SlothSpeech: Denial-of-service Attack Against Speech Recognition Models

Mirazul Haque*[1], Rutvij Shah*[1], Simin Chen[1], Berrak Sisman[1], Cong Liu[2], Wei Yang[1]

[1]University of Texas at Dallas, USA
[2]University of California, Riverside, USA

`mirazul.haque@utdallas.edu, rutvij.shah@utdallas.edu, simin.chen@utdallas.edu,`
`berrak.sisman@utdallas.edu, congl@ucr.edu, wei.yang@utdallas.edu`

Abstract

Deep Learning (DL) models have been popular nowadays to execute different speech-related tasks, including automatic speech recognition (ASR). As ASR is being used in different real-time scenarios, it is important that the ASR model remains efficient against minor perturbations to the input. Hence, evaluating efficiency robustness of the ASR model is the need of the hour. We show that popular ASR models like Speech2Text model and Whisper model have dynamic computation based on different inputs, causing dynamic efficiency. In this work, we propose SlothSpeech, a denial-of-service attack against ASR models, which exploits the dynamic behavior of the model. SlothSpeech uses the probability distribution of the output text tokens to generate perturbations to the audio such that the efficiency of the ASR model is decreased. We find that SlothSpeech-generated inputs can increase the latency up to 40X times the latency induced by benign input.

1. Introduction

Deep Learning (DL) models have been popular for executing different tasks like object recognition, machine translation, sentence classification, etc, with high accuracy. With the increasing popularity, DL models are also being used in speech-related tasks like Automatic Speech Recognition (ASR). ASR is a task in which a given audio signal is transcribed to text. ASR has been significantly instrumental in tasks like caption generation of audio or using virtual speech assistants.

Because of their usage in real-time scenarios, ASR models are needed to have high-efficiency robustness. Efficiency robustness evaluates if minor perturbation to the input would significantly decrease the efficiency of a model or not. However, unlike accuracy robustness [1, 2, 3, 4], efficiency robustness has not been evaluated on ASR models. To evaluate efficiency robustness, we first need to investigate if any computation in the model is dynamic w.r.t input in nature, as it will cause dynamic efficiency based on different inputs.

First, we investigated the architectures of popular ASR models in Huggingface and found that there are mainly two types of decoder used in ASR models. We refer to the first type as static decoder, where the decoder generates a static number of word or character tokens and then removes unessential tokens. Popular ASR models like CTC models [5] use the static decoder. The second type is referred as dynamic decoder, where the number of generated tokens is dynamic based on input. Popular ASR models like Speech2Text [6, 7] and Whisper [8] use dynamic decoder-based mechanisms. As the dynamic decoder shows dynamic efficiency, the efficiency robustness of these models is needed to be evaluated. In this work, we focus on the evaluation of the efficiency robustness of these dynamic decoder-based ASR models.

The efficiency robustness of a model ensures that the efficiency of the model is not impacted significantly by adding perturbation to the input. If a model lacks efficiency robustness, this could lead to fatalities. For example, an ASR system is used in an autonomous vehicle to recognize the driver's instructions. If the model is not efficiency-robust, the model could take a significant amount of time to respond, denying the service and leading to an accident. Hence, the efficiency robustness in a system is needed to be evaluated.

To evaluate the efficiency robustness of ASR models, first, we need to establish to the relation between input and dynamic efficiency. As mentioned earlier, for the dynamic decoder, number of generated tokens is dynamic w.r.t input and the number of generated tokens is dependent on a specific end-of-sentence($< EOS >$) token. If the occurrence of $< EOS >$ is delayed, the number of generated tokens is increased. The occurrence of any token is dependent on output probability distribution of the decoder. Hence, we need to modify the input, which can modify the output probability distribution of the decoder.

Based on the aforementioned relation between input and efficiency, we propose SlothSpeech, a denial-of-service attack on the ASR models. Denial-of-service attack increases the latency or the response time of the model significantly, ultimately denying the model service to the users. SlothSpeech is an iterative white-box attack that uses output probability distribution of different tokens to increase latency.

We evaluate SlothSpeech[1] on three popular models: Speech2Text, Whisper, and Speech2Text2 [6, 7, 8] and three popular datasets: LibriSpeech, OpenSLR and VCTK [9, 10, 11]. We evaluate SlothSpeech on two criteria: effectiveness and quality. Effectiveness measures the effect of SlothSpeech on the latency of the model. While quality measures the distance between adversarial input and benign input. We find that SlothSpeech-generated inputs can increase the latency up to 4000% of the latency induced by benign input.

Our work makes the following contributions:

- **Problem Characterization.** Our work is the first work to characterize the latency surge vulnerability in ASR models.

- **Approach.** We propose a novel loss function to synthesize an iterative white-box attack.

- **Experimentation.** We evaluate SlothSpeech on two criteria with three popular datasets and three popular models.

The rest of the paper is organized as follows: In Section 2, we introduce the the background of dynamic-decoder-based ASR model and adversarial attack. In Section 3, we formulate the optimization problem of the denial-of-service attack on

---

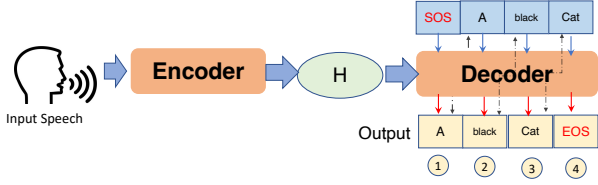[1]*Both authors contributed equally to this research.

Figure 1: *Working mechanism of dynamic-decoder-based ASR*

ASR models and we explain SlothSpeech approach. In Section 4, we discuss the evaluation results.

2. Background

2.1. Automatic Speech Recognition Systems

Given the input speech x, the ASR systems compute the output probability for a sequence of tokens $\Pr(y|x)$ through the Bayes' theorem.

$$\Pr(y|x) = \prod_{u=1} \Pr(y_i)|x, y_1, \cdots, y_{u-1}) \qquad (1)$$

The computation process is shown in Equation 1, where y_i is the ith output token. In this paper, our focus is on ASR systems that are based on the dynamic decoder architecture. Such systems comprise two key components - an encoder and a decoder. As illustrated in Figure 1, the encoder neural network is responsible for encoding the input speech into a hidden representation, while the decoder begins with a special token SOS and generates subsequent output tokens iteratively by leveraging the decoder neural networks. The decoding process will continue until it reaches another token EOS. A notable observation regarding the working mechanism of dynamic-decoder-based ASR systems is that the decoder is invoked more frequently for inputs that arrive at the end of the sequence (EOS). This observation implies that the ASR system tends to allocate more computational resources to inputs that have longer output sequences, thereby prioritizing the decoding of such sequences. Therefore, longer outputs lead to wastage of computational resources on the part of the victim ASR system.

2.2. Adversarial Attacks against DNNs

Recently, several adversarial attacks [1, 2, 3, 4, 12] have been developed for targeting DNN-based systems. These attacks can create human-imperceptible adversarial perturbations, which, when applied to benign inputs, generate adversarial examples that can easily evade even the most advanced DNNs. Based on the availability of the DNNs parameters, the adversarial attacks can be categorized as white-box attacks and black-box attacks. Apart from correctness-based adversarial attacks, researchers have recently proposed denial-of-service attacks [13, 14, 15, 16, 17, 18, 19] for targeting neural networks with dynamic decision routes. The objective of these attacks is to maximize the computational cost of the victim model, thereby decreasing its availability.

3. SlothSpeech

In this section, we discuss the proposed approach SlothSpeech. First, we formulate the problem, then we focus on how we create the objective function, and finally, we explain the iterative optimization approach.

3.1. Problem Formulation

Our objective is to produce audio that is imperceptible to humans and can decrease the efficiency of the victim's ASR model during inference. By reducing the efficiency of the ASR model, the adversary can deplete its computational resources, such as battery consumption, and make it unavailable, ultimately achieving the goal of denial-of-service. Our objective consists of two main factors: (i) reducing the efficiency of the victim's ASR model, and (ii) ensuring imperceptibility to humans. We formulate our objective as an optimization problem, as shown in Equation 2,

$$\Delta = \operatorname{argmin}_\delta \text{Efficiency}_f(x + \delta) \qquad s.t. \|\delta\| \le \epsilon \qquad (2)$$

where x is an audio input fed to ASR system $f(\cdot)$, our aim is to generate an audio perturbation δ that minimizes the efficiency of the ASR system while also satisfying imperceptibility constraints."

3.2. Differentiable Proxy of Latency

As the Efficiency$_f$ is not a differentiable term w.r.t input, we need to find a proxy of Efficiency$_f$, which is differentiable. As discussed in the section 2, the Efficiency$_f$ is dependent on the length of the output text. Also, the length of the output text is dependent on the occurrence of end-of-sentence or $<EOS>$ token. Our objective is to delay the occurrence of the $<EOS>$ token. To achieve this, we will first discuss how a token is selected for the output.

Formally, the ASR model's output is a sequence of probability distributions, $f(x) = [p_1(x), p_2(x), \cdots, p_n(x)]$ and the output token sequences are $[t_1(x), t_2(x), \cdots, t_n(x)]$. Here, $t_i(x) = \operatorname{argmax}(p_i(x))$. Also, the likelihood of the output tokens and the EOS tokens are represented as $[p_1(x)^{t_1}, p_2(x)^{t_2}, \cdots, p_n(x)^{t_n}]$ and $[p_1(x)^{EOS}, p_2(x)^{EOS}, \cdots, p_n(x)^{EOS}]$ respectively.

To delay the occurrence of the EOS tokens, one approach would be to minimize the likelihood of the EOS tokens. But this approach would be resource-consuming because cross-entropy would be used on the large vocabulary. Instead of that, we convert the probability distribution p_i for the multi-classification task to a binary classification task *i.e.,*, is or not EOS token. The new probability distribution is represented as $q_i = [l_i(x)^{EOS}, \sum_j l_i(x)^j - l_i(x)^{EOS}]$, where $j \ne EOS$.

Our first objective is to decrease the likelihood of all EOS tokens in the output; hence, Efficiency$_f$ can be replaced by, $\mathcal{L}_{EOS} = \frac{1}{n} \sum_{i=1}^n l_i(x + \delta)^{EOS}$. However, considering only EOS tokens during the loss function can reduce the effectiveness of the attack. For the last token of the output sequence, EOS token's likelihood would be highest, which we need to decrease. Also, if we increase the likelihood of another specific token (token with second highest likelihood), it would increase the effectiveness of the attack because the specific token's likelihood would increase, and EOS token's likelihood would decrease. Hence the proxy of Efficiency$_f$ will be

$$\mathcal{L}_{EOS} = \left(\frac{1}{n} \sum_{i=1}^n l_i(x + \delta)^{EOS} + q_n(x + \delta) \right)$$

where, q_n represents the probability of the token with second highest likelihood in the last (n^{th}) token. If P is the distance norm, and c is a weight value defined by attacker, then the final optimization loss function can be represented by,

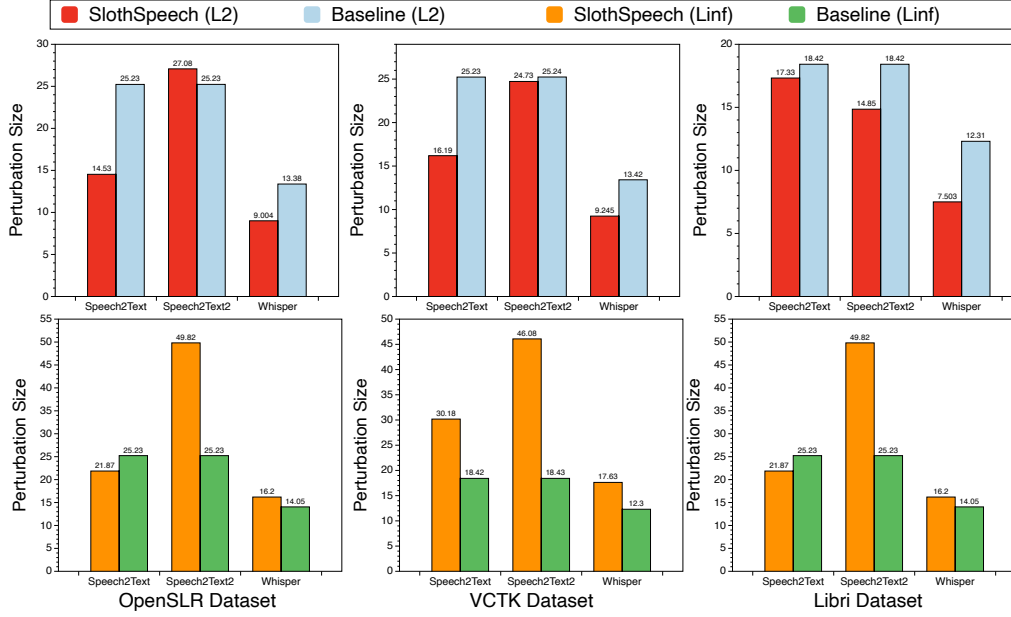$$\mathcal{L} = \|\delta\|_P + c \cdot \mathcal{L}_{EOS} \qquad (3)$$

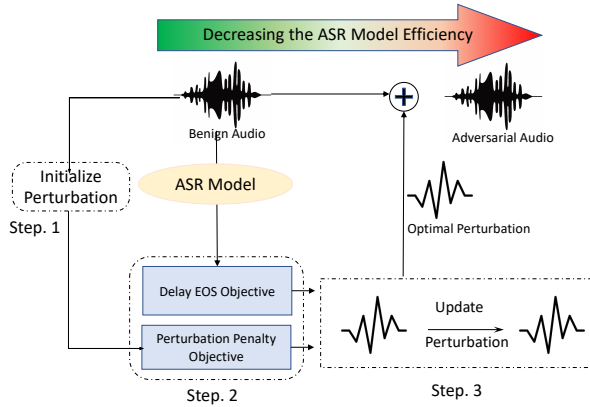Figure 2: *Comparison of Distance between SlothSpeech and Gaussian Noise.*

Figure 3: *Design Overview of SlothSpeech*

3.3. Approach

Algorithm 1 and Figure 3 show the details of SlothSpeech. The SlothSpeech approach can be divided into three parts. (i) *Initialize.* First, we initialize different variables that would be needed for synthesizing perturbation. (ii) *Calculating loss function.* Next, we calculate loss function based on equation 3. (iii) *Update adversarial input.* Based on the optimization of loss function, we update the adversarial input. Below, we explain each step.

Intitialize. First, we initialize δ (Perturbation), x^* (Final Adversarial Input) and max_N (Maximum Number of Tokens), $iter$ (Iteration Index) in Line 2-3.. These values will be updated iteratively based on the optimization procedure.

Calculating loss function. In this step, we calculate the loss function that will be optimized. Based on equation 3, we have two components to the loss function. First loss component, $\mathcal{L}_{EOS}$ is calculated based on the likelihood of different tokens (Line 6-7), whereas the second loss component $\mathcal{L}_d$ (Line 8)

signifies the distance between adversarial and benign examples (*e.g.*, L_2 and L_∞). Then both components are added based on the weight c (Line 9).

Update adversarial input. Finally, we update the adversarial input $hatx$ (Line 10), and check if the output length generated by $\hat{x}$ is better than the saved max_N (Line 11-12). If yes, the final adversarial input x^* and max_N is updated.

Algorithm 1 SlothSpeech

> **Input:** Benign input x, victim ASR $f(\cdot)$, maximum iteration number T, weight val c.
>
> **Output:** Latency-based adversarial example x^*

2: Initialize δ, x^*, max_N
3: $iter \leftarrow 0$
4: **while** $iter < T$ **do**
5:     $\hat{x} = x + \delta$
6:     $H = f(\hat{x})$
7:     $\mathcal{L}_{EOS} \leftarrow GetEOSLoss(H)$
8:     $\mathcal{L}_d \leftarrow Distance(\hat{x}, x)$
9:     $\mathcal{L}_{adv} \leftarrow \mathcal{L}_{EOS} + c \cdot \mathcal{L}_d$
10:     $\hat{x} \leftarrow \hat{x} + \frac{\partial \mathcal{L}_{adv}}{\partial \hat{x}}$
11:     $N \leftarrow \text{GetLength}(\hat{x})$
12:     $max_N, x^* = \text{Update}(max_N, N, \hat{x})$
13: **end while**
14: Return x^*

---

4. Evaluation

We evaluate SlothSpeech based on two criterion: effectiveness and quality.

4.1. Experimental Setup

Datasets and Models. For evaluation, LibriSpeech dataset [9], OpenSLR [10], and VCTK dataset [11] have been used for synthesizing the adversarial examples. We use three popular ASR models: Speech2Text [6], Whisper [8] and Speech2Text2 [5].

Table 1: *Mean and Max latency and number of outut tokens of the models against SlothSpeech and baseline techniques. seed represents the original input, whereas abs and inc represent the absolute value and increase in metric respectively. All the experiments are performed on LibriSpeech, OpenSLR and VCTK datasets and three aforementioned Models.*

Metric	Dataset	Model	Mean Absolute Value (Abs)				Mean % Increase (Inc)			Max Absolute Value (Abs)			
			seed.	Gaussian.	L_2 (Ours)	L_∞ (Ours)	Gaussian.	L_2 (Ours)	L_∞ (Ours)	seed.	Gaussian.	L_2 (Ours)	L_∞ (Ours)
# of Tokens	Libri	Speech2Text	22.20	22.17	**1001.00**	**1001.00**	0	**4415.00**	**4415.00**	85.00	80.00	**1001.00**	**1001.00**
		Whisper	22.95	22.84	**42.82**	42.58	0.00	**86.50**	85.50	98.00	96.00	151.00	**173.00**
		Speech2Text2	24.36	29.46	152.50	**429.61**	20.90	526.00	**1663.00**	56.00	463.00	**1001.00**	**1001.00**
	OpenSLR	Speech2Text	91.05	102.99	**1001.00**	**1001.00**	13.11	**999.00**	**999.00**	**1001.00**	**1001.00**	**1001.00**	**1001.00**
		Whisper	21.65	19.98	36.85	**38.36**	-7.70%	70.20	**77.18**	48.00	58.00	249.00	**366.00**
		Speech2Text2	24.32	12.33	**315.62**	292.53	-49.00	**1197.00**	1102.00	157.00	22.00	**501.00**	**501.00**
	VCTK	Speech2Text	22.35	32.59	**1001.00**	1000.00	46.00	**4378.00**	4374.00	199.00	**1001.00**	**1001.00**	**1001.00**
		Whisper	17.66	17.68	32.03	**32.59**	0.00	81.30	**84.50**	50.00	43.00	102.00	**441.00**
		Speech2Text2	18.05	12.12	**260.50**	206.57	-32.80	**1343.00**	1044.00	54.00	19.00	**501.00**	**501.00**
Latency (ms)	Libri	Speech2Text	218.00	214.53	8722.00	**8746.00**	-1.80	3900.00	**3911.00**	875.00	823.00	9039.00	**9224.00**
		Whisper	376.00	369.00	**637.00**	630.00	-1.80	69.00	67.50	1329.00	1348.00	2068.00	**2293.00**
		Speech2Text2	567.00	1738.00	10722.00	**10725.00**	206.00	1791.00	**1791.00**	2237.00	60493.00	99793.00	70287.00
	OpenSLR	Speech2Text	932.00	905.00	8767.00	**8778.00**	-2.80	840.60	**842.00**	8778.00	8660.00	9036.00	**9038.00**
		Whisper	376.00	352.00	600.00	**618.00**	-6.30	59.50	**64.30**	759.00	902.00	3587.00	**5640.00**
		Speech2Text2	893.00	558.00	16101.00	13565.00	-37.40	1703.00	1428.00	8996.00	944.00	32190.00	36124.00
	VCTK	Speech2Text	242.00	364.00	**10091.00**	9960.00	50.40	**4069.00**	4015.00	2003.00	10219.00	**10996.00**	10835.00
		Whisper	308.00	304.00	499.00	**506.00**	-1.20	62.00	**64.20**	764.00	681.00	1430.00	**6006.00**
		Speech2Text2	720.00	502.00	**12704.00**	9604.00	-30.77	**1670.00**	1233.00	11269.00	906.00	**35377.00**	29965.00

All the pre-trained weights are gathered from Huggingface.

Baseline and Metric As this is the first denial-of-service attack on ASR models, we use Gaussian noise as baseline. We examine two metrics to reflect the ASR models' computational costs in order to measure the effectiveness of SlothSpeech in increasing the victim ASR models' computational costs. The first is the number of tokens generated by the decoder (hardware-independent metric), and the second is the latency of ASR models (hardware-dependent metric) in handling an input. We first measure the absolute computational costs (Abs.) of the benign inputs and then generate adversarial examples. After we compute the computational cost increments (Inc.). For quality evaluation, we use the distance between adversarial and benign input as metric, while for transferability evaluation, we measure percentage increase in latency as metric.

Implementation Details. For this work, we use c value as 1 and T value as 100 to generate adversarial inputs. We set max_length of the tokens as 1001, and as distant norm, we use L_2 and L_∞ norms. Only for Speech2Text2 model tested with OpenSLR and VCTK datasets, we use 500 as max_length, because extending that limit caused significant load on the GPU.

4.2. Effectiveness

As mentioned earlier, we measure the effectiveness of Sloth-Speech by measuring computational latency and number of tokens in the output. Table 1 shows our results of the effectiveness of SlothSpeech and the Gaussian noise. We show both the mean absolute values of the metric in the table and the mean percentage increase in the metric due to perturbation. (seed being the original input). Also, we show maximum absolute values for number of tokens and latency achieved by seed input and different technique-generated inputs.

It can be noticed that all three models are vulnerable against SlothSpeech and, SlothSpeech-generated inputs perform significantly better than baseline w.r.t increasing computation in ASR models For Speech2Text model, all the adversarial examples induce maximum length of the output token. The results reflect that Speech2Text model is the least efficiency-robust against SlothSpeech. Speech2Text2 model also shows low robustness against SlothSpeech, however the percentage increase in latency for Speech2Text is higher than Speech2Text2. Whisper model has shown higher robustness against SlothSpeech than the other

two models. Inputs generated using both distant norms have similar effectiveness, however, for S2T-Libri pair, inputs generated through L_∞ norm have significantly higher effectiveness than L_2 norm.

Summary. SlothSpeech generated input outperforms the baseline by significantly increasing the latency of the ASR models (up to 4000%).

4.3. Quality

We evaluate quality of adversarial examples w.r.t magnitude of the perturbation added to the audio signal due to SlothSpeech and baseline. Figure 2 shows the results. We use bargraph to show the mean of different perturbations added to the input by SlothSpeech and baseline. It can be observed that mean value of L2 perturbation for baseline and SlothSpeech is similar for all case scenarios, however, for L_∞ norm, SlothSpeech perturbation is slightly higher than the baseline. Although except S2T2 model, the mean perturbation of SlothSpeech is always similar to or lower than baseline for both norm. Hence, it can be noted that with similar mean perturbation, SlothSpeech has a significantly higher effectiveness than baseline.

Summary. The mean perturbation magnitude added to SlothSpeech generated input is similar to the magnitude of the Gaussian noise.

5. Conclusion

In this work, we propose SlothSpeech[2], a white-box denial-of-service attack that can decrease the efficiency of the ASR models significantly. SlothSpeech uses the likelihood of output tokens to generate adversarial inputs. We evaluate SlothSpeech on three popular datasets and three popular models. We find that SlothSpeech-generated inputs can increase the model latency up to 40 times more than the benign input. However, in this work, we do not focus on improving the efficiency robustness of the ASR models.

___
[2]https://github.com/0xrutvij/SlothSpeech

6. References

[1] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[2] A. Bojchevski and S. Günnemann, "Adversarial Attacks on Node Embeddings via Graph Poisoning," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 695–704.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial Attacks on Neural Networks for Graph Data," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 2847–2856.

[5] T. Zenkel, R. Sanabria, F. Metze, J. Niehues, M. Sperber, S. Stüker, and A. Waibel, "Comparison of decoding strategies for ctc acoustic models," *arXiv preprint arXiv:1708.04469*, 2017.

[6] C. Wang, Y. Tang, X. Ma, A. Wu, S. Popuri, D. Okhonko, and J. Pino, "fairseq s2t: Fast speech-to-text modeling with fairseq," *arXiv preprint arXiv:2010.05171*, 2020.

[7] C. Wang, A. Wu, J. M. Pino, A. Baevski, M. Auli, and A. Conneau, "Large-scale self- and semi-supervised learning for speech translation," *CoRR*, vol. abs/2104.06678, 2021. [Online]. Available: https://arxiv.org/abs/2104.06678

[8] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[9] 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)*. IEEE, 2015, pp. 5206–5210.

[10] M. Korvas, O. Plátek, O. Dušek, L. Žilka, and F. Jurčíček, "Free English and Czech telephone speech corpus shared under the CC-BY-SA 3.0 license," in *Proceedings of the Eigth International Conference on Language Resources and Evaluation (LREC 2014)*, 2014, p. To Appear.

[11] C. Veaux, J. Yamagishi, K. MacDonald *et al.*, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit," *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2017.

[12] M. Haque, C. J. Budnik, and W. Yang, "Corrgan: Input transformation technique against natural corruptions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 194–197.

[13] M. Haque, A. Chauhan, C. Liu, and W. Yang, "ILFO: Adversarial Attack on Adaptive Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 264–14 273.

[14] M. Haque, Y. Yadlapalli, W. Yang, and C. Liu, "Ereba: Black-box energy testing of adaptive neural networks," *arXiv preprint arXiv:2202.06084*, 2022.

[15] S. Chen, M. Haque, C. Liu, and W. Yang, "Deepperform: An efficient approach for performance testing of resource-constrained neural networks," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–13.

[16] S. Chen, Z. Song, M. Haque, C. Liu, and W. Yang, "Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 365–15 374.

[17] S. Chen, C. Liu, M. Haque, Z. Song, and W. Yang, "Nmtsloth: understanding and testing efficiency degradation of neural machine translation systems," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 1148–1160.

[18] M. Haque, S. Chen, W. A. Haque, C. Liu, and W. Yang, "Nodeattack: Adversarial attack on the energy consumption of neural odes," 2021.

[19] S. Chen, H. Chen, M. Haque, C. Liu, and W. Yang, "The dark side of dynamic routing neural networks: Towards efficiency backdoor injection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 24 585–24 594.