TinyML has a Security Problem - An Adversarial Perturbation Perspective

Swapnil Sayan Saha (605353215), Khushbu Pahwa (805498854), Basheer Ammar (705471708)

Motivation: Recent advancements in machine learning have opened a new opportunity to bring intelligence to low-end Internet-of-Things nodes for making complex and time-critical inferences from unstructured data. Dubbed TinyML, the advancements include model compression, lightweight machine-learning blocks, AutoML frameworks, and software suites designed to perform ultra-low-power, always-on, and onboard sensor data analytics on resource-constrained platforms [1].

Problem: The first-generation TinyML workflow does not include attack surface analysis and tools to defend the inference pipeline against attacks at various layers in the cyber-physical system. Moreover, the security cost of using "lightweight by design" models on embedded hardware against adversarial attacks has not been explored [1].

Contributions: Our contributions are two-fold:

- We perform two well-known adversarial attacks on 10 state-of-the-art TinyML models and 13 state-of-the-art large neural networks across three applications. We show that TinyML models are less robust to adversarial perturbations than large models.
- We propose an efficient neural architecture search (NAS) framework to yield models that have both high utility and adversarial accuracy within the target platform bounds while adding negligible search and training costs in the TinyML workflow.

While solutions for certifiable adversarial robustness are well studied in the machine learning community [2-7], our focus is on providing a starting point to make the recent advancements in TinyML more backward compatible and functionally equivalent (high fidelity) to upstream models without adding significant training and compute overhead.

Related Work:

- Prior work has shown that compressed models are prone to adversarial attacks and false data injection with a higher success rate than larger models [8-12]. However, these works focus only on derivative models and not models lightweight by design.
- Current methods of automatic exploration of adversarially robust models using NAS include limiting the network Lipschitz constant [8] and performing additional adversarial training steps [13]. The former is challenging to quantify for complex and non-neural models, while the latter adds significant search and training costs.

Attack Model: The TinyML model is assumed to run on an embedded platform, making inferences in real-time from observed sensor data. The goal of the adversary is to perturb the observed sensor data passively and covertly to cause the model to provide erroneous outputs without alerting the client. For image recognition, audio keyword spotting, and wearable human activity recognition, such undetectable perturbations can be injected via adversarial patches or pixel patterns [14-15], inaudible ultrasonic injection or laser injection or background audio [16-18], and spoofing or acoustic attacks [19]. The adversary neither knows anything about the model architectural encodings or parameter values (white-box) nor can query the model (black-box). The adversary knows about the input representation and the output. Regardless of an unprivileged adversary, adversarial perturbations have been shown to be universal [20] and transferable across domains [21-22]. While side-channel attacks are widespread in the embedded domain [23], we do not consider these attacks in this project.

Results of Adversarial Perturbations: To quantitatively highlight the utility-adversarial accuracy gap between large models and TinyML models, we perform two white-box adversarial attacks:

- Fast Gradient Sign Method (FGSM) [24]: Maximizes the value of a loss function J of a model f by producing an adversarial input x_p based on the sign of the gradients of that loss function with respect to the original input x and label y. ϵ controls the perturbation strength.
$$x_p = x + \epsilon \cdot \mathrm{sign}(\nabla_x J(f, x, y))$$
- Projected Gradient Descent (PGD) [25]: Iterative or multi-step variant of FGSM. ϵ and α control the perturbation strength.
$$x_p^{t+1} = \mathrm{clip}_\epsilon(x^t + \alpha \cdot \mathrm{sign}(\nabla_x J(f, x^t, y)))$$

We use adversarial accuracy as a metric to measure adversarial robustness:

$$\frac{1}{N}\sum_{i=0}^{N} q_i, \quad q_i = \begin{cases} 1, & \text{if} \quad y_{\text{pred}}^{x_i} = y_{\text{pred}}^{x_{i,p}} \\ 0, & \text{otherwise} \end{cases}$$

Table 1 highlights the properties of the chosen state-of-the-art large and TinyML models across three multiclass classification applications, namely image recognition (CIFAR-10 dataset [26]), audio keyword spotting (Google

Speech Commands dataset [27]), and wearable human activity detection (AURITUS dataset [28]). The first two applications are present in the MLPerf Tiny v0.5 Inference Benchmark [29].

Table 1: Characteristics of candidate models for adversarial robustness test.

Application	Model Type	Model	Test Accuracy (%)	Parameters (M)	Features	NAS@	Transfer Learning
Image Recognition	Large	EfficientNetB0 [30]	93.2	4.07	✗	✓	✓*
		EfficientNetB4 [30]	93.5	17.70	✗	✓	✓*
		EfficientNetv2B0 [31]	96.7	5.93	✗	✓	✓*
		EfficientNetv2B3 [31]	97.0	12.95	✗	✓	✓*
		ResNet50 [32]	89.7	23.62	✗	✗	✓*
		VGG19 [33]	92.3	20.03	✗	✗	✓*
	TinyML	ResNet8 [29]	87.1	0.079	✗	✗	✗
		MCUNet DS-CNN (320-1)` [34]	87.7	0.57	✗	✓	✓*
		MCUNet DS-CNN (256-1)` [34]	87.5	0.56	✗	✓	✓*
Audio Keyword Spotting	Large	Attention RNN [35]	93.9	1.29	✓^	✗	✗
		CNN [36]	82.4	0.095	✓^	✗	✗
		GRU [36]	92.2	0.50	✓^	✗	✗
		LSTM [36]	92.9	1.03	✓^	✗	✗
	TinyML	DS-CNN [29]	92.2	0.025	✓^	✗	✗
		TCN [37][38]	76.0	0.019	✓^	✗	✗
Human Activity Recognition	Large	CNN-LSTM [39]	99.7	1.74	✗	✗	✗
		CNN [39]	99.0	3.03	✗	✗	✗
		LSTM [39]	97.3	0.20	✗	✗	✗
	TinyML	Bonsai [40]	72.6	0.00063	✓	✗	✗
		ProtoNN [41]	72.0	0.00062	✓	✗	✗
		TCN [37][38]	93.0	0.00106	✗	✗	✗
		FastRNN [42]	95.0	0.00015	✗	✗	✗
		FastGRNN [42]	98.6	0.00030	✗	✗	✗

* Transfer learning from ImageNet-1000 to CIFAR-10; ^ Operates on log-Mel spectrograms; @ Optimized via AutoML frameworks; ` Models targeted towards two different microcontrollers.

Figure 1 shows the effects of increased perturbation strength on adversarial accuracy on all models. From Figure 1(g) and 1(h), we can see that overall, TinyML models are 4.4% - 11.4% and 1.0% - 10.9% less robust in terms of adversarial accuracy over large models for FGSM and PGD attacks, respectively. In addition, we also observe:

- TinyML Models which operate on raw sensor data are more likely to be less robust to input perturbations than TinyML models operating on features.
- Models generated by NAS are more sensitive to small perturbations compared to human-engineered models (also observed in [13][43]). This is probably due to high loss smoothness and low gradient variance in the loss contour of NAS-generated models [43].
- TinyML models that have used transfer learning to operate in the new domain are less robust to adversarial perturbations for small changes in the new domain over TinyML models trained from scratch in that domain.
- While there is no significant correlation between model size and adversarial accuracy across different applications, for the same application, on average, TinyML models are less robust than large models.
- Models with high utility (test accuracy) are more likely to be sensitive to small adversarial perturbations over models with low utility, probably due to overfitting.

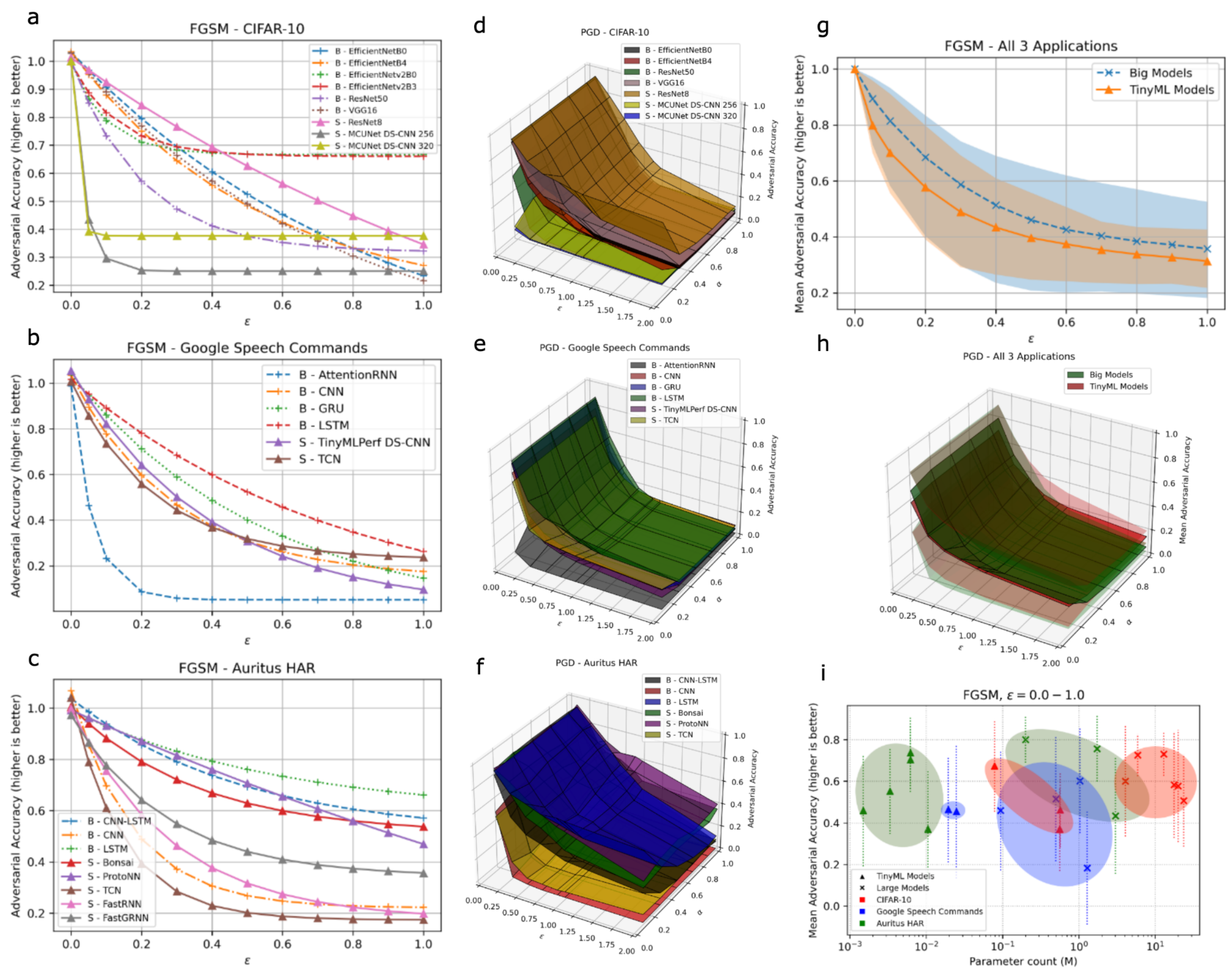

Figure 1: Summary of results of adversarial perturbation. (a-f) show the effect of increasing adversarial perturbation strength on the adversarial accuracy of the candidate models for FGSM and PGD attacks. B represents big models, while S represents TinyML models. (g) and (h) shows the global effect on big and TinyML models of increasing adversarial perturbation. (i) shows the effect of parameter count on mean adversarial accuracy for all models.

Results of Robust and Efficient NAS: For automatic generation of TinyML models robust to adversarial perturbations, we propose a model and data-agnostic NAS with an adversarial accuracy cost parameter, which has negligible additional training and search time. We developed a gradient-free, platform-in-the-loop, and Bayesian NAS framework that acts as a black-box optimizer, based on state-of-the-art Bayesian optimizer ARM Mango [44]. The search space Ω consists of neural network weights w, hyperparameters θ, network structure denoted as a directed acyclic graph (DAG) g with edges E and vertices V representing activation maps and common ML operations v (e.g., convolution, batch normalization, pooling, etc.) respectively, which act on V. The goal is to find a neural network that maximizes the hardware SRAM and flash usage within the device capabilities while minimizing latency and validation RMSE, and maximizing adversarial accuracy on the validation set.

$$f_{\text{opt}} = \lambda_1 f_{\text{error}}(\Omega) + \lambda_2 f_{\text{flash}}(\Omega) + \lambda_3 f_{\text{SRAM}}(\Omega) + \lambda_4 f_{\text{latency}}(\Omega) + \lambda_5 f_{\text{adversarial error}}(\Omega)$$

$$f_{\text{error}}(\Omega) = \mathcal{L}_{\text{validation}}(\Omega), \Omega = \{\{V, E\}, w, \theta, v\}$$

$$f_{\text{flash}}(\Omega) = \begin{cases} -\frac{||h_{\text{FB}}(w,\{V,E\})||_0}{\text{flash}_{\max}} \vee -\frac{\text{HIL information}}{\text{flash}_{\max}} \\ \infty, f_{\text{flash}}(\Omega) > \text{flash}_{\max} \end{cases}$$

$$f_{\text{latency}}(\Omega) = \frac{\text{FLOPS}}{\text{FLOPS}_{\text{target FLOPS}}} \vee \frac{\text{HIL information}}{\text{Latency}_{\text{target latency}}}$$

$$f_{\text{SRAM}}(\Omega) = \begin{cases} -\frac{\max_{l\in[1,L]}\{||x_l||_0 + ||a_l||_0\}}{\text{SRAM}_{\max}} \vee -\frac{\text{HIL information}}{\text{SRAM}_{\max}} \\ \infty, f_{\text{SRAM}}(\Omega) > \text{SRAM}_{\max} \end{cases}$$

$$f_{\text{adversarial error}}(\Omega) = 1 - \frac{1}{N}\sum_{i=0}^{N} q_i, \quad q_i = \begin{cases} 1, & \text{if} \quad y_{\text{pred}}^{x_{i,\text{validation}}} = y_{\text{pred}}^{x_{i,\text{validation},p}} \\ 0, & \text{otherwise} \end{cases}$$

Firstly, validation RMSE serves as a proxy for the error characteristics $f_{\text{error}}(\Omega)$ of the model candidate. Secondly, when real hardware is absent, we use the size of the flatbuffer model schema $h_{\text{FB}}(\cdot)$ [45] as a proxy for flash usage. Thirdly, we use the standard RAM usage model as a proxy for SRAM usage $f_{\text{SRAM}}(\Omega)$, with intermediate layer-wise activation maps and tensors being stored in SRAM [46]. Fourth, since model latency is linearly proportional to the FLOPS count for a variety of convolutional models for microcontrollers, we use FLOPS as a proxy for runtime latency $f_{\text{latency}}(\Omega)$ [47]. When HIL is available, we obtain the SRAM, flash, and latency parameters directly from the target compiler and real-time operating system (RTOS). All hardware parameters are normalized by device capacity or target metrics. Lastly, to get adversarial accuracy with negligible training or search cost, we perform FGSM attack on the trained candidate model on the validation set.

We use Gaussian process as the surrogate model to approximate f_{opt}, which allows priors on the distribution of moments to propagate forward as the search progresses. In addition, the domain of random scalarizations λ can be specified by the user to guide the parallel search acquisition functions (hallucination or K-means clustering) into the promising Pareto-optimal regions of the gradient plane. The acquisition function decides the next set of Ω_n to sample from the design space using Monte Carlo sampling with Upper-Confidence Bounds, which balances exploration and exploitation.

$$\hat{f}(\Omega) \sim \mathcal{GP}(\mu(\Omega), k(\Omega, \Omega'))$$
$$\Omega_t = \arg\max_{\Omega}(\mu_{t-1}(\Omega) + \beta^{0.5}\sigma_{t-1}(\Omega))$$

Table 2 showcases the results of applying our platform-aware robust NAS to optimize models for different hardware and model size. On average, the NAS formulation with adversarial robustness term yields models that have, on average, 9% more adversarial accuracy than handcrafted models and models found via NAS with no adversarial robustness cost. While the robust models are also larger than the non-robust models found via NAS, the models still fit within the memory bounds of the target device.

Table 2: Comparison of handcrafted models, models found via NAS without adversarial robustness cost, and models found via NAS with adversarial robustness cost.

Model	**Test Accuracy (%)**	**Adversarial Accuracy**	**Model Size (kB)**	**FLOPS (M)**
TCN (STM32F746ZG*, handcrafted)	93	18	68.8	12.57
TCN (NAS, STM32F746ZG, no adversarial term)	90	20	**44.4**	**8.52**
TCN (NAS, STM32F746ZG, with adversarial term)	**97**	**44**	101.4	23.2
TCN (NAS, STM32F446RE*, no adversarial term)	93	14	**54.93**	**8.79**
TCN (NAS, STM32F446RE, with adversarial term)	**93**	**27**	68.95	15.87
TCN (NAS, STM32L476RG*, no adversarial term)	90	14.3	**38.7**	**5.37**
TCN (NAS, STM32L476RG, with adversarial term)	**95**	**36.3**	96.97	21.57
ProtoNN (handcrafted)	**72**	**78**	27.8	-
ProtoNN (NAS, no adversarial term)	71	72	**1.16**	-
ProtoNN (NAS, with adversarial term)	70	76	20.4	-
Bonsai (handcrafted)	**73**	19.6	14.9	-
Bonsai (NAS, no adversarial term)	71	18.6	1.5	-
Bonsai (NAS, with adversarial term)	72	**20.6**	**1.5**	-

* Cortex M4 boards - STM32F746ZG: 320kB RAM, 1 MB flash; STM32F446RE: 192 kB RAM, 512 kB flash, STM32L476RG: 128 kB RAM, 1 MB flash

Conclusion: We showcased that TinyML models have a security problem from an adversarial perturbation perspective, and provided an inexpensive AutoML solution to provide robust models within tight resource constraints of TinyML devices. The code is open-sourced at https://github.com/swapnilsayansaha/tinyml_security

References:

[1]. Swapnil Sayan Saha, Sandeep Singh Sandha, and Mani Srivastava, “Machine Learning for Microcontroller-Class Hardware – A Review”, in IEEE Sensors Journal, 2022. (under review).
[2]. Leino, Klas, Zifan Wang, and Matt Fredrikson. "Globally-robust neural networks." International Conference on Machine Learning. PMLR, 2021.
[3]. Cohen, Jeremy, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing." International Conference on Machine Learning. PMLR, 2019.
[4]. Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." 2017 ieee symposium on security and privacy (sp). IEEE, 2017.
[5]. Ganin, Yaroslav, et al. "Domain-adversarial training of neural networks." The journal of machine learning research 17.1 (2016): 2096-2030.
[6]. Zhai, Runtian, et al. "MACER: Attack-free and Scalable Robust Training via Maximizing Certified Radius." International Conference on Learning Representations. 2019.
[7].Li, Bai, et al. "Certified adversarial robustness with additive noise." Advances in neural information processing systems 32 (2019).
[8]. Lin, Ji, Chuang Gan, and Song Han. "Defensive Quantization: When Efficiency Meets Robustness." International Conference on Learning Representations. 2018.
[9]. Gui, Shupeng, et al. "Model compression with adversarial robustness: A unified optimization framework." Advances in Neural Information Processing Systems 32 (2019).
[10]. Ye, Shaokai, et al. "Adversarial robustness vs. model compression, or both?." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.
[11]. Shumailov, Ilia, et al. "To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression." Proceedings of Machine Learning and Systems 1 (2019): 230-240.
[12]. Sehwag, Vikash, et al. "Hydra: Pruning adversarially robust neural networks." Advances in Neural Information Processing Systems 33 (2020): 19655-19666.
[13]. Guo, Minghao, et al. "When nas meets robustness: In search of robust architectures against adversarial attacks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
[14]. Zhao, Yue, et al. "Seeing isn't believing: Towards more robust adversarial attack against real world object detectors." Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019.
[15]. Eykholt, Kevin, et al. "Robust physical-world attacks on deep learning visual classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[16]. Zhang, Guoming, et al. "Dolphinattack: Inaudible voice commands." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.
[17]. Li, Juncheng, et al. "Adversarial music: Real world audio adversary against wake-word detection system." Advances in Neural Information Processing Systems 32 (2019).
[18]. Sugawara, Takeshi, et al. "Light Commands:{Laser-Based} Audio Injection Attacks on {Voice-Controllable} Systems." 29th USENIX Security Symposium (USENIX Security 20). 2020.
[19]. El-Rewini, Zeinab, et al. "Cybersecurity attacks in vehicular sensors." IEEE Sensors Journal 20.22 (2020): 13752-13767.
[20]. Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[21]. Zhou, Wen, et al. "Transferable adversarial perturbations." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
[22]. Fawaz, Hassan Ismail, et al. "Adversarial attacks on deep neural networks for time series classification." 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019.
[23]. Deogirikar, Jyoti, and Amarsinh Vidhate. "Security attacks in IoT: A survey." 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 2017.
[24]. Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." International Conference on Learning Representations (2015).
[25]. Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale." International Conference on Learning Representations (2017).
[26]. Krizhevsky, A. "Learning Multiple Layers of Features from Tiny Images." Master's thesis, University of Toronto (2009).
[27]. Warden, Pete. "Speech commands: A dataset for limited-vocabulary speech recognition." arXiv preprint arXiv:1804.03209 (2018).
[28]. Swapnil Sayan Saha, Sandeep Singh Sandha, Siyou Pei, Vivek Jain, Ziqi Wang, Yuchen Li, Ankur Sarker, and Mani Srivastava, “AURITUS: An Open-Source Optimization Toolkit for Training and Deployment of Human Movement Models and Filters using Earables,” in Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), ACM New York, NY, USA, 2022. (under re-review after revision).
[29]. Banbury, Colby, et al. "MLPerf Tiny Benchmark." Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). 2021.
[30]. Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR, 2019.
[31]. Tan, Mingxing, and Quoc Le. "Efficientnetv2: Smaller models and faster training." International Conference on Machine Learning. PMLR, 2021.
[32]. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[33]. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." International Conference on Learning Representations (2015).
[34]. Lin, Ji, et al. "Mcunet: Tiny deep learning on iot devices." Advances in Neural Information Processing Systems 33 (2020): 11711-11722.
[35]. de Andrade, Douglas Coimbra, et al. "A neural attention model for speech command recognition." arXiv preprint arXiv:1808.08929 (2018).
[36]. Zhang, Yundong, et al. "Hello edge: Keyword spotting on microcontrollers." arXiv preprint arXiv:1711.07128 (2017).
[37]. Lea, Colin, et al. "Temporal convolutional networks: A unified approach to action segmentation." European Conference on Computer Vision. Springer, Cham, 2016.
[38]. Oord, Aaron van den, et al. "Wavenet: A generative model for raw audio." Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9), 125, 2016..
[39]. Saha, Swapnil Sayan, Sandeep Singh Sandha, and Mani Srivastava. "Deep Convolutional Bidirectional LSTM for Complex Activity Recognition with Missing Data." Human Activity Recognition Challenge. Springer, Singapore, 2020. 39-53.
[40]. Kumar, Ashish, Saurabh Goyal, and Manik Varma. "Resource-efficient machine learning in 2 KB RAM for the internet of things." International Conference on Machine Learning. PMLR, 2017.
[41]. Gupta, Chirag, et al. "Protonn: Compressed and accurate knn for resource-scarce devices." International Conference on Machine Learning. PMLR, 2017.
[42]. Kusupati, Aditya, et al. "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network." Advances in Neural Information Processing Systems 31 (2018).
[43]. Pang, Ren, et al. "On the Security Risks of AutoML." 31st USENIX Security Symposium (USENIX Security 22). 2022.

[44]. Sandeep Singh Sandha, Mohit Aggarwal, Swapnil Sayan Saha, and Mani Srivastava. ”Enabling Hyperparameter Tuning of Machine Learning Classifiers in Production” Third IEEE International Conference on Cognitive Machine Intelligence. IEEE, 2021. 1-10.
[45]. David, Robert, et al. "TensorFlow lite micro: Embedded machine learning for tinyml systems." Proceedings of Machine Learning and Systems 3 (2021): 800-811.
[46]. Fedorov, Igor, et al. "Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers." Advances in Neural Information Processing Systems 32 (2019).
[47]. Banbury, Colby, et al. "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers." Proceedings of Machine Learning and Systems 3 (2021): 517-532.