual adapters for targeted updates in RNN-transducer based speech recognition system," in *IEEE Spoken Language Technology Workshop (SLT)*, 2023.

[12] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. ICLR*, 2022.

[13] Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen, "Voice2series: Reprogramming acoustic models for time series classification," in *Proc. ICML*, 2021.

[14] Chao-Han Huck Yang, Bo Li, Yu Zhang, Nanxin Chen, Rohit Prabhavalkar, Tara N Sainath, et al., "From English to more languages: Parameter-efficient model reprogramming for cross-lingual speech recognition," *in Proc. of ICASSP*, 2023.

[15] Yun-Ning Hung, Chao-Han Huck Yang, Pin-Yu Chen, and Alexander Lerch, "Low-resource music genre classification with cross-modal neural model reprogramming," in *Proc. ICASSP*, 2023.

[16] Kai-Wei Chang, Wei-Cheng Tseng, Shang-Wen Li, and Hung yi Lee, "An exploration of prompt tuning on generative spoken language model for speech processing tasks," in *Proc. Interspeech*, 2022.

[17] Heting Gao, Junrui Ni, Kaizhi Qian, Yang Zhang, Shiyu Chang, and Mark Hasegawa-Johnson, "WAVPROMPT: Towards few-shot spoken language understanding with frozen language models," in *Proc. Interspeech*, 2022.

[18] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein, "Adversarial reprogramming of neural networks," in *Proc. ICLR*, 2019.

[19] Kai-Wei Chang, Yu-Kai Wang, Hua Shen, Iu-thing Kang, Wei-Cheng Tseng, Shang-Wen Li, and Hung-yi Lee, "Speechprompt v2: Prompt tuning for speech classification tasks," *arXiv preprint arXiv:2303.00733*, 2023.

[20] Brian Lester, Rami Al-Rfou, and Noah Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. of EMNLP*, 2021, pp. 3045–3059.

[21] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang, "P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks," in *Proc. of ACL*, 2022, pp. 61–68.

[22] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May, "Warp: Word-level adversarial reprogramming," *Proc. of ACL*, 2021.

[23] Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser, "SLURP: A spoken language understanding resource package," in *Proc. EMNLP*, 2020.

[24] Tomoki Hayashi, Ryuichi Yamamoto, Katsuki Inoue, Takenori Yoshimura, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Yu Zhang, and Xu Tan, "ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit," in *Proc. ICASSP*, 2020.

[25] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, et al., "LibriTTS: A corpus derived from LibriSpeech for text-to-speech," in *Proc. Interspeech*, 2019.

[26] Wei-Ning Hsu et al., "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[27] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[28] Yangyang Shi, Yongqiang Wang, et al., "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6783–6787.