Interspeech 2022
18-22 September 2022, Incheon, Korea

Preventing sensitive-word recognition using self-supervised learning to preserve user-privacy for automatic speech recognition

Yuchen Liu, Apu Kapadia, Donald S. Williamson

Department of Computer Science, Indiana University, USA
liu477@iu.edu, kapadia@indiana.edu, williads@indiana.edu

Abstract

Smart voice assistants that rely on automatic speech recognition (ASR) are widely used by people for multiple reasons. These devices, however, feature "always on" microphones that enable sensitive and private user information to be maliciously or inadvertently collected. In this paper, we develop an end-to-end approach that generates utterance-specific perturbations that obscure a set of words that have been deemed sensitive. In particular, spoken digits, which may be contained in credit card or social security numbers, have been chosen as the words that an ASR system should not be able to recognize, though all other words should be recognized accordingly. Our approach consists of a self-supervised learning feature extractor and U-Net style network for generating noise perturbations. The proposed approach shows promising performance that will help address privacy concerns, without affecting the main functionality of an ASR model.

Index Terms: audio privacy, automatic speech recognition, deep neural networks, self-supervised learning

1. Introduction

In 2019, more than 3 billion smart voice assistants, such as the Amazon Echo, Google Home, and Apple Homepod, were in use.[1] This number is anticipated to rise to 8 billion by 2023. These intelligent devices contain voice assistants, such as Amazon Alexa, Google Now and Apple Siri, which use automatic speech recognition (ASR) to execute spoken commands. These devices offer many conveniences, including controlling other smart devices, performing online shopping and sharing information with family and friends. Unfortunately, the conveniences also introduce privacy and security concerns [1].

The 'always on' mode of the smart devices indicates that the devices are constantly receiving voice information from the target user, and those in the vicinity of the device. This functionality raises privacy concerns for users where false positives can cause unauthorized conversations to be uploaded to the cloud [1] or malicious attackers can gain access to private conversations [2]. Furthermore, the newly developed 'conversation' mode may cause conversations from unintended bystanders to be erroneously collected, stored and processed. This threat is exacerbated by the fact that most ASR systems are designed with deep neural networks (DNNs) [3, 4, 5], which have been shown to be vulnerable to adversarial attacks [6, 7]. Unlike cameras, microphones are not obscured by covering, so other "tangible" defense measures are needed to preserve privacy [8].

Different approaches have been developed to preserve speech privacy. Carlini and Wagner [9] apply an iterative Fast Gradient Sign Method (FGSM) [10] that uses an iterative optimization approach to find a perturbation that causes the ASR system to output a desired transcription that differs from the true one in the audio signal. The approach by Qin *et al.* [11] generates a human-imperceptible perturbation that can be played over loudspeakers in real environments. Xu *et al.* propose a high-performance adaptive security enhancement solution called HASP [12], which generates adversarial noise that maximizes the word error rate (WER). These approaches, in general, find a desired perturbation by solving a complex iterative optimization problem, which updates the input signal directly using the gradient from the back-propagation process. This technique does have a significant disadvantage in that the inference step is computationally expensive. Alternatively, Chen *et al.* [13] create a wearable device that generates ultrasonic jamming signals. Although this approach is effective, unfortunately it requires highly-specialized equipment, so it is not yet a feasible approach for most users.

Many approaches have been developed to address the high computational costs of optimization-based approaches. Poursaeed *et al.* proposed a generative adversarial perturbation framework applied in the image domain [14]. The technique uses both U-Net and ResNet based generators to learn the perturbation from the input image in a computationally efficient manner. Likewise, Xiao *et al.* use a generative adversarial network (GAN)-based approach with a generator to produce a desired perturbation and a discriminator to classify the real and fake samples [15]. In the audio domain, Wang *et al.* [16] use a 1D convolutional based U-Net approach to generate adversarial noise and a fully convolutional discriminator to limit the perturbation's amplitude. The same authors also propose an adversarial generation network [17] for keyword spotting, where it uses a conditional GAN based approach. These approaches have improved performance and computational efficiency, however, they all operate at the single word level and have not been designed for more natural speech where sensitive words are contained within longer utterances, such as phrases or sentences.

In this paper, we develop an approach that generates adversarial noise for a given speech waveform. Unlike prior approaches, our goal is not to render the ASR model completely ineffective, where it cannot recognize any spoken words. Rather our goal is to enable selective word recognition, where words that have been deemed sensitive are not recognized by the ASR model. All other words (e.g., non-sensitive words) should still be recognized correctly, which enables continual usage of the device. This is accomplished by additively injecting the generated noise perturbation into the speech waveform. This is a simulated white-box approach that requires access to an ASR system and where the noise is injected digitally in software. This work serves as a proof-of-concept, but future efforts with focus on real-world implementations. The resulting transcription should contain all the non-sensitive words, but be devoid of all possible sensitive words. We employ an encoder-decoder

[1] https://voicebot.ai/2019/12/31/the-decade-of-voice-assistant-revolution/

10.21437/Interspeech.2022-85

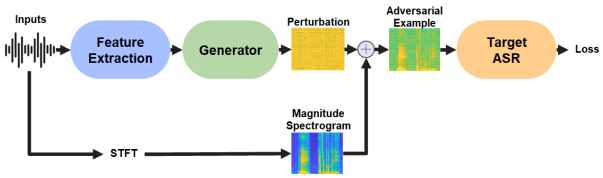

Figure 1: *Model structure for proposed speech filter.*

architecture that is given features that have been extracted from a pre-trained self-supervised learning (SSL) model [18]. We use the self-supervised representations because studies [19, 20] have shown that these representations contain an abundance of higher-level content that are more distinguishable than traditional hand-crafted features, such as the magnitude spectrogram or mel-frequency cepstral coefficients (MFCCs). The SSL approach has also been pre-trained with a large unlabeled speech corpus, which should help with generalization. The features are then fed to a 2D convolutional U-Net to generate a desired perturbation. To the best of our knowledge, this is the first time that SSL representations have been used to prevent sensitive words from being recognized by an ASR model. This work differs from our prior work [21], which used voice-conversion techniques to generate specially-crafted babble noise that sounded like the user. That approach masked all words in an utterance, while our current approach performs selective masking.

2. Method

In this section, we first formulate the problem and provide details about the network architecture. Lastly, the evaluation metrics are introduced.

2.1. Problem formulation

We are first given an audio signal, x, a list of sensitive words, $w = [w_1, w_2....w_n]$, an ASR system, f, and the original transcription, y. The goal of the ASR system is to output the transcription, given the audio signal (e.g., $f(x) = y$). In our case, we want to generate a small perturbation, δ, that when added to the audio signal, results in an adversarial example, $x' = x + \delta$, that when supplied to the ASR system, results in a transcription that recognizes all words, except those that are contained in w. In other words, we want $f(x') = y'$, where $y' = y \backslash \{w\}$. The problem can be formulated as an optimization problem that minimizes the loss between the output of the ASR system, when supplied with the adversarial example, and y'. This is depicted in Eq. (1) below

$$\begin{aligned} &\text{minimize } L(f(x+\delta), y') \\ &\text{such that } y' = f(x) \backslash \{w\} \end{aligned} \tag{1}$$

where $L(\cdot)$ is the loss function that calculates the distance between the desired transcription, y', and the ASR-generated transcription, $f(x + \delta)$.

2.2. Model Structure

The model structure of the proposed approach is shown in Figure 1, which consists of three main components: feature extraction, E, the perturbation generator, G, and the ASR approach f. The time-domain audio signal is first provided to the feature extraction module that extracts high-level features. The extracted high-level features are then fed into the generator, G, which produces a magnitude response for the desired noise perturbation, δ_m. The magnitude spectrum for the adversarial example, x'_m, is then generated from the element wise addition of the magnitude spectrogram of the audio signal and the perturbation. The magnitude spectrogram, x_m, is computed from the audio signal using the short-time Fourier transform (STFT). The perturbed audio signal can be expressed as:

$$x'_m = x_m + G(E(x_m)) \tag{2}$$

The adversarial example is provided as the input to the ASR system, f. esired transcript and the predicted transcript.

2.3. Feature Extraction

We use a pre-trained self-supervised learning (SSL) framework during the feature extraction stage, where we follow the wav2vec approach from [18]. The pre-trained model (e.g. 'wav2vec_large') and the code can be found on FAIRSEQ's official GitHub page.[2] Wav2vec consists of an encoder network and a context network. The encoder network has five convolutional layers and it generates a compressed latent representation. The context network has nine convolutional layers and it combines the latent representations into a contextualized tensor. We use the contextualized tensor as our SSL feature. The model is trained with a noise contrastive binary classification loss, which distinguishes a latent representation from other distractor representations, so that the model can train in a unsupervised manner using a large unlabeled dataset.

Experimental results show that the speech representations obtained using wav2vec perform better than traditional features (e.g., STFT, MFCC,...) on a frame-level phoneme classification task and that they significantly improve ASR performance [18]. It is believed that the extracted features contain more content information that will allow the generator to produce noise perturbations that obscure the recognition of sensitive words, while also not impacting the ability of the ASR model to recognize non-sensitive words. We experimented with fine-tuning the network with our data, but our results were consistent with a recent study [22] that showed that SSL representations can be used directly without fine-tuning on different speech related tasks.

2.4. Generator

Figure 2 shows the architecture of our proposed generator network that produces the noise perturbation. The network consists of a U-Net based encoder-decoder structure that further processes the 2D features extracted by the pre-trained wav2vec model. This framework has proven to be effective in many audio related tasks, including speech enhancement [23], audio source separation [24] and voice conversion [25].

In our implementation, the encoder contains eight 2D-convolutional layers (see the left half of Figure 2). Each 2D-convolution layer is followed by a batch normalization (BN) layer and a leaky ReLU activation function. The output after each convolutional layer is downsampled by a factor of 128. The latent representation from the encoder is then fed into an 8-layer decoder (see the right half of Figure 2). Each layer from the decoder contains a 2D transposed convolution that is followed by batch normalization and ReLU activations. The last decoder layer uses a hyperbolic tangent (Tanh) activation that restricts the magnitude of the perturbation to a low level. A dropout layer is included in the first three decoder layers to prevent overfitting.

[2]https://github.com/pytorch/fairseq/blob/main/examples/wav2vec

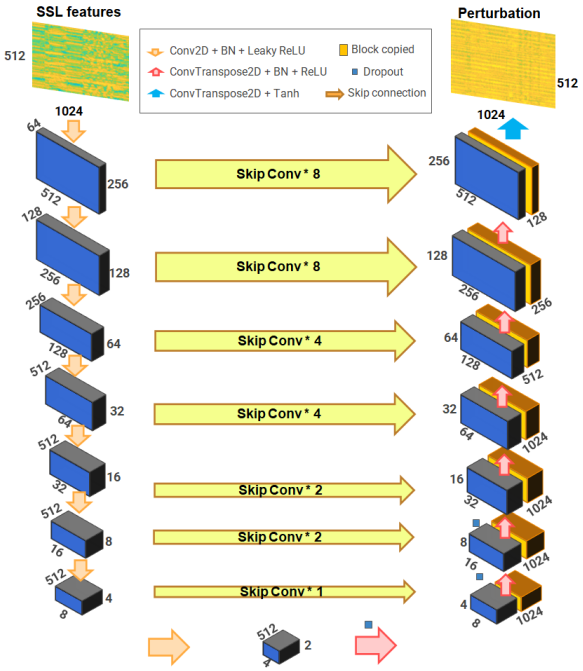

Figure 2: *A depiction of the generator network.*

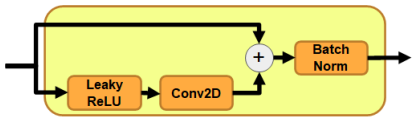

Figure 3: *Skip Convolution Block.*

Unlike the traditional U-Net that directly concatenates the encoder outputs to the decoder output at each level, we use the skip convolution technique that was first proposed in SkipConvNet [26] for dereverberation. The skip convolution block helps to fill the semantic gap between the low-level encoder outputs and the high-level decoder outputs, which increases the learning capability of the generator. The structure for each skip convolution block is shown in Figure 3. More specifically, the encoder outputs are fed into a leaky ReLU layer and then a 2D convolutional layer. The output from the convolutional layer is then added element-wise to the original input and concatenated to the corresponding decoder layer after batch normalization. Fewer skip convolution blocks are used in the lower (or bottom) levels of the encoder since the semantic gap is smaller there [26].

2.5. ASR systems

We choose two different end-to-end ASR models to test the robustness of our proposed approach. DeepSpeech2 [27] is a state-of-the-art recurrent neural network (RNN) based approach that is composed of 3 convolutional layers, 7 recurrent layers and 1 fully connect layer. The second model is the low-rank transformer (LRT) ASR approach [4] that uses a lightweight transformer to significantly reduce inference time. LRT has 4 transformer-encoder layers, which are constructed with 8-head low-rank attention layers and low-rank feed forward layers.

Both ASR models are held frozen during training, but the gradient is used during the backpropagation process to update the weights in the generator. The normalized mag-spectrogram is provided as the input to both models. We use the connectionist temporal classification (CTC) loss [28] to train both ASR approaches [29, 30]. The final loss L is expressed as: $L = \mathbb{E}_x[CTC(f(x+\delta), y')]$.

2.6. Evaluation method

We propose two novel evaluation metrics for this task. The first is the manipulation rate (MR), which is the ratio of the number of correctly removed sensitive words and the total number of sensitive words in an utterance:

$$\text{MR} = \frac{\#\{\text{Removed sensitive words }\}}{\#\{\text{Total sensitive words}\}} \tag{3}$$

The removed sensitive words indicates the number of desired words in the word list, w, that have been correctly misclassified. The manipulation rate shows the success rate of fooling the ASR system and filtering the sensitive words from the transcription. A higher manipulation rate indicates more desired words are filtered. Other words in the utterance, however, may be negatively affected during the perturbation process. Therefore, we propose a second evaluation metric, which we term the preservation rate (PR):

$$\text{PR} = \frac{\#\{\text{Remaining non-sensitive words }\}}{\#\{\text{Total non-sensitive words}\}} \tag{4}$$

The preservation rate indicates how many of the non-sensitive words are correctly recognized, and hence not disturbed by the perturbation. A higher preservation rate is desired, as it indicates that non-sensitive words are correctly recognized by the ASR model.

3. Experiments and Results

We use the Librispeech corpus [31] as the data source for the clean speech data, x. The corpus contains 982 hours of read English speech. All of the signals have a 16 kHz sampling rate. Spoken digits from one to nine are selected as the sensitive words that should not be recognized by the ASR approaches. We use all the signals that are between 3 to 10.2 seconds long that contain digits. All the signals are zero padded to lengths of 10.2 seconds (or 163400 samples) for convenience. The resulting training dataset for DeepSpeech2 contains 7050 files with a total of 8472 spoken digits. The validation set contains 413 files with 476 spoken digits in total, while the testing set has 371 signals with 413 spoken digits in total (see Table 1). DeepSpeech2 is pretrained from the Librispeech corpus and achieves a 9.919% average word error rate (WER) on the Librispeech clean speech testing set. For the LRT ASR system, the training set contains 7934 spoken digits from 6689 files. The validation set contains 392 files with 446 spoken digits. The testing set has 336 signals with 378 spoken digits in total. The LRT ASR model is also pre-trained on the Librispeech corpus and achieves a 14.2% average WER on the corresponding testing data. The detailed digits distribution for each ASR system is shown in Table 1. Note that the ground truth transcription is generated by supplying the noise-free signal to each ASR model, since we wanted to ensure that the ASR model initially recognizes sensitive words. This is why the counts are different for each implementation.

The spectrogram of the speech signal is computed using a 1024-dimensional FFT, a window size of 1024, and a 160-point

	Deep Speech2					LRT				
	Data Distribution			Results		Data Distribution			Results	
	Training	Validation	Test	**Proposed**	Baseline	Training	Validation	Test	**Proposed**	Baseline
one	4071	252	230	216	159	3865	238	211	196	142
two	1659	105	67	64	48	1486	96	62	54	42
three	840	51	44	35	22	832	50	48	36	18
four	488	30	16	13	10	490	31	18	16	11
five	462	13	21	15	8	430	10	15	13	10
six	368	13	13	11	8	316	11	10	8	6
seven	236	4	8	6	5	188	4	7	3	2
eight	193	5	8	6	3	189	3	5	2	2
nine	155	3	6	3	1	138	3	2	1	0
Total	8472	476	413	369	264	7934	446	378	329	233

Table 1: *The data distribution of the spoken digits, along with the per-digit performance of the proposed (SSL features) and baseline (STFT features) approaches for the two ASR models. The results show the number of sensitive words that were not recognized, so higher numbers are better.*

hop size, where this hop size is selected to aid dimension matching. The magnitude spectrum is computed, mean-variance normalized and truncated to match the dimensions of the perturbation that is generated.

The 2-D convolutional layers of our encoder use a kernel size of 3×3 and a stride of 2, while the decoder uses kernel size of 2×2 and a stride of 2. The convolution layers in the skip convolutions block have a kernel size of 3×3 and a stride of 1. The leaky ReLU activation has a 0.2 negative slope. The dropout rate is a 0.3. We use the Nesterov momentum based stochastic gradient descent (SGD) optimizer with a learning rate of 5e-4 to train the generator. A 1e-5 L2 penalty is added to the weight decay to prevent the model from overfitting.

3.1. Results

Fig 4 shows the experimental results for our proposed approach (SSL features) along with a baseline model (STFT features). The baseline model uses the magnitude spectrogram as the input to the generator. This allows us to quantify the impact of the SSL features. The results of our proposed approach are promising according to both ASR models. The manipulation rate of our proposed approach reaches 89.35% and 87.04% for Deepspeech2 and LRT models, respectively, which indicates that the spoken digits are unrecognizable due to the injection of the noise perturbation. For the baseline system, the STFT features result in 63.92% and 61.64% MRs for the respective ASR systems, which indicates that SSL features help produce more distinguishable noise perturbations for sensitive words.

While the approach successfully renders the desired words unnoticeable, we still want non-sensitive words to be correctly recognized. By using SSL features, the preservation rate reaches 71.15% and 71.2%, while the STFT features result in preservation rates of 57.71% and 62.36%, on the respective ASR models. These results show that our proposed end-to-end model performs promisingly well at removing desired words from a given utterance, while keeping other words the same on two differently structured ASR models.

Table 1 also shows the number of sensitive words that were not recognized, as a function of each spoken digit, for the DeepSpeech2 and LRT ASR models. Most of the spoken "one" and "two" digits are not recognized. Other digits, such as "three" have lower manipulation rates than others. In Table 1, we can find that there are less samples for higher digits. However, our proposed model still learns well and successfully manipulates many of these digits.

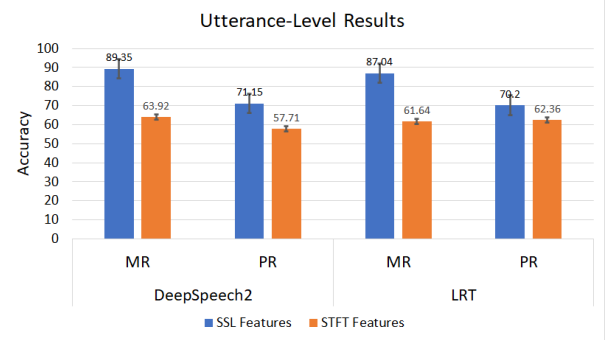

Figure 4: *The speech filter result for utterance level input*

4. Discussion and Conclusion

The proposed approach generates a specific perturbation and then injects it into the signal to prevent a white-box ASR system from recognizing sensitive words. The approach is robust and has promising results (MR $\geq 87\%$ and PR $\geq 70\%$), where the noise does not obscure non-sensitive words. The experiments are limited in certain ways, so below we briefly discuss future research directions.

Over-the-air injection. For now, all the experiments are conducted in simulated environments, where the noise-perturbation is mathematically added to the speech. One future direction is to play the noise over a loudspeaker for injection. It could make the defense mechanism stronger and harder to detect.

Universal adversarial defense. The proposed defense mechanism in this paper aims to inject a specific perturbation digitally to a given signal which brings limited real word feasibility. A universal perturbation may increase the effectiveness of this defense. Universal adversarial defense indicates that a single perturbation can be used on all of the signals, while still filtering all desired words. The inference time can be further decreased to make it a real-time defense.

5. References

[1] N. Malkin, J. Deatrick, A. Tong, P. Wijesekera, S. Egelman, and D. Wagner, "Privacy attitudes of smart speaker users," *Proceedings on Privacy Enhancing Technologies*, no. 4, pp. 250–271, 2019.

[2] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2011, pp. 17–33.

[3] Y. Miao, "Kaldi+ pdnn: building dnn-based asr systems with kaldi and pdnn," *arXiv preprint arXiv:1401.6984*, 2014.

[4] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, and P. Fung, "Lightweight and efficient end-to-end speech recognition using low-rank transformer," in *ICASSP 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6144–6148.

[5] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[7] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine noodles: exploiting the gap between human and machine speech recognition," in *9th USENIX Workshop on Offensive Technologies (WOOT'15)*, 2015.

[8] I. Ahmad, R. Farzan, A. Kapadia, and A. J. Lee, "Tangible privacy: Towards user-centric sensor designs for bystander privacy," *Proceedings of the ACM Journal: Human-Computer Interaction: Computer Supported Cooperative Work and Social Computing (CSCW '20)*, no. CSCW2, pp. 116:1–116:28, Oct. 2020.

[9] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 1–7.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[11] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5231–5240.

[12] Z. Xu, F. Yu, C. Liu, and X. Chen, "Hasp: A high-performance adaptive mobile security enhancement against malicious speech recognition," *arXiv preprint arXiv:1809.01697*, 2018.

[13] Y. Chen, H. Li, S.-Y. Teng, S. Nagels, Z. Li, P. Lopes, B. Y. Zhao, and H. Zheng, "Wearable microphone jamming," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–12.

[14] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4422–4431.

[15] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.

[16] D. Wang, L. Dong, R. Wang, D. Yan, and J. Wang, "Targeted speech adversarial example generation with generative adversarial network," *IEEE Access*, pp. 124 503–124 513, 2020.

[17] D. Wang, L. Dong, R. Wang, and D. Yan, "Fast speech adversarial example generation for keyword spotting system with conditional gan," *Computer Communications*, pp. 145–156, 2021.

[18] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[19] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, pp. 12 449–12 460, 2020.

[20] A. Van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv e-prints*, pp. arXiv–1807, 2018.

[21] Y. Liu, Z. Xiang, E. J. Seong, A. Kapadia, and D. S. Williamson, "Defending against microphone-based attacks with personalized noise." *Proc. Priv. Enhancing Technol.*, no. 2, pp. 130–150, 2021.

[22] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[23] S. Pascual, A. Bonafonte, and J. Serra, "Segan: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[24] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.

[25] D.-Y. Wu, Y.-H. Chen, and H.-Y. Lee, "Vqvc+: One-shot voice conversion by vector quantization and u-net architecture," *arXiv preprint arXiv:2006.04154*, 2020.

[26] V. Kothapally, W. Xia, S. Ghorbani, J. H. Hansen, W. Xue, and J. Huang, "Skipconvnet: Skip convolutional neural network for speech dereverberation using optimally smoothed spectral mapping," *arXiv preprint arXiv:2007.09131*, 2020.

[27] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[28] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[29] J. Cui, C. Weng, G. Wang, J. Wang, P. Wang, C. Yu, D. Su, and D. Yu, "Improving attention-based end-to-end asr systems with sequence-based loss functions," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 353–360.

[30] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[31] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2015, pp. 5206–5210.