If It's Not Efficient, It's Not Usable: Real-Time OOD Detection with Latent De-Biasing and High-Quality Negative Samples

Yuchuan Li[1], Jae-Mo Kang[2], and Il-Min Kim[1,✉]

[1] Queen's University, Canada
[2] Kyungpook National University, South Korea

Abstract. On-device AI for resource-constrained systems, such as wearables and IoT sensors, requires models that are both highly efficient and reliable for mission-critical applications. Out-of-Distribution (OOD) detection is vital for ensuring this reliability, yet mainstream OOD research has focused on achieving state-of-the-art (SOTA) accuracy without regard for inference-time computational cost. This has led to powerful but complex methods (e.g., CLIP-based or supervised learning-based models) that are orders of magnitude too slow for real-world on-device deployment. We argue that the more meaningful benchmark for this domain is maximizing accuracy within a practical, constrained inference computational budget. However, reducing inference complexity is typically achieved at the expense of lower performance, exhibiting a trade-off between inference speed and accuracy. This paper breaks that trade-off. We introduce a novel framework that establishes an accuracy-SOTA under constrained inference complexity. This is achieved through two key innovations: (i) a latent-space de-biasing technique that learns to remove pixel-intensity bias during training, drastically reducing inference complexity, and (ii) an intelligent negative sampling strategy that generates high-quality negative samples to significantly boost performance. On both a real edge device and standard GPU infrastructure, extensive experiments reveal that our method surpasses existing accuracy-SOTA OOD detection methods within a TinyML-appropriate inference budget.

Keywords: Out-of-Distribution Detection · On-device AI

1 Introduction

Deploying machine learning on small edge devices, such as wearables, Internet of Things (IoT) sensors, and small drones, is rapidly emerging as a critical paradigm in Artificial Intelligence (AI), often referred to as TinyML. These devices are integral to mission-critical applications like health monitoring, wildfire detection, and search-and-rescue missions, where (near) real-time decision-making is essential and delays can have severe consequences [1, 9, 28]. However, these

✉ Corresponding author: IlMin.Kim@queensu.ca

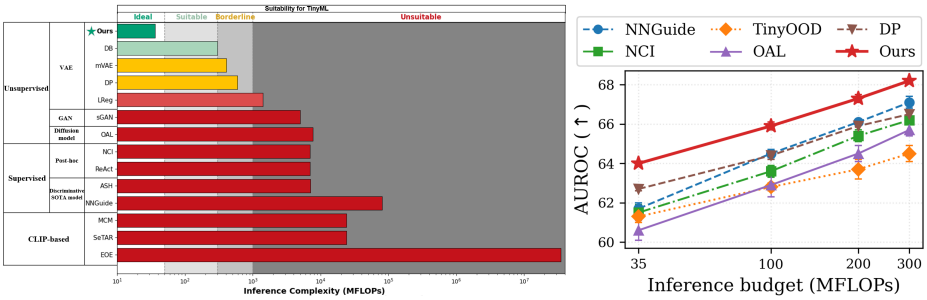
devices are battery-powered and possess highly limited computing resources, making (near) real-time inference a significant challenge. While cloud-based computation offers a powerful alternative, it is often impractical due to latency, connectivity issues, data privacy concerns, and cost. Thus, there is an urgent need for compact AI models capable of performing fast *inference* under stringent resource constraints [4, 17]. Clearly, training complexity is much less of a concern for TinyML since models can be trained offline on powerful machines before being deployed to resource-constrained devices.

Out-of-Distribution (OOD) detection is a crucial safeguard for ensuring the reliability and safety of AI by identifying inputs that deviate from in-distribution (ID) data. This is particularly vital in TinyML, where devices operate in diverse and unpredictable environments. Notably, the mainstream research today pursues maximizing performance by leveraging larger and larger models, which are often equipped with increasingly sophisticated inference mechanisms. The prevailing goal is to achieve the state-of-the-art (SOTA) accuracy *without constraining inference complexity*. However, this pursuit has largely ignored the limited inference computational capabilities of the devices targeted by TinyML, which require inference complexities that are orders of magnitude beyond what they can support. SOTA accuracy becomes meaningless for resource-constrained devices if inference latency prevents meeting (near) real-time application requirements. This trend runs counter to the core principles of TinyML, where compute resources are scarce and (near) real-time inference is often essential.

This reality necessitates a different research paradigm: the pursuit of accuracy-SOTA *under constrained inference complexity*. In this paradigm, we first define an acceptable computational budget by the target hardware (e.g., in Mega Floating-Point Operations (MFLOPs)) and then aim to maximize detection accuracy *within* that fixed budget in inference. This aligns with the de-facto industry standard: *if a model cannot run efficiently on-device, its accuracy is irrelevant for real-world deployment.*

Modern OOD detection methods can be grouped into three major paradigms: Contrastive Language-Image Pre-training (CLIP)-based, supervised learning-based, and unsupervised learning-based. CLIP-based methods achieve strong OOD performance via prompt-based image-text similarity; however, their efficacy depends on large-scale, pre-trained vision–language models whose cross-modal alignment cannot be preserved under compression, making them impractical for resource-constrained settings. Supervised methods, including discriminative and post-hoc approaches, require explicit class labels within the in-distribution (ID) training data. In contrast, unsupervised methods, including Generative Adversarial Networks (GANs), diffusion models, and Variational Autoencoders (VAEs), require only the *unlabeled* ID samples, relying on coarse labeling to separate ID from OOD samples, without needing class labels within the ID data. This is more practical for many real-world applications (e.g., medical imaging).

To assess the viability of OOD detection methods for TinyML, it is essential to examine whether their inference complexities fall within the constraints of TinyML hardware. Numerous studies establish that TinyML applications operate



(a) Inference complexity (MFLOPs, ↓) comparison **(b)** AUROC (↑) scores comparison

Fig. 1: Efficiency & accuracy comparison. **(a)** Inference complexity (in MFLOPs) comparison on ImageNet-1k benchmark for image resolution of 224×224 based on the original model of each method. **(b)** AUROC score comparison via multi-budget Pareto curves, averaged over test cases of the ImageNet-1k benchmark. For each method and budget, the best variant with an inference cost no higher than the budget cap is selected.

within a ceiling of approximately 300 MFLOPs, with many requiring even far less complexity [13, 15, 30]. For a detailed discussion of TinyML requirements, refer to Appendix A. Unfortunately, as illustrated in Figure 1a, all supervised, all CLIP-based, and most unsupervised methods (except some VAE methods), which collectively dominate the accuracy-SOTA space, far exceed 300 MFLOPs. For example, the inference complexity of the modern CLIP-based method is as high as about 35,000,000 MFLOPs. Notably, VAE-based methods offer lower inference complexity than others, although they still require several hundred MFLOPs or more for inference complexity, which often narrows (or may preclude) their applicability in TinyML scenarios. Moreover, the lower inference complexity of the existing VAE methods often comes at the cost of reduced OOD detection accuracy. Meanwhile, the inference complexity of our proposed method is the lowest, at about 35 MFLOPs. In the context of TinyML, blindly comparing OOD detection accuracy is typically meaningless due to the substantial disparity in inference complexity (e.g., 35 MFLOPs vs. 35,000,000 MFLOPs).

This raises an important and practical question: which approach delivers the highest accuracy when constrained to a TinyML-appropriate inference budget? This question highlights a clear research need: a method that is natively designed for the TinyML regime and capable of achieving accuracy-SOTA without exceeding the inference complexity limit. This is precisely the goal of our work.

In this work, we propose a novel unsupervised (VAE-based) framework by holistically designing a model for both efficiency and performance at the same time. To examine the *real-world* efficiency, we perform experiments on an actual TinyML platform, the NVIDIA Jetson Orin Nano 8 GB devkit. Figure 1b shows the Area Under the Receiver Operating Characteristic Curve (AUROC) scores of our method and other methods including GAN, diffusion model, post-hoc and SOTA discriminative methods via multi-budget Pareto analysis within a fair, TinyML-appropriate inference budget range (up to TinyML's 300 MFLOPs

ceiling). For each method and each budget, we select the best-performing variant whose inference cost is no higher than the budget cap. CLIP-based methods are not compared because their efficacy is tied to "pre-trained" (transformer-based) vision and language encoders, which cannot be reduced without losing the learned cross-modal alignment, and although compact CLIP variants (e.g., TinyCLIP [39] and MobileCLIP [38]) exist, none meets the TinyML inference budget while preserving the cross-modal alignment these methods rely on. Figure 1b shows that our method consistently surpasses competing methods across the TinyML-appropriate inference budget range.

Our goal is to achieve two critical objectives: (i) ultra-low inference complexity and (ii) high performance across diverse datasets. To address the first goal, we design a novel de-biasing approach that trains the model to internalize the de-biasing process directly within the latent space. To accomplish the second, we propose an intelligent strategy that generates high-quality negative (virtual OOD) samples directly from ID data. Both real-device experiments and evaluations on standard GPU infrastructure show that our method consistently outperforms existing accuracy-SOTA OOD methods across multiple benchmarks under a TinyML-appropriate inference budget.

Our study's key contributions include the following:

- We propose a novel and inference-efficient pixel-intensity de-biasing method that operates directly in the latent space, allowing the model to learn this process during training. This significantly reduces inference complexity, making it ideal for resource-constrained TinyML applications.
- We introduce a new and effective mechanism to generate high-quality negative samples, which effectively enhances OOD detection accuracy in TinyML.
- Experiments on both real TinyML device and standard GPU setup show that our method sets a new accuracy-SOTA, outperforming existing SOTA OOD detection methods under the same TinyML-appropriate inference budget.

2 TinyML and Related Work

TinyML devices typically operate within a computational envelope ranging from roughly 0.1 MFLOPs to a few hundred MFLOPs, with most practical applications targeting the 0.1–100 MFLOP range to balance real-time performance and energy efficiency. Ultra-low complexity models (0.1–1 MFLOPs) are designed for extremely constrained hardware, such as microcontrollers in wearables, enabling tasks like always-on keyword spotting with only a few hundred thousand operations per inference and significant energy savings [44]. Moderate complexity models (1–100 MFLOPs) expand capabilities to more demanding tasks, including image classification, image segmentation, and human posture recognition on devices like IoT sensors and low-power embedded systems, while still maintaining low latency and power draw [3, 7, 24, 32]. Higher-end TinyML models (100–$\sim$300 MFLOPs) are employed in more capable devices where additional computational resources allow for more sophisticated deep networks, though they require careful trade-offs regarding latency and battery life [18, 19, 27]. Generally, $\sim$300 MFLOPs

is considered an upper limit for TinyML, and if the inference complexity significantly exceeds this threshold, deploying such models becomes highly challenging. To provide a fair performance comparison for TinyML, existing SOTA OOD detection methods are benchmarked at the same inference speed (latency) based on an actual TinyML device. Further details on the inference computational complexity for TinyML devices are available in Appendix A.

Modern OOD detection research can be grouped into three paradigms. Supervised learning-based methods, including discriminative model-based and post-hoc techniques, use class labels to build detection models [10, 11, 15, 23, 25, 31, 35]. In contrast, unsupervised learning-based methods require only unlabeled in-distribution data. This category includes large, complex generative models like GANs and diffusion models [14, 34] as well as more efficient (especially in the inference phase) VAEs [2, 5, 8, 40, 42], which offer a more viable foundation for resource-constrained environments, as demonstrated by Figure 1a. Recently, CLIP-based methods [6,22,29] hold the unconstrained accuracy-SOTA but require thousands of MFLOPs, rendering them incompatible with TinyML scenarios.

In terms of inference complexity, VAE-based methods offer lightweight architectures better-suited for TinyML than others. However, their performance has remained suboptimal, as methods optimized for speed tend to suffer significant drops in detection accuracy [8], while attempts to improve accuracy often lead to increased computational cost [2, 5, 40, 42]. Our work breaks this impasse. By introducing novel architectural and training strategies, we achieve accuracy-SOTA with the same inference speed on an actual TinyML device, outperforming existing supervised and unsupervised-based SOTA OOD detection methods.

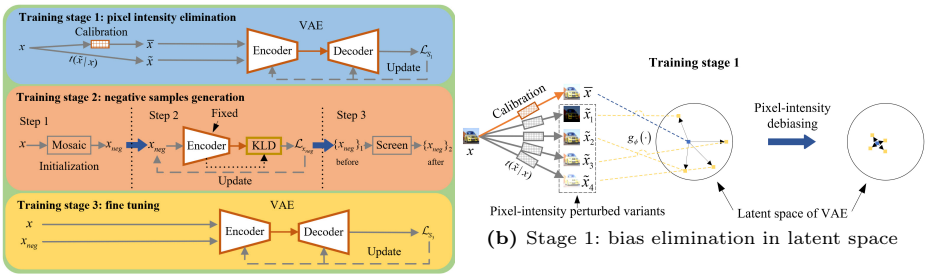
3 Proposed Method

As previously discussed, we select VAE as our base model, considering it the most suitable model for TinyML. To simultaneously enhance inference efficiency and optimize OOD detection performance, we propose two innovations, each achieving one of these goals: (i) eliminating pixel-intensity bias in the latent space of VAEs and (ii) intelligently generating "hard" negative samples for training.

3.1 Training Procedure

In our method, only the dataset D_{ID} of ID samples is given for training and the VAE is trained through a three-stage process, as shown in Figure 2a. Each stage is detailed in the following subsections.

Training Stage 1: learning pixel-intensity bias elimination in latent space In the first training stage, we address pixel-intensity bias, a crucial factor in VAE-based OOD detection. In existing methods, this bias is typically corrected during the inference phase by applying adjustments at the output of the decoder, mostly relying on importance sampling to estimate log-likelihood values [8].



(a) Three-stage training overview

(b) Stage 1: bias elimination in latent space

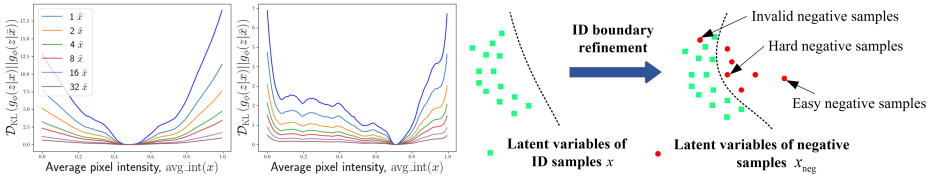
Fig. 2: Training pipeline. **(a)** An overview of the three stages of the training procedure for our proposed approach (top to bottom). The first stage effectively eliminates pixel-intensity bias within the latent space. In the second stage, hard and useful high-quality negative samples are synthesized through three steps: (i) initializing, (ii) updating, and (iii) screening. The third stage fine-tunes the model pre-trained in Stage 1, yielding the ultimate model used for inference. **(b)** Training Stage 1 for pixel-intensity bias elimination in the VAE's latent space (i.e., at the output of the encoder). For each x in D_{ID}, its pixel-intensity calibrated version $\bar{x}$, which serves as a reference point in the latent space, and its pixel-intensity perturbed variants $\{\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4\}$ are generated. Our loss $\mathcal{L}_{S_1}$ is designed to push all $\tilde{x}$ toward $\bar{x}$ in the latent space.

However, achieving high estimation accuracy with this approach demands a large number of importance samples (e.g., hundreds), resulting in high inference complexity, which makes it impractical for resource-constrained settings. In contrast, our method trains the model to *learn* how to eliminate pixel-intensity bias directly within the *latent* space during *training*. By learning to de-bias in the latent space, our approach removes the need for repeated sampling at inference, substantially reducing inference complexity while maintaining high accuracy. This innovation not only improves the efficiency of OOD detection but also decreases model size, making it highly suitable for deployment in limited-resource environments.

Our approach of pixel-intensity bias elimination, shown in Figure 2b, is to ensure that any pixel-intensity perturbed variants of an input image exhibit close proximity with one another in the latent space. To this end, based on inspiration from CR-VAE [33], we propose a novel loss $\mathcal{L}_{S_1}$ using a pixel-intensity perturbation operation, denoted by $t(\tilde{x}|x)$, as follows:

$$\mathcal{L}_{S_1} = \mathcal{L}^{\text{VAE}}_{\theta,\phi}(\bar{x}) + \mathbb{E}_{\tilde{x}}\left[\mathcal{L}^{\text{VAE}}_{\theta,\phi}(\tilde{x})\right] + \lambda_{S_1} \cdot \mathbb{E}_{\tilde{x}}\left[\mathcal{D}_{\text{KL}}\left(g_\phi(z|\bar{x}) \,||\, g_\phi(z|\tilde{x})\right)\right] \quad (1)$$

where $\tilde{x} \sim t(\tilde{x}|x)$ represents the pixel-intensity variants of the original input x. The encoder parameterized by ϕ maps an input x to its latent variable z with the posterior distribution $g_\phi(z|x) = \mathcal{N}_z(\mu_\phi(x), \sigma^2_\phi(x))$, where $\mathcal{N}(\mu, \sigma^2)$ represents a multivariate Gaussian distribution with mean vector μ and diagonal covariance matrix σ^2, which will be simply called mean and variance. With the reparameterization trick [20], the mean $\mu_\phi(x)$ and variance $\sigma^2_\phi(x)$ are produced by the encoder for input x. The decoder is parameterized by θ. Also, $\mathcal{L}^{\text{VAE}}_{\theta,\phi}(\cdot)$



(a) Impact of varying quantities of $\tilde{x}$ on the KL divergence values $\mathcal{D}_{\mathrm{KL}}\left(g_\phi(z|x)||g_\phi(z|\bar{x})\right)$ **(b)** Demonstration of different types of negative samples in latent space

Fig. 3: Pixel-intensity de-biasing and negative-sample taxonomy. **(a)** An illustration of the impact of varying quantities of $\tilde{x}$ on the KL divergence values $\mathcal{D}_{\mathrm{KL}}\left(g_\phi(z|x)||g_\phi(z|\bar{x})\right)$. The two figures are plotted from the corresponding KL divergence values for continuous Bernoulli (left) and categorical (right) VAEs on all samples x from CIFAR-10 with a VAE model trained on the CIFAR-10 dataset. **(b)** Negative samples are those that are external to the boundary of ID samples in the latent space. Easy negative samples are far from the boundary, whereas hard negative samples are close to the boundary, yet outside of the ID. Such hard negative samples are more advantageous in effectively shaping the decision boundary and ultimately enhancing the performance of OOD detection. Negative samples inside ID are invalid.

symbolizes the standard VAE loss composed of KL divergence loss and the reconstruction loss. The third term of $\mathcal{D}_{\mathrm{KL}}(\cdot)$ serves as a regularizer in the latent space of VAE.

As shown in Figure 2b, for each x in the ID dataset D_{ID}, its pixel-intensity calibrated version $\bar{x}$ is constructed such that in the latent space it serves as a reference point for all pixel-intensity perturbed variants $\tilde{x}$ of x. In essence, $\bar{x}$ constructed for each x yields the minimum KL divergence with the zero-mean unit standard Gaussian in the latent space. More details of constructing $\bar{x}$ are given in Appendix B.1. For each x, its multiple pixel-intensity variants $\tilde{x}$ are also generated by $\tilde{x} \sim t(\tilde{x}|x)$, where the average pixel-intensity distribution $t(\tilde{x}|x)$ of $\tilde{x}$ given x is a uniform distribution over the entire (normalized) pixel-intensity range $[0, 1]$. Then, through the third term $\mathcal{D}_{\mathrm{KL}}(\cdot)$ in Equation (1), all variants $\tilde{x}$ of x are forced to converge towards $\bar{x}$ in the latent space to mitigate the pixel-intensity bias in the latent space. In our proposed scheme, those $\tilde{x}$ are generated such that they are uniformly distributed over the entire pixel-intensity range of $[0, 1]$, which is effective in accommodating any OOD samples in the inference that are drawn from unknown distributions. To confirm the regularization effect, we ran simulations on CIFAR-10 for varying numbers of variants $\tilde{x}$ from one to 32, which are shown in Figure 3a. Clearly, the more $\tilde{x}$ are used, the flatter the KL divergence curves are. When 32 $\tilde{x}$ variants are used, the curves are almost flattened, meaning that the pixel-intensity bias is essentially removed in the latent space. Extended results of pixel-intensity bias elimination in other datasets and a comparison between different techniques for negative sample generation are available in Appendix B.2.

Training Stage 2: generation of negative samples Stage 2 constitutes another pivotal phase of our proposed method tasked with the generation of special samples (from ID samples), namely negative samples, which will be utilized in training Stage 3 to further enhance OOD detection performance. In practice, it is not possible to create the OOD samples that will actually be faced in the inference phase, because the distributions of the actual OOD samples are unknown a priori. What is possible in practice is to generate the samples that are external to the (distribution of) ID samples, which we will call the negative samples, denoted by x_{neg} (to distinguish them from the actual OOD samples). In the latent space, if the negative samples are too far from the (distribution of) ID samples, they are not really useful to help the model better learn (or refine) the boundary of the (distribution of) ID samples. To be useful, the negative samples must be proximate, yet external, to the boundary of the ID samples. The former type of negative samples is easy (and less useful), and the latter is hard (and useful), as shown in Figure 3b. The primary goal of our training Stage 2 is to generate hard negative samples.

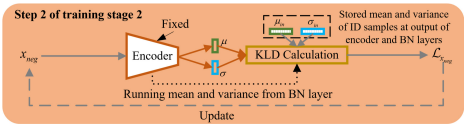
To generate hard negative samples during the second stage of training, as shown in the middle sub-figure of Figure 2a, we developed a three-step approach: (i) initializing, (ii) updating, and (iii) screening. In the first step, we initialize the negative samples x_{neg} by applying a Mosaic data augmentation to ID samples x, followed by perturbations of the partitioned blocks of Mosaic. Through this process, the high-level semantic information of the input samples is altered, although the low-level semantic information is still (partially) preserved. In the latent space, the initialized sample is marginally pushed towards the periphery of the ID area.

In the second step of training Stage 2, following the procedure in Figure 4a, we update x_{neg} such that they are sufficiently pushed away from ID samples, ensuring that x_{neg} become *valid* negative samples. To this end, we extract the encoder portion of the VAE model trained during Stage 1, which is kept fixed (frozen) throughout this step, and we apply x_{neg} as input to the encoder. While fixing the encoder, the input x_{neg} itself is directly updated by minimizing our proposed loss function $\mathcal{L}_{x_{\text{neg}}}$:

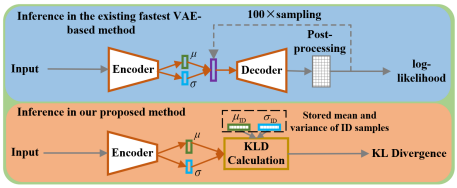
$$\begin{aligned}\mathcal{L}_{x_{\text{neg}}} = &- \mathcal{D}_{\text{KL}}\left(g_\phi(z|x_{\text{neg}})||\mathcal{N}_z(\mu_{\text{ID}}, \sigma^2_{\text{ID}})\right) \\ &- \sum_{i=1}^{L} \lambda_i \mathcal{D}_{\text{KL}}\left(g_\phi^{i\text{-BN}}(z_i|x_{\text{neg}})||\mathcal{N}_{z_i}(\mu_{\text{ID}}^{i\text{-BN}}, \sigma_{\text{ID}}^{2,i\text{-BN}})\right) + \lambda \left\|x_{\text{neg}}\right\|\end{aligned} \tag{2}$$

where $\mu_{\text{ID}} = \mathbb{E}_{x\in D_{\text{ID}}}[\mu_\phi(x)]$ and $\sigma^2_{\text{ID}} = \mathbb{E}_{x\in D_{\text{ID}}}[\sigma^2_\phi(x)]$. Also, $g_\phi^{i\text{-BN}}(z_i|x_{\text{neg}}) = \mathcal{N}_{z_i}(\mu_\phi^{i\text{-BN}}(x_{\text{neg}}), \sigma_\phi^{2,i\text{-BN}}(x_{\text{neg}}))$, $\mu_{\text{ID}}^{i\text{-BN}} = \mathbb{E}_{x\in D_{\text{ID}}}[\mu_\phi^{i\text{-BN}}(x)]$, and $\sigma_{\text{ID}}^{2,i\text{-BN}} = \mathbb{E}_{x\in D_{\text{ID}}}[\sigma_\phi^{2,i\text{-BN}}(x)]$, where $\mu_\phi^{i\text{-BN}}(y)$ and $\sigma_\phi^{2,i\text{-BN}}(y)$ are the mean and variance at the i-th BatchNorm (BN) layer within the encoder ϕ when the input is y.

In Equation (2), the first KL divergence term ensures that, in the latent space of VAE, x_{neg} differs from ID samples. This KL divergence is computed in the latent space using the running mean $\mu_\phi(x_{\text{neg}})$ and variance $\sigma^2_\phi(x_{\text{neg}})$ for input x_{neg} and the average mean μ_{ID} and variance σ^2_{ID} for the entire ID samples in



(a) Step 2 of training Stage 2 of our method



(b) Inference procedures of the existing fastest VAE-based method [8] and our method

Fig. 4: Step 2 of training Stage 2 of our method and inference pipeline comparison. **(a)** Step 2 of training Stage 2. Using the encoder part, which is retrieved from the trained VAE model in the first stage and kept fixed, x_{neg} are directly updated by minimizing $\mathcal{L}_{x_{\text{neg}}}$ in order to become valid negative samples. **(b)** Inference procedures of the existing fastest VAE-based method [8] and our method. The method [8] needs hundreds of sampling operations to estimate the log-likelihood. By contrast, our method obviates this computationally demanding operation and directly generates OOD detection predictions in the latent space by the KL divergence, which significantly reduces the computational burden at a lower memory requirement.

D_{ID} that were computed and stored from Stage 1. To further ensure that x_{neg} differs from ID samples at all feature levels (in addition to the latent space), we introduce the second KL divergence term in Equation (2), which is inspired by [41]. The fundamental idea we adopted from [41] is to use the feature statistics of the training data stored in the BN layers of the model. Assuming there are L BN layers in the encoder, those L KL divergence terms are computed using the running mean $\mu_\phi^{i\text{-BN}}(x_{\text{neg}})$ and variance $\sigma_\phi^{2,i\text{-BN}}(x_{\text{neg}})$ for input x_{neg} and the average mean $\mu_{\text{ID}}^{i\text{-BN}}$ and variance $\sigma_{\text{ID}}^{2,i\text{-BN}}$ for all ID samples in D_{ID}, which are stored in the BN layers of the encoder trained in Stage 1. Although inspired by [41], our approach functions in an opposite way to the method of [41]: in our approach, the KL divergence is maximized to generate the "negative" samples that are different from the ID samples at all feature levels, whereas in [41] the KL divergence is minimized to generate "positive" samples with the same distributions of the ID (or training) data. The third term $\lambda \|x_{\text{neg}}\|$ serves as regularization.

Finally, the third step of training Stage 2 is to screen out both invalid and easy negative samples, which contributes significantly to refining the OOD detection decision boundary. Letting D_{neg} denote the set of all negative samples obtained in Step 2, we filter them as follows:

$$D_{\text{neg}}^{\text{hard}} = \left\{ x_{\text{neg}} \middle| \epsilon_2 < q(x_{\text{neg}}; \mu_{\text{neg}}, \sigma_{\text{neg}}^2) < \epsilon_1, x_{\text{neg}} \in D_{\text{neg}} \right\} \tag{3}$$

where $q(x_{\text{neg}}; \mu_{\text{neg}}, \sigma_{\text{neg}}^2)$ is given by

$$q(x_{\text{neg}}; \mu_{\text{neg}}, \sigma_{\text{neg}}^2) = \frac{e^{-\frac{1}{2}(z(x_{\text{neg}})-\mu_{\text{neg}})^T(\sigma_{\text{neg}}^2)^{-1}(z(x_{\text{neg}})-\mu_{\text{neg}})}}{(2\pi)^{K/2}\det(\sigma_{\text{neg}}^2)^{1/2}} \tag{4}$$

In the above equation, $z(x_{\text{neg}}) \sim g_\phi(z|x_{\text{neg}})$, $\mu_{\text{neg}} = \mathbb{E}_{y \in D_{\text{neg}}}[\mu_\phi(y)]$, $\sigma^2_{\text{neg}} = \mathbb{E}_{y \in D_{\text{neg}}}[\sigma^2_\phi(y)]$, and K is the dimension of the latent variable z. In light of the mechanism of Step 2 (the negative samples were pushed away from ID samples), "too highly (negative sample) likely" ones in D_{neg} with their probabilities higher than ϵ_1 should be removed, because they could be *easy* negative samples. On the other hand, "too (highly negative sample) unlikely" ones in D_{neg} with their probabilities lower than ϵ_2 should be removed, because they could be *invalid* negative samples. In our experiments, ϵ_1 and ϵ_2 are chosen based on the 95th and the 5th percentiles, respectively, of all negative samples x_{neg}. Further discussions about this screening procedure can be found in Appendix B.3.

Training Stage 3: fine-tuning by ID and negative samples Training Stage 3, as illustrated in Figure 2a, fine-tunes the model trained in Stage 1 utilizing both the original ID samples in D_{ID} and the carefully crafted negative samples in $D^{\text{hard}}_{\text{neg}}$ from Stage 2 as input based on the following loss:

$$\mathcal{L}_{S_3} = \mathcal{L}^{\text{VAE}}_{\theta,\phi}(x) + \lambda_{S_3} \cdot \mathcal{L}^{\text{neg}}_{\theta,\phi}(x_{\text{neg}}) \tag{5}$$

for $x \in D_{\text{ID}}$ and $x_{\text{neg}} \in D^{\text{hard}}_{\text{neg}}$, where $\mathcal{L}^{\text{neg}}_{\theta,\phi}(x_{\text{neg}})$ is defined as:

$$\mathcal{L}^{\text{neg}}_{\theta,\phi}(x_{\text{neg}}) = \max\left(0, \delta - \mathcal{L}_{\text{BCE}}\left(f_\theta\left(g_\phi\left(z|x_{\text{neg}}\right)\right)\right)\right) \tag{6}$$

In this equation, $f_\theta(\cdot)$ represents the decoder. The parameter δ denotes a predetermined threshold for the reconstruction loss of the negative sample x_{neg}. Also, $\mathcal{L}_{\text{BCE}}$ corresponds to the Binary Cross-Entropy loss. Rationales behind our selection of $\mathcal{L}_{\text{BCE}}$ against $\mathcal{L}_{\text{MSE}}$ are available in Appendix B.4.

3.2 Inference Procedure

Figure 4b shows the inference procedures of our method in comparison to the fastest existing OOD detection method [8], which utilized post-processing for pixel-intensity de-biasing and estimated the log-likelihood by importance sampling, demanding hundreds of sampling repetitions for inference. By contrast, in the inference phase of our proposed scheme, as the OOD detection score, we simply compute the KL divergence $D_{\text{KL}}(g_\phi(z|x_{\text{test}}) \| \mathcal{N}_z(\mu_{\text{ID}}, \sigma^2_{\text{ID}}))$ using the running mean $\mu_\phi(x_{\text{test}})$ and variance $\sigma^2_\phi(x_{\text{test}})$ for each test image x_{test} and the stored average mean μ_{ID} and variance σ^2_{ID}, which were learned on ID samples and stored during training. If the computed KL divergence is greater than a threshold, the test image is declared to be OOD; otherwise, it is ID. The threshold is selected to achieve a specific target performance metric, such as FPR95.

Benefiting from pixel-intensity de-biasing in the latent space that *was* already learned during *training*, our method obviates any need for estimating log-likelihood values, importance sampling, or de-biasing post-processing in inference. Furthermore, the intelligently crafted hard negative samples improve the OOD detection performance. Overall, our proposed method simultaneously offers faster inference than the existing fastest method [8], even with a smaller model in inference and higher OOD detection accuracy.

4 Experiments

We follow popular OOD detection benchmarks [43], including CIFAR-100 for both near (Tiny-ImageNet and CIFAR-10) and far (Texture, SVHN, and Places365) OOD test cases, and ImageNet-1k for both near (NINCO and SSB-hard) and far (iNaturalist, Texture, and OpenImage-O) OOD test cases. The performance comparisons are presented in AUROC (↑, the Area Under the Receiver Operating Characteristic Curve) and FPR95 (↓, False-Positive Rate at a True-Positive Rate of 95%), which are both reported in average and Standard Error of the Mean (SEM) values based on 10 different random seeds. Additional details on the experiments and dataset setups can be found in Appendices B.5 and B.6.

4.1 Baseline Methods

We consider a comprehensive set of competing OOD detection methods, including discriminative model-based (ASH [10], NNGuide [31], TinyOOD [23]), post-hoc (Energy [26], ReAct [35], NCI [25]), GAN-based (sGAN [34]) and diffusion-based (OAL [14]) methods, as well as representative VAE-based methods (DB [8], mVAE [2], DP [5]). Their backbones are scaled down to match the inference speed (measured in latency) of ours on the TinyML device for a fair comparison in TinyML. We include CLIP-based methods only as unconstrained references (Appendix C.1), since their performance hinges on fixed, large-scale pre-training: the parameters of a vision–language backbone cannot be scaled down without losing the learned cross-modal alignment, and although compact CLIP variants (e.g., TinyCLIP [39] and MobileCLIP [38]) exist, none meets the TinyML inference budget while retaining the cross-modal alignment that OOD scores require.

4.2 Performance on Real TinyML Device

For real TinyML device experiments, we evaluate all methods on an NVIDIA Jetson Orin Nano 8 GB devkit, a popular TinyML device, to ensure the alignment of test conditions with real-world TinyML-based OOD detection applications. We present the main results for the OOD detection in Table 1 and comparison with other negative sample generation methods in Table 2. Extended results of performance comparison with the comprehensive set of competing methods and SEM for all methods, more details on the TinyML hardware setup and inference speed comparison, and more visualizations of our synthetic negative samples are available in Appendices C.1, C.2, and C.3, respectively.

Main results on OOD detection performance. Table 1 presents a comparison of OOD detection performance between our method and other baseline methods. Our method consistently outperforms all baselines under a TinyML-appropriate inference budget on a real TinyML device, including methods designed for high performance with large backbones (e.g., NNGuide) and methods proposed for high efficiency (e.g., TinyOOD and DP). This is achieved through the holistic design of our method, balancing efficiency and accuracy by introducing novel architectural and training strategies for both targets.

Table 1: AUROC (↑)/FPR95 (↓) scores on (**n**)ear and (**f**)ar-OOD test cases of CIFAR-100 and ImageNet-1k benchmarks. All methods have the same inference speed (latency) on the real TinyML device in this comparison. For readability, the table is split into two parts that share the same ID-OOD pairs: the upper part reports supervised methods and the lower part reports unsupervised methods. **Bold** numbers denote the best results.

ID dataset	OOD dataset	Supervised methods					
		Post-hoc			Discriminative model		
		Energy	ReAct	NCI	ASH	NNGuide	TinyOOD
CIFAR-100	Tiny-ImageNet (**n**)	63.5/88.9	63.7/88.7	64.2/87.6	64.9/87.9	65.1/87.4	64.5/87.9
	CIFAR-10 (**n**)	59.2/92.9	59.4/92.5	61.1/90.9	60.6/91.4	61.3/91.1	60.8/91.6
	Texture (**f**)	69.8/73.6	69.6/72.4	70.9/69.6	70.7/70.1	71.0/69.2	70.8/69.5
	SVHN (**f**)	72.2/68.5	71.9/66.1	73.2/63.1	72.4/64.5	73.4/63.5	73.0/64.1
	Places365 (**f**)	70.4/70.3	70.8/68.2	71.4/67.2	70.9/68.3	71.5/67.0	71.1/67.7
ImageNet-1k	NINCO (**n**)	57.4/80.6	57.7/80.8	59.6/79.2	58.5/80.3	59.9/78.9	59.2/79.5
	SSB-hard (**n**)	55.9/82.7	55.6/82.4	56.3/82.7	55.9/83.0	57.2/81.0	57.0/81.2
	iNaturalist (**f**)	62.5/76.9	62.9/75.4	63.2/74.5	62.7/75.1	63.1/74.9	62.7/75.8
	Texture (**f**)	63.8/73.4	64.0/72.7	64.4/71.3	63.2/72.6	64.0/71.5	63.5/72.6
	OpenImage-O (**f**)	64.2/74.8	64.5/73.2	64.1/73.8	63.8/74.5	64.4/73.5	64.1/74.2

ID dataset	OOD dataset	Unsupervised methods					
		Diffusion model	GAN	VAE			
		OAL	sGAN	DP	mVAE	DB	**Ours**
CIFAR-100	Tiny-ImageNet (**n**)	63.6/89.0	63.9/88.2	65.9/86.3	63.1/88.2	58.0/89.5	**67.3**$_{\pm 0.1}$/**85.1**$_{\pm 0.1}$
	CIFAR-10 (**n**)	59.2/93.3	59.0/92.1	62.0/90.6	59.1/92.5	54.0/93.6	**63.4**$_{\pm 0.2}$/**89.2**$_{\pm 0.2}$
	Texture (**f**)	70.5/70.8	70.1/71.8	71.9/68.6	65.7/71.5	62.1/74.6	**73.1**$_{\pm 0.3}$/**66.0**$_{\pm 0.2}$
	SVHN (**f**)	72.1/64.3	72.6/65.0	74.0/61.8	68.2/65.7	63.7/68.8	**75.5**$_{\pm 0.4}$/**59.4**$_{\pm 0.2}$
	Places365 (**f**)	70.4/68.6	70.7/69.2	71.9/65.5	66.3/69.4	62.5/72.9	**73.2**$_{\pm 0.2}$/**63.8**$_{\pm 0.3}$
ImageNet-1k	NINCO (**n**)	58.1/81.0	57.8/80.1	60.8/78.5	59.2/80.9	57.0/86.4	**62.0**$_{\pm 0.2}$/**76.9**$_{\pm 0.4}$
	SSB-hard (**n**)	55.3/84.4	54.2/83.6	58.1/80.5	56.5/83.7	54.3/89.6	**59.6**$_{\pm 0.1}$/**79.1**$_{\pm 0.2}$
	iNaturalist (**f**)	62.2/76.5	61.9/78.0	64.0/73.8	64.0/86.6	60.6/83.4	**65.2**$_{\pm 0.3}$/**71.7**$_{\pm 0.3}$
	Texture (**f**)	63.8/73.2	63.4/73.9	65.2/70.4	64.8/83.6	62.2/78.8	**66.5**$_{\pm 0.2}$/**68.2**$_{\pm 0.1}$
	OpenImage-O (**f**)	63.5/75.9	63.0/77.7	65.6/72.2	65.5/85.5	62.3/82.6	**66.9**$_{\pm 0.1}$/**70.3**$_{\pm 0.3}$

Results on synthetic negative sample generation. We evaluate the effectiveness of the synthetic negative sample generation in our method by comparing it with other popular negative sample generation techniques for OOD detection tasks, including CSI [36], CnC [16], VOS [12], NPOS [37], and HamOS [21]. To be specific, the performance comparison is done by replacing the second stage of our method with those techniques while Stages 1 and 3 remain unchanged, as shown in Table 2. It reveals that our method is the most effective technique for generating synthetic negative samples for OOD detection in the TinyML setup.

Inference efficiency and relative inference model size. Table 3 reports inference latency and relative inference model size on a real TinyML device for each method's *raw* (unshrunk) model. Note that for retrieval-based methods such as NNGuide, the stored nearest-neighbor feature bank is excluded from the relative model size, as it consists of cached features rather than model parameters. However, the cost of traversing this bank at inference is reflected in latency, which explains why such methods can exhibit a moderate model size yet a disproportionately high latency. Our method is significantly faster than competing methods and has the smallest footprint among competing OOD detection methods.

Table 2: AUROC (↑) scores for our method and other negative sample generation methods on (**n**)ear and (**f**)ar-OOD test cases for CIFAR-100 and ImageNet-1k benchmarks. This is done by replacing the second stage of our method with those techniques.

ID dataset	OOD dataset	Negative sample generation methods					
		CSI	CnC	VOS	NPOS	HamOS	Ours
CIFAR-100	Tiny-ImageNet (**n**)	63.8±0.2	63.2±0.3	65.0±0.2	65.6±0.3	66.0±0.2	**67.3**±0.1
	CIFAR-10 (**n**)	61.5±0.4	60.2±0.2	60.9±0.1	61.7±0.2	62.1±0.1	**63.4**±0.2
	Texture (**f**)	67.3±0.3	65.9±0.5	68.4±0.6	70.4±0.5	71.2±0.2	**73.1**±0.3
	SVHN (**f**)	72.2±0.4	71.6±0.4	73.1±0.7	73.4±0.2	74.0±0.2	**75.5**±0.4
	Places365 (**f**)	72.0±0.2	71.4±0.5	71.7±0.3	71.9±0.1	72.3±0.1	**73.2**±0.2
ImageNet-1k	NINCO (**n**)	53.4±0.2	52.3±0.2	59.5±0.4	59.9±0.3	60.5±0.2	**62.0**±0.2
	SSB-hard (**n**)	57.0±0.3	55.5±0.1	56.8±0.2	57.8±0.4	58.3±0.2	**59.6**±0.1
	iNaturalist (**f**)	56.1±0.2	55.0±0.6	62.4±0.4	63.0±0.5	63.8±0.1	**65.2**±0.3
	Texture (**f**)	64.8±0.3	64.2±0.2	64.9±0.5	65.2±0.3	65.1±0.2	**66.5**±0.2
	OpenImage-O (**f**)	62.7±0.4	63.1±0.4	63.5±0.2	64.6±0.4	65.2±0.3	**66.9**±0.1

Table 3: Inference efficiency and relative model size evaluated on ImageNet-1k benchmark based on a real TinyML device, using each method's *raw* (unshrunk) model. Latency is reported as the mean over 200 iterations and relative inference model size divides each method's inference-phase parameter count by our proposed method.

	Supervised		Diffusion	GAN	VAE-based (unsupervised)			
	NNGuide	TinyOOD	OAL	sGAN	DP	mVAE	DB	**Ours**
Latency (ms) ↓	292.53	1.52	29.21	18.26	2.66	1.75	1.27	**0.12**
Rel. model size ↓	1.5	1.4	5.1	4.2	1.5	3.5	3.5	**1.0**

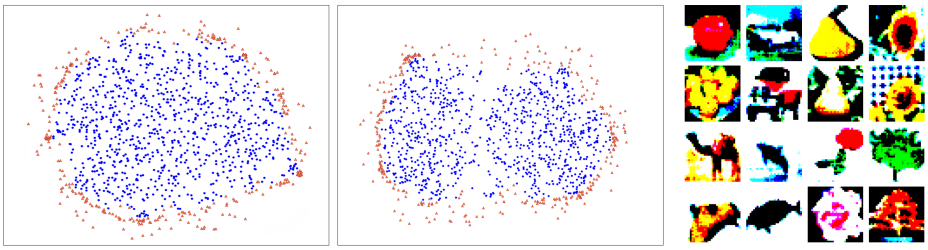
This joint advantage in speed and size holds across all paradigms, indicating that the gain is algorithmic rather than a by-product of backbone shrinkage.

Visualization of synthetic negative samples. Figure 5 visualizes the synthetic negative samples generated by our method using Uniform Manifold Approximation and Projection (UMAP) in the VAE latent space (**(a)**–**(b)**) and pixel space (**(c)**). In the latent space (**(a)**–**(b)**), ID samples form a compact cluster, while the synthetic negative samples produced by our method concentrate near the boundary of this cluster yet remain outside the ID distribution, acting as "hard negative samples". In pixel space (**(c)**), the synthetic negative samples exhibit structured, semantically meaningful visual distortions to the original ID images.

4.3 Ablation Studies on Real TinyML Device

We present the results of ablation studies on the number of pixel-intensity perturbed variants and hard negative samples in Tables 4 and 5, respectively.

Number of pixel-intensity perturbed variants without negative samples. In Table 4, we present an ablation study on the number of pixel-intensity variants $\tilde{x}$ *without* negative samples. It shows that pixel-intensity bias significantly impacts OOD detection performance. Without any pixel-intensity variants $\tilde{x}$ for bias correction, the performance declines drastically for most of the test cases. But



(a) UMAP for latent space, ID: CIFAR-100 **(b)** UMAP for latent space, ID: ImageNet-1k **(c)** Pixel-space, ID: CIFAR-100

Fig. 5: Visualization of our synthetic negative samples. **(a) and (b)**: UMAP projections of a subset of ID samples and our synthetic negative samples in the VAE latent space with CIFAR-100 and ImageNet-1k as the ID dataset, respectively. Blue points denote ID samples and brown points denote the corresponding synthetic negatives. **(c)**: Pixel-space examples of our synthetic negative samples with CIFAR-100 as ID dataset.

Table 4: Ablation study on the number of pixel-intensity perturbed variants $\tilde{x}$ to OOD detection performance, AUROC scores (↑), (*without* using negative samples) on (**n**)ear and (**f**)ar-OOD test cases for CIFAR-100 and ImageNet-1k benchmarks. Because *no* negative samples are used for training, DB becomes the baseline for ours.

ID dataset	**OOD dataset**	Number of pixel-intensity perturbed variants $\tilde{x}$				
		DB	w/o $\tilde{x}$	4 $\tilde{x}$	16 $\tilde{x}$	32 $\tilde{x}$
CIFAR-100	Tiny-ImageNet (**n**)	58.0	47.5	51.9	55.3	58.0
	CIFAR-10 (**n**)	54.0	45.5	49.6	52.1	54.0
	Texture (**f**)	62.1	49.4	53.5	57.6	62.1
	SVHN (**f**)	63.7	52.7	56.9	61.1	63.7
	Places365 (**f**)	62.5	50.3	56.0	60.4	62.5
ImageNet-1k	NINCO (**n**)	57.0	43.5	47.4	52.2	57.0
	SSB-hard (**n**)	54.3	44.7	48.8	51.0	54.3
	iNaturalist (**f**)	60.6	46.9	51.1	56.4	60.6
	Texture (**f**)	62.2	49.2	55.6	58.9	62.2
	OpenImage-O (**f**)	62.3	52.3	56.6	59.4	62.3

as the number of pixel-intensity variants increases, performance incrementally rebounds, eventually achieving almost the same performance as the DB.

Generation of hard negative samples. In our proposed method, generating "high-quality" negative samples is a pivotal phase in training Stage 2, which includes three steps (also, see the second sub-figure of Figure 2a). Table 5 shows that each step for the synthetic negative sample generation contributes to an improvement in OOD detection performance. Step 1 is effective in creating initial negative samples; Step 2 iteratively enhances the quality of those samples; and Step 3 serves to select valid and hard negative samples.

4.4 Experiments on Standard GPU

We also conduct experiments on standard GPU platforms to facilitate reproducibility and ensure fair comparisons with competing OOD detection methods,

Table 5: Ablation study on the three steps to generate negative samples on (**n**)ear and (**f**)ar-OOD test cases for CIFAR-100 and ImageNet-1k benchmarks, AUROC scores (↑). The "Base" is the model using 32 $\tilde{x}$ but does not utilize any negative samples. "s1" and "s2" denote steps 1 and 2 of stage 2, respectively. "Ours" is with all three steps.

ID dataset	**OOD dataset**	Steps of stage 2			
		Base	Base + s1	Base + s1 + s2	Ours
CIFAR-100	Tiny-ImageNet (**n**)	58.0	61.4	64.8	67.3
	CIFAR-10 (**n**)	54.0	58.6	61.7	63.4
	Texture (**f**)	62.1	65.9	70.2	73.1
	SVHN (**f**)	63.7	67.6	71.4	75.5
	Places365 (**f**)	62.5	65.3	69.1	73.2
ImageNet-1k	NINCO (**n**)	57.0	58.5	60.2	62.0
	SSB-hard (**n**)	54.3	56.0	58.1	59.6
	iNaturalist (**f**)	60.6	62.4	63.8	65.2
	Texture (**f**)	62.2	63.7	65.0	66.5
	OpenImage-O (**f**)	62.3	63.6	65.2	66.9

which originally reported their results on GPU infrastructure. In these experiments, inference time is matched to our method on the GPU. Due to architectural differences in compute-to-memory ratios between GPUs and TinyML devices, competing methods require even greater scaling down to match the inference speed of our approach on the GPU. As a result, our method exhibits an even larger performance advantage in GPU experiments than in the TinyML setting, establishing a new SOTA with a wider margin. Full details of the experimental setup and the results are provided in Appendix C.4.

5 Conclusions

Integration of OOD detection into TinyML devices demands both low inference complexity and strong detection accuracy. Addressing this, we made two innovations: (i) a latent space de-biasing technique that eliminates pixel-intensity bias during training and (ii) a novel strategy for generating high-quality synthetic negative samples. On both the real TinyML device and standard GPU infrastructure, extensive validation demonstrates that our method outperforms competing OOD detection methods when adhering to TinyML's inference constraints. By resolving the inherent trade-off between speed and accuracy, our work establishes a framework for low-latency OOD detection models in TinyML applications.

Acknowledgements

This research has been funded by the Industrial Technology Innovation Program [P0030255, Project OptiBatt - AI: Digital Twin-Integrated AI Agent for Manufacturing of EV Battery Systems – Cooling, High Voltage, and ICB Modules] of the Ministry of Trade, Industry and Energy of the Republic of Korea. This work was also supported by the Digital Research Alliance of Canada through the Researcher Resource Grant (RRG) 5193 (2025).

References

1. Abu-Samah, A., Ghaffa, D., Abdullah, N.F., Kamal, N., Nordin, R., Dela Cruz, J.C., Magwili, G.V., Mercado, R.J.: Deployment of TinyML-based stress classification using computational constrained health wearable. Electronics **14**(4), 687 (2025). https://doi.org/10.3390/electronics14040687
2. Ataeiasad, F., Elizondo, D., Calderón Ramírez, S., Greenfield, S., Deka, L.: Out-of-distribution detection with memory-augmented variational autoencoder. Mathematics **12**(19), 3153 (2024). https://doi.org/10.3390/math12193153
3. Banbury, C., Reddi, V.J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., Montino, P., Kanter, D., Ahmed, S., Pau, D., Thakker, U., Torrini, A., Warden, P., Cordaro, J., Di Guglielmo, G., Duarte, J., Gibellini, S., Parekh, V., Tran, H., Tran, N., Niu, W., Xu, X.: MLPerf tiny benchmark. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (2021)
4. Bartoli, P., Veronesi, C., Giudici, A., Siorpaes, D., Trojaniello, D., Zappa, F.: Benchmarking energy and latency in TinyML: A novel method for resource-constrained AI. In: 2025 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2025). https://doi.org/10.1109/IJCNN64981.2025.11228997
5. Caetano, F., Viviers, C., Zavala-Mondragón, L.A., De With, P.H., van der Sommen, F.: DisCoPatch: Taming adversarially-driven batch statistics for improved out-of-distribution detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 2898–2908 (2025). https://doi.org/10.1109/ICCV51701.2025.00278
6. Cao, C., Zhong, Z., Zhou, Z., Liu, Y., Liu, T., Han, B.: Envisioning outlier exposure by large language models for out-of-distribution detection. In: Proceedings of the 41st International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 235, pp. 5629–5659 (2024). https://doi.org/10.5555/3692070.3692289
7. Cao, Z., Wu, X., Wu, C., Jiao, S., Xiao, Y., Zhang, Y., Zhou, Y.: KeypointNet: An efficient deep learning model with multi-view recognition capability for sitting posture recognition. Electronics **14**(4), 718 (2025). https://doi.org/10.3390/electronics14040718
8. Chauhan, K., U, B.M., Shenoy, P., Gupta, M., Sridharan, D.: Robust outlier detection by de-biasing VAE likelihoods. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9881–9890 (2022). https://doi.org/10.1109/CVPR52688.2022.00965
9. Ciccone, F., Ceruti, A.: Real-time search and rescue with drones: A deep learning approach for small-object detection based on YOLO. Drones **9**(8), 514 (2025). https://doi.org/10.3390/drones9080514
10. Djurisic, A., Bozanic, N., Ashok, A., Liu, R.: Extremely simple activation shaping for out-of-distribution detection. In: Proceedings of the International Conference on Learning Representations (ICLR) (2023)
11. Dong, X., Guo, J., Li, A., Ting, W.T., Liu, C., Kung, H.: Neural mean discrepancy for efficient out-of-distribution detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 19217–19227 (2022). https://doi.org/10.1109/CVPR52688.2022.01862
12. Du, X., Wang, Z., Cai, M., Li, Y.: VOS: Learning what you don't know by virtual outlier synthesis. In: Proceedings of the International Conference on Learning Representations (ICLR) (2022)

13. El Zeinaty, C., Hamidouche, W., Herrou, G., Menard, D.: Designing object detection models for TinyML: Foundations, comparative analysis, challenges, and emerging solutions. ACM Computing Surveys **58**(2), 50:1–50:48 (2026). https://doi.org/10.1145/3744339
14. Gao, H., Li, J.: Enhancing OOD detection using latent diffusion. In: Proceedings of the 2025 International Conference on Multimedia Retrieval (ICMR). pp. 358–367. ACM (2025). https://doi.org/10.1145/3731715.3733326
15. Ghanathe, N.P., Wilton, S.J.: QUTE: Quantifying uncertainty in TinyML models with early-exit-assisted ensembles for model-monitoring. In: Proceedings of the Forty-second International Conference on Machine Learning (ICML). vol. 267, pp. 19286–19306. PMLR (2025). https://doi.org/10.5555/3780338.3781087
16. Hebbalaguppe, R., Ghosal, S.S., Prakash, J., Khadilkar, H., Arora, C.: A novel data augmentation technique for out-of-distribution sample detection using compounded corruptions. In: Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2022. Lecture Notes in Computer Science, vol. 13715, pp. 529–545. Springer (2023). https://doi.org/10.1007/978-3-031-26409-2_32
17. Heydari, S., Mahmoud, Q.H.: Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions. Sensors **25**(10), 3191 (2025). https://doi.org/10.3390/s25103191
18. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for MobileNetV3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1314–1324 (2019). https://doi.org/10.1109/ICCV.2019.00140
19. Incel, O.D., Bursa, S.Ö.: On-device deep learning for mobile and wearable sensing applications: A review. IEEE Sensors Journal **23**(6), 5501–5512 (2023). https://doi.org/10.1109/JSEN.2023.3240854
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Proceedings of the International Conference on Learning Representations (ICLR) (2014)
21. Li, H., Zhang, T.: Outlier synthesis via Hamiltonian Monte Carlo for out-of-distribution detection. In: Proceedings of the International Conference on Learning Representations (ICLR) (2025)
22. Li, Y., Xiong, B., Chen, G., Chen, Y.: SeTAR: Out-of-distribution detection with selective low-rank approximation. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS). vol. 37, pp. 72840–72871 (2024). https://doi.org/10.52202/079017-2319
23. Li, Y., Jia, J., Zuo, Y., Zhu, W.: TinyOOD: effective out-of-distribution detection for TinyML. In: Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1–5. IEEE (2023). https://doi.org/10.1109/ICASSP49357.2023.10094746
24. Li, Y., Chen, Y., Dai, X., Chen, D., Liu, M., Yuan, L., Liu, Z., Zhang, L., Vasconcelos, N.: MicroNet: Improving image recognition with extremely low FLOPs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 468–477 (2021). https://doi.org/10.1109/ICCV48922.2021.00052
25. Liu, L., Qin, Y.: Detecting out-of-distribution through the lens of neural collapse. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15424–15433 (2025). https://doi.org/10.1109/CVPR52734.2025.01437
26. Liu, W., Wang, X., Owens, J., Li, Y.: Energy-based out-of-distribution detection. Advances in Neural Information Processing Systems (NeurIPS) **33**, 21464–21475 (2020). https://doi.org/10.5555/3495724.3497526

27. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 116–131 (2018). https://doi.org/10.1007/978-3-030-01264-9_8
28. Mahmoudi, S.A., Gloesener, M., Benkedadra, M., Lerat, J.S.: Edge AI system for real-time and explainable forest fire detection using compressed deep learning models. In: Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Volume 3: VISAPP. pp. 847–854 (2025). https://doi.org/10.5220/0013382500003912
29. Ming, Y., Cai, Z., Gu, J., Sun, Y., Li, W., Li, Y.: Delving into out-of-distribution detection with vision-language representations. Advances in Neural Information Processing Systems (NeurIPS) **35**, 35087–35102 (2022). https://doi.org/10.52202/068431-2543
30. Ning, Z., Vandersteegen, M., Van Beeck, K., Goedemé, T., Vandewalle, P.: Power consumption benchmark for embedded AI inference. In: Proceedings of the International Conferences on Applied Computing 2024 and WWW/Internet 2024. pp. 355–360 (2024). https://doi.org/10.33965/ac_icwi_2024_202407c003
31. Park, J., Jung, Y.G., Teoh, A.B.J.: Nearest neighbor guidance for out-of-distribution detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1686–1695 (2023). https://doi.org/10.1109/ICCV51070.2023.00162
32. Ragusa, E., Dosen, S., Zunino, R., Gastaldo, P.: Affordance segmentation using tiny networks for sensing systems in wearable robotic devices. IEEE Sensors Journal **23**(19), 23916–23926 (2023). https://doi.org/10.1109/JSEN.2023.3308615
33. Sinha, S., Dieng, A.B.: Consistency regularization for variational auto-encoders. Advances in Neural Information Processing Systems (NeurIPS) **34**, 12943–12954 (2021). https://doi.org/10.5555/3540261.3541252
34. Subramanyam, R., Narayanaswamy, V., Naufel, M., Spanias, A., Thiagarajan, J.J.: Improved StyleGAN-v2 based inversion for out-of-distribution images. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 162, pp. 20625–20639. PMLR (2022)
35. Sun, Y., Guo, C., Li, Y.: ReAct: Out-of-distribution detection with rectified activations. Advances in Neural Information Processing Systems (NeurIPS) **34**, 144–157 (2021). https://doi.org/10.5555/3540261.3540273
36. Tack, J., Mo, S., Jeong, J., Shin, J.: CSI: Novelty detection via contrastive learning on distributionally shifted instances. Advances in Neural Information Processing Systems (NeurIPS) **33**, 11839–11852 (2020). https://doi.org/10.5555/3495724.3496717
37. Tao, L., Du, X., Zhu, J., Li, Y.: Non-parametric outlier synthesis. In: Proceedings of the International Conference on Learning Representations (ICLR) (2023)
38. Vasu, P.K.A., Pouransari, H., Faghri, F., Vemulapalli, R., Tuzel, O.: MobileCLIP: Fast image-text models through multi-modal reinforced training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15963–15974 (2024). https://doi.org/10.1109/CVPR52733.2024.01511
39. Wu, K., Peng, H., Zhou, Z., Xiao, B., Liu, M., Yuan, L., Xuan, H., Valenzuela, M., Chen, X.S., Wang, X., Chao, H., Hu, H.: TinyCLIP: CLIP distillation via affinity mimicking and weight inheritance. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 21970–21980 (2023). https://doi.org/10.1109/ICCV51070.2023.02008

40. Xiao, Z., Yan, Q., Amit, Y.: Likelihood regret: An out-of-distribution detection score for variational auto-encoder. Advances in Neural Information Processing Systems (NeurIPS) **33**, 20685–20696 (2020). https://doi.org/10.5555/3495724.3497461
41. Yin, H., Molchanov, P., Alvarez, J.M., Li, Z., Mallya, A., Hoiem, D., Jha, N.K., Kautz, J.: Dreaming to distill: Data-free knowledge transfer via DeepInversion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8715–8724 (2020). https://doi.org/10.1109/CVPR42600.2020.00874
42. Zeng, Z., Liu, B.: Unsupervised out-of-distribution detection by restoring lossy inputs with variational autoencoder. arXiv preprint arXiv:2309.02084 (2023). https://doi.org/10.48550/arXiv.2309.02084
43. Zhang, J., Yang, J., Wang, P., Wang, H., Lin, Y., Zhang, H., Sun, Y., Du, X., Li, Y., Liu, Z., Chen, Y., Li, H.: OpenOOD v1.5: Enhanced benchmark for out-of-distribution detection. Journal of Data-centric Machine Learning Research **2**(3), 1–32 (2024)
44. Zhang, Y., Suda, N., Lai, L., Chandra, V.: Hello edge: Keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128 (2017). https://doi.org/10.48550/arXiv.1711.07128