

Improving the Energy Efficiency and Robustness of tinyML Computer Vision using Log-Gradient Input Images

Qianyun Lu
Stanford University
Stanford, United States
savylu@stanford.edu

Boris Murmann
Stanford University
Stanford, United States
murmann@stanford.edu

ABSTRACT

This paper studies the merits of applying log-gradient input images to convolutional neural networks (CNNs) for tinyML computer vision (CV). We show that log gradients enable: (i) aggressive 1.5-bit quantization of first-layer inputs, (ii) potential CNN resource reductions, and (iii) inherent robustness to illumination changes (1.7% accuracy loss across $2^{-5} \dots 2^3$ brightness variation vs. up to 10% for JPEG). We establish these results using the PASCAL RAW image data set and through a combination of experiments using neural architecture search and a fixed three-layer network. The latter reveal that training on log-gradient images leads to higher filter similarity, making the CNN more prunable. The combined benefits of aggressive first-layer quantization, CNN resource reductions, and operation without tight exposure control and image signal processing (ISP) are helpful for pushing tinyML CV toward its ultimate efficiency limits.

KEYWORDS

Computer vision pipeline, filter similarity, illumination invariance, image signal processor, log gradients, neural network quantization, sensor datasets

ACM Reference Format:

Qianyun Lu and Boris Murmann. 2022. Improving the Energy Efficiency and Robustness of tinyML Computer Vision using Log-Gradient Input Images. In *Proceedings of tinyML Research Symposium (tinyML Research Symposium'22)*. ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Fueled by advancements in machine learning algorithms and hardware, computer vision has extended its reach to battery powered Internet of Things (IoT) devices, enabling a variety of new applications that thrive on near-sensor data analytics [22]. However, to further expand the commercial adoption of this technology, both the cost and energy consumption must continue to improve steadily. Our work focuses on this need through the lens of holistic hardware optimization, considering the computer vision (CV) pipeline from the physical image sensor to the classifier output. As shown in Figure 1, the traditional CV pipeline consists of three parts: the image sensor, the image signal processor (ISP), and task-specific algorithms such as convolutional neural networks (CNNs) [27]. The

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

tinyML Research Symposium'22, March 2022, San Jose, CA
© 2022 Copyright held by the owner/author(s).

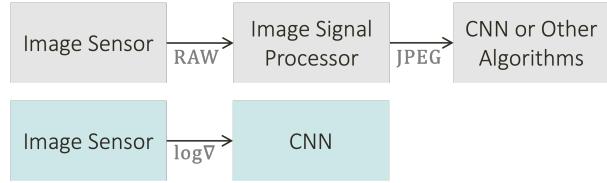


Figure 1: Conventional (top) and our $\log \nabla$ computer vision pipeline (bottom).

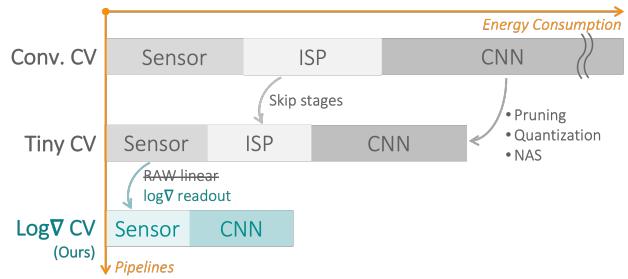


Figure 2: Energy breakdown of CV pipelines.

image sensor produces raw images that are not visually appealing (to humans) and hence the ISP performs a variety of operations such as white balancing and gamma correction. While prior work has shown that a subset of ISP operations can be omitted for machine consumption [5, 10, 17], our approach omits the digital ISP block entirely and instead delegates a small amount of preprocessing to the analog circuitry of the image sensor. Specifically, we explore the use of log gradients ($\log \nabla$) as a direct input to CNNs. A key feature of $\log \nabla$ is that they convey a normalized, illumination-invariant representation of the characteristic edges in the input image. The feasibility and energy efficiency of extracting log gradients within the image sensor has already been established in [26], but this investigation was based on algorithms outside the modern deep learning framework. Our main contribution is to examine the merits of log gradients for CNN-based systems.

Figure 2 illustrates the motivation for our work from an energy perspective. In the shift from conventional CV pipelines to a solution optimized for tiny machine learning (tinyML), most energy savings have come from the remarkable progress in CNN compression [8]. However, as these compression techniques continue to yield smaller CNN footprints, the rest of the pipeline becomes important and can no longer be ignored. For context, we note

that optimized hardware for small CV CNNs operates at around 3-6 nJ/pixel [3, 13]. This number is expected to decrease significantly given the large community working on CNN optimizations across the software-algorithm-hardware stack [22]. Depending on their specs, image sensors consume about 0.2-2 nJ/pixel [12], while high-end ISP macros consume 1-2 nJ/pixel [14] in the latest CMOS technology. While skipping ISP stages can reduce the latter significantly, we show that the **benefits of a log-gradient approach is not merely limited to ISP energy savings but extends to potential improvements in robustness and CNN compressibility.**

The following sections expand on the contributions of this work, which can be summarized as follows:

- We formulate the concept of employing a $\log \nabla$ sensor readout within a tinyML CV pipeline and discuss data set aspects.
- We use neural network architecture search (NAS) to show that a CNN with $\log \nabla$ inputs becomes more compressible (Section 3.1). We offer some intuition for this behavior using filter similarity and validate this hypothesis using experiments with a fixed 3-layer toy CNN (Section 3.2).
- We show that the $\log \nabla$ approach enables aggressive quantization of the CNN’s first layer inputs down to 1.5 bits (3 levels). This not only reduces data movement between the image sensor and the CNN, but also helps with memory requirements. The quantization of CNN weights and activations is not considered here and remains as future work.
- We perform experiments to assess the robustness to simulated illumination changes for $\log \nabla$, RAW and JPEG inputs. We observe only 1.7% accuracy loss for $\log \nabla$ with 8x change in brightness versus up to 10% for JPEG. Robustness aspects are often neglected in the CNN compression literature but are becoming increasingly important as real-word applications proliferate.

2 LOG ∇ CV PIPELINE

2.1 Log Gradient Computation

Denote an image by $P \in \mathbb{R}^{H \times W}$, where H and W are the image height and width in pixels. We compute the $\log \nabla$ of an image as:

$$P' = \log(P + 1) \quad (1)$$

$$\log \nabla = P' * f \quad \text{where} \quad f = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2)$$

where the “+1” is needed to shift the input into the domain of the log, “*” means 2D-convolution, and f is a gradient filter that extracts and combines horizontal and vertical gradients. For RGB data, the gradients can be computed channel-wise, but we consider only grayscale images in the present work. As shown in [26], the gradient computations can be performed efficiently within the analog readout circuitry of an otherwise standard image sensor. Though this aspect is not the focus of this paper, we provide a brief review here. Consider for example the difference between two logarithms. We can write:

$$d = \log(a) - \log(b) = \log(a/b) \approx Q(a/b) \quad (3)$$

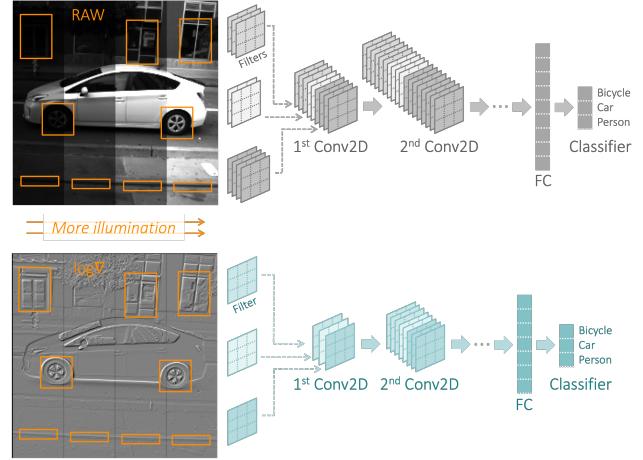


Figure 3: Illumination invariance of $\log \nabla$ and intuition about Conv2D filter savings.

where Q represents a quantizer with log-spaced decision levels. The work of [26] thus uses a ratio-to-digital converter (RDC) instead of a conventional analog-to-digital converter (ADC).

An important aspect that we will examine below is that the gradients (and hence the ratios) can be coarsely quantized, making the log distortion of only a few decision levels a relatively simple task. In absence of the specialized RDC readout used in [26], it can be emulated in hardware using a standard image sensor readout with relatively simple, equivalent digital postprocessing operations (no logarithms needed).

2.2 Log Gradient Intuition

To gain intuition about the potential benefits of log gradients, consider the contrived example in Figure 3. The RAW image (top) contains four illumination levels while the computed $\log \nabla$ (bottom) image strips these away. The reason for this is that the difference of logarithms corresponds to ratios, and the pixel ratios in the top picture are not affected by the illumination changes. To keep this discussion concise, we ignore various second-order effects related to noise and saturation; the interested reader can refer to [26].

Given the normalizing behavior of log gradients, we expect to see three benefits. First, while the $\log \nabla$ representation is not visually appealing, it should help in making the CNN classification more robust to global illumination changes. In a tinyML application, this may help relax exposure control requirements. Second, since the $\log \nabla$ operation reduces the image’s dynamic range, it should be more amenable to aggressive quantization. Both of these properties have already been established in [26] but will be reiterated here for CNN-based applications. Third, since CNNs basically perform pattern matching using Conv2D filters, the RAW image should need a larger variety of (scaled) filters to detect the same object under different illuminations. In contrast, $\log \nabla$ should only need one filter for each characteristic pattern, hence enabling a smaller model size (see Figure 3).

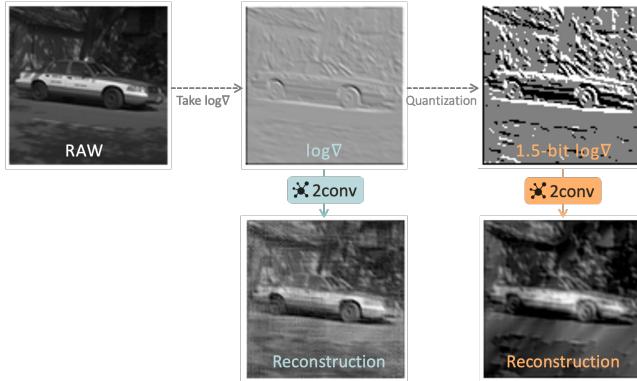


Figure 4: Image reconstruction from $\log \nabla$ using 2conv.

To provide a feel for the losses introduced by the $\log \nabla$ representation, we reconstructed an original RAW image from its log gradients using a two-layer toy CNN (2conv). Figure 4 shows the outcomes without quantization and for quantized 1.5-bit (3-level) $\log \nabla$. Despite the smudges, we are able to recognize the car in both cases, indicating that the $\log \nabla$ preserve most of the useful information even with aggressive quantization. Note that this example is for illustrative purposes only (it would make no sense to reconstruct the original image in the end application). Details of the training and inference setups for this toy example are provided in Appendix A.

2.3 Datasets

Experimentation with a $\log \nabla$ pipeline requires RAW image data. Unfortunately, most public datasets consist of post-ISP 8-bit JPEG images, which are obtained through irreversible and lossy transformations. An exception is the PASCAL RAW 2014 dataset [19], which contains physical image sensor data for three object types: bicycle, car, and person. Our experiments in Section 3 use images from this dataset that were cropped according to classification bounding boxes. This set includes 6550 images total (bicycle : car : person = 708 : 1765 : 4077). All images are 16-bit grayscale after demosaicing. To prove that our approach is not overly sensitive to the choice of dataset, we performed ancillary experiments using approximate RAW data reconstructed from the popular Visual Wake Words (VWW) dataset (see Appendix B).

3 EXPERIMENTS

To demonstrate the advantages of the $\log \nabla$ approach, we performed experiments using several formats for the first-layer CNN-inputs:

$$\mathcal{I} = \{ \quad \text{8-bit JPEG}, \quad \text{16-bit RAW}, \\ \quad \text{FP } \log \nabla, \quad \text{1.5-bit } \log \nabla, \quad \text{2.25-bit } \log \nabla \quad \}.$$

Here, 16-bit RAW corresponds to demosaiced grayscale images from the PASCAL RAW 2014 dataset. Full precision floating point (FP) $\log \nabla$ are log gradients computed from these raw data without explicit quantization. The labels 1.5-bit and 2.25-bit denote 3- and 5-level representations that were quantized using empirical thresholds (the threshold values are not critically sensitive, but can potentially be tuned during training). We generated the data

labelled "8-bit JPEG" from the demosaiced RAW images using only gamma correction ($\gamma = 2.2$). We found that using additional ISP stages made only insignificant differences in our experiments with small CNNs, which is in line with the conclusions of [5]. All experiments focus on image classification which can be further extended to downstream tasks like transfer learning and object detection.

Due to the lack of explainability of ML algorithms, we study the impact of $\log \nabla$ inputs from two different perspectives. First, we use NAS to show that $\log \nabla$ lead to lower CNN resource requirements. Next, we study the impact of $\log \nabla$ on a 3-layer toy network. We observe higher filter similarity and thus higher prunability with $\log \nabla$ inputs, which is consistent with the NAS results.

3.1 CNN architecture search

We employ the μ NAS algorithm [15], which considers the available size of RAM, persistent storage and processor speed to derive bounds on peak memory usage, model size, and latency (approximated by the number of multiply-accumulate operations (MACs)). Aging evolution is used as the main search algorithm and can be combined with structural pruning. The search space has high granularity and few restrictions on layer connectivity. For details of uNAS' search space, refer to Table 1 in [15]. All CNN-internal weights and activations are kept in floating point precision to cleanly isolate the impact of varying the input data format. Further details of the training setup are provided in Appendix C.1.

Figure 5 shows the resource versus error rate Pareto fronts when μ NAS is fed with three different image types. Relative to the 8-bit JPEG baseline case, we see that going to 16-bit RAW leads to a significant increase in the required resources for a given error rate. This is consistent with the benefits of ISP observed in [10]. However, when we input FP $\log \nabla$, we recover these losses and in fact outperform the JPEG baseline case. For example, for an error rate of 2%, the model size, MACs and peak memory usage of FP $\log \nabla$ area about 5x, 5x, 1.1x smaller than for JPEG.

Since FP $\log \nabla$ would be too costly to compute in hardware, we repeat the experiment using the quantized 1.5-bit and 2.25-bit log gradient inputs (see Figure 6). As expected, there is some accuracy loss with quantization, but the degradations are modest and still in line with or slightly better than the JPEG baseline. We note that there is room for further experimentation with different hyperparameters as well as quantized CNN weights and activations for a more exact assessment of the required resources. The main conclusion from this experiment is that coarsely quantized $\log \nabla$ appear to be well suited as inputs to small, NAS-optimized CNNs.

3.2 Fixed CNN architectures

Section 2.2 argued that $\log \nabla$ inputs should require fewer Conv2D filters than RAW data for classification. If this is true, we should observe a change in filter redundancy if we keep the CNN architecture fixed and vary the input $I \in \mathcal{I}$. Thus, the experiments in this section employ a toy CNN with two convolutional and one fully connected layer (2conv1fc, see Figure 7), using a kernel size of 5 and channel counts of $c_1 = 150$, $c_2 = 5$. Further details of the training setup are provided in Appendix C.2.

Higher filter redundancy means that there is a higher degree of similarity among the filters. To investigate, we thus compute the

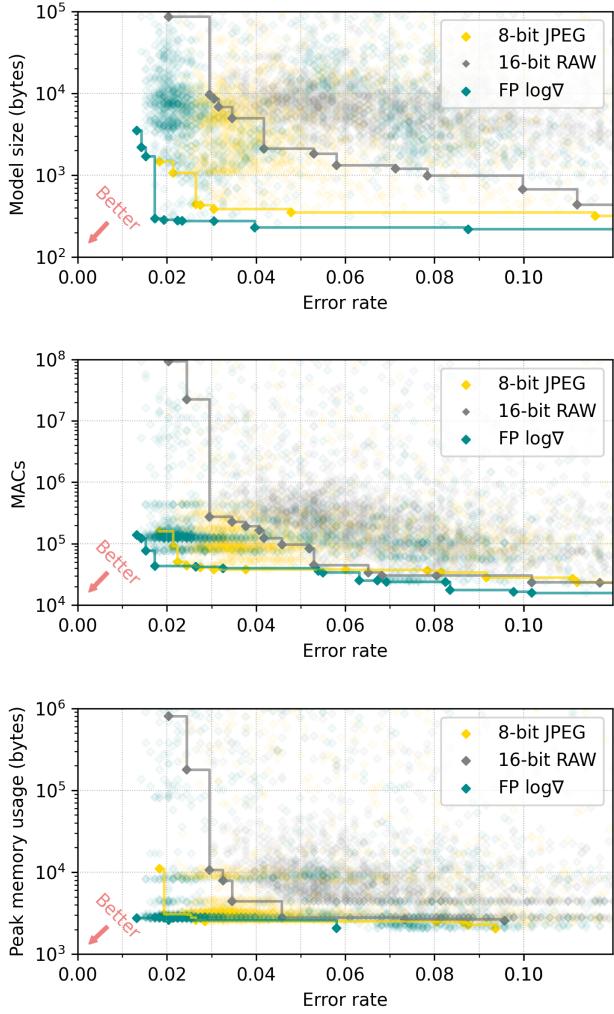


Figure 5: CNN resources vs. classification error rate for high-resolution inputs.

layer-wise cosine similarities among CNN filters [20] after training:

$$s_{i,j} = \mathbf{w}'_i \cdot \mathbf{w}'_j \quad (4)$$

where \mathbf{w}'_i and \mathbf{w}'_j are normalized filters in the same Conv2D layer.

Figure 8 shows histograms of filter similarities for a few representative cases. We observe that FP and 1.5-bit log-∇ models have heavier tails around $s \approx 1$ when compared to the CNN trained on 16-bit RAW, implying that a larger fraction of the filters are similar. The same is true for JPEG, but to a lesser extent. For better interpretability, we also plot the cumulative histograms of absolute similarities $|s|$ which saturate abruptly as s approaches 1 for the cases of FP and 1.5-bit log ∇, corresponding to the peaks at $s \approx 1$ in the normal histograms. Examples of highly similar filters are visualized in Appendix D.

RoyChowdhury *et al.* have shown that higher filter similarity allows more channel pruning [20]. We confirm this by fixing $c_2 = 8$

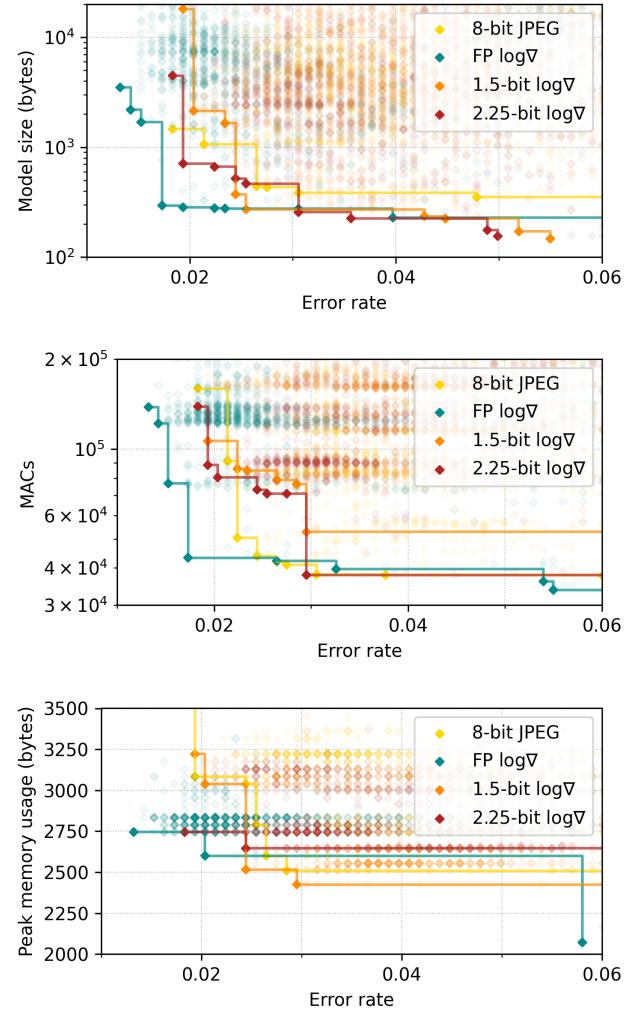


Figure 6: CNN resources vs. classification error rate for high-resolution and aggressively-quanted inputs.

and varying $c_1 \in \{2, 4, 8, 16, 32\}$. From Figure 9, we see that log ∇ inputs need only 2 filters in the first Conv2D layer to achieve 95.3% accuracy, whereas 16-bit RAW requires 32 filters (and JPEG performing somewhere inbetween). This further corroborates the above findings on filter redundancy. Using log ∇ gradient inputs induces more redundancy and hence leads to higher CNN prunability when compared to RAW and JPEG inputs.

As a final experiment using the fixed CNN, we consider the sensitivity to simulated illumination changes. We take the largest networks from the previous experiment ($c_1 = 32, c_2 = 8$) and vary the brightness of the test images by a factor $b \in \{2^{-6}, 2^{-5}, \dots, 2^8\}$ relative to the nominal training brightness. The results in Figure 10 show that both FP and 1.5-bit log ∇ always have the best accuracy and most gradual drops when the images get brighter ($b > 1$). Consistent with the results of [26], the log ∇ accuracy also shows insignificant drops for darker images ($b < 1$). In contrast, the 8-bit

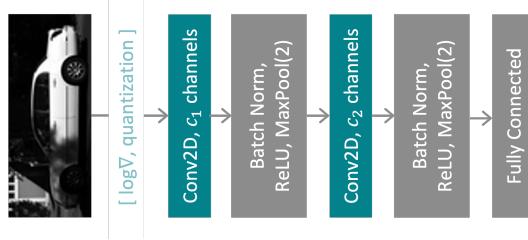


Figure 7: Toy CNN architecture 2conv1fc.

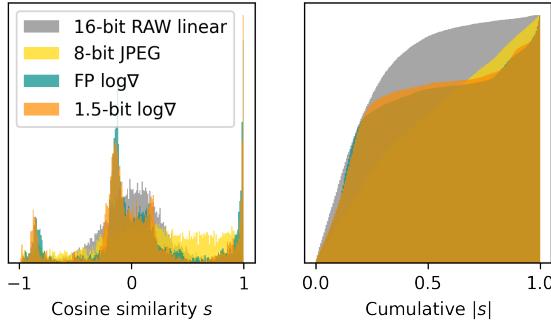
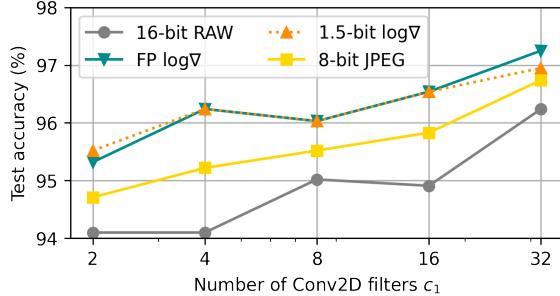


Figure 8: Histograms of cosine similarity.

Figure 9: Test accuracy with $c_1 \in \{2, 4, 8, 16, 32\}$, $c_2 = 8$.

JPEG case shows significant drops in either direction, e.g. 10.9% and 4.3% accuracy deterioration with $2^{-6} \approx 0.016x$ and 8x brightness perturbations while the log ∇ models only drop by about 0.1% and 1.7%. This shows that gamma correction is a useful normalization, but it appears to be inferior to the the log-gradient approach.

4 RELATED WORK

Model compression. There exists a considerable body of literature on reducing the footprint of CNNs using compression techniques like weight and channel pruning as well as quantization [8]. A related approach that is also leveraged in our work is neural architecture search. For tinyML applications, NAS has been particularly effective in mapping CNN workloads onto resource-constrained microcontrollers [2, 15, 16]. Our approach is complementary to this prior work since it focuses more on the input image representation. We have shown that log gradient inputs make the CNN more

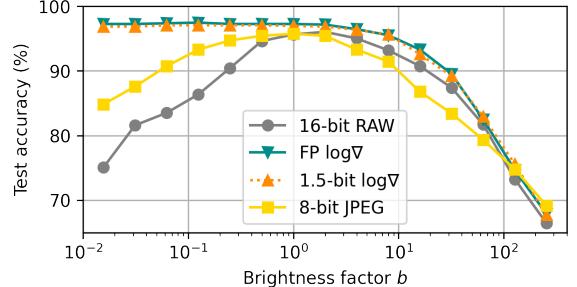


Figure 10: Test accuracy vs. image brightness.

compressible and less sensitive to illumination. Additionally, our approach enables aggressive quantization of the first layer inputs down to 1.5 bits. Typically, at least 5 bits tend to be required using standard JPEG-based inputs [25]. In essence, our work focuses on stripping largely irrelevant information from the CNN input to enable better model compression. Recent work on RNN-based pooling for image patch summarization [21] follows a related train of thought, albeit using more complex methods.

Dataset limitations. Applications of tinyML CV will typically involve real-time image analysis “in the wild,” with varying camera angles and non-ideal or non-uniform illumination. Recent work has established that these effects can lead to significant accuracy reductions for CNNs trained on idealized data [9, 23]. Unfortunately, popular datasets such as ImageNet [7] or Visual Wake Words (VWW) [6] (used for tinyMLPerf benchmarking [1]) are based on Internet photos and do not necessarily reflect what real-world sensors see. Additionally, the original RAW output from the image sensor cannot be reproduced (only estimated) from these JPEG datasets. We sidestep these issues by using the PASCAL RAW 2014 [19] dataset, which contains unprocessed RAW images. Access to the RAW pixel data allowed us to perform simulated illumination changes as shown in Section 3.

ISP simplifications. Various researchers have pointed out that the traditional ISP pipeline is overprovisioned for image consumption by machines [5, 10, 17]. Buckler *et al.* [5] found that demosaicing and gamma correction are the most critical ISP stages for small CNNs. Hansen *et al.* [10] performed an ablation study on ISP stages using the ImageNet dataset [7] and found that tone mapping is important for high dynamic range inputs. Overall, they argue that the energy consumed by some ISP operations is worth investing, since it takes a significantly larger neural network to achieve the same accuracy and generalization without preprocessing. The latter observation agrees with our results in Section 3, but we advocate for log ∇ as a simplified and tinyML-specific preprocessing alternative. Using log ∇ is motivated by physics and collapses the image’s dynamic range close to the sensor (essentially providing a more comprehensible alternative to gamma compression). This stands in contrast to traditional ISP operations, which are remnants of a pipeline optimized for image consumption by humans.

Logarithmic imaging. The Weber-Fechner Law states that the intensity of human perception is proportional to the logarithm of the stimulus [4], providing a biological motivation for log gradients.

Log gradients have also been advocated in high-end imaging for motion rejection and noise reduction [24]. The work of [5] also considered a logarithmic pixel readout but did not investigate gradient preprocessing. The work of Young *et al.* introduced in-sensor log-gradient computation using a linear grayscale imager with ratio-to-digital conversion at 1.5 and 2.75 bits. The energy consumption of this image sensor is in line with the state of the art, consuming only 0.13 nJ/pixel with moderate dynamic range performance (59 dB). However, their backend was based on histograms of oriented gradients (HOG) with a deformable parts model (DPM) as the detection algorithm. These techniques are now outdated and hence this paper investigates log ∇ as inputs to modern CNNs for tinyML.

5 CONCLUSION

Image preprocessing plays an important role in computer vision but is often neglected in the literature. The de-facto standard in today's ML research is to work with JPEG images that are appealing to the human eye. While it is well understood that some of the JPEG transformations are also beneficial for machine vision, tinyML systems of the future will look for an optimum solution that maximizes CNN compressibility and robustness. Our work identified log ∇ preprocessing as a promising option, as it enables aggressive quantization of first-layer inputs, improved CNN prunability, and increased robustness to illumination changes. Future work should consider training-based optimization of the log ∇ quantization thresholds, quantized training to reduce the internal compute precision of the CNN, as well as the response to adversarial inputs.

ACKNOWLEDGMENTS

This work was supported in part by the ACCESS AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR. Qianyun Lu was supported by Stanford Graduate Fellowship in Science & Engineering.

REFERENCES

- [1] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, Urmish Thakker, Antonio Torrini, Peter Warden, Jay Cordaro, Giuseppe Di Guglielmo, Javier Duarte, Stephen Gibellini, Videet Parekh, Honson Tran, Nhan Tran, Niu Wenxu, and Xu Xuesong. 2021. MLPerf Tiny Benchmark. arXiv:2106.07597 [cs.LG]
- [2] Colby Banbury, Chuteng Zhou, Igor Fedorov, Ramon Matas, Urmish Thakker, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul Whatmough. 2021. MicroNets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. *Proceedings of Machine Learning and Systems 3* (2021), 517–532.
- [3] Daniel Bankman, Lita Yang, Bert Moons, Marian Verhelst, and Boris Murmann. 2019. An Always-On 3.8 μ J/86% CIFAR-10 Mixed-Signal Binary CNN Processor With All Memory on Chip in 28-nm CMOS. *IEEE Journal of Solid-State Circuits* 54, 1 (2019), 158–172. https://doi.org/10.1109/JSSC.2018.2869150
- [4] Marcelo Bertalmio. 2020. Chapter 5 - Brightness perception and encoding curves. In *Vision Models for High Dynamic Range and Wide Colour Gamut Imaging*. Marcelo Bertalmio (Ed.). Academic Press, 95–129. https://doi.org/10.1016/B978-0-12-813894-6.00010-7
- [5] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. 2017. Reconfiguring the imaging pipeline for computer vision. In *Proceedings of the IEEE International Conference on Computer Vision*. 975–984.
- [6] Aakanksha Chowdhery, Pete Warden, Jonathon Shlens, Andrew Howard, and Rocky Rhodes. 2019. Visual Wake Words Dataset. arXiv:1906.05721 [cs.CV]
- [7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [8] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proc. IEEE* 108, 4 (2020), 485–532. https://doi.org/10.1109/JPROC.2020.2976475
- [9] Param Hanji, Muhammad Z Alam, Nicola Giuliani, Hu Chen, and Rafal K Mantiu. 2021. HDR4CV: High Dynamic Range Dataset with Adversarial Illumination for Testing Computer Vision Methods. In *London Imaging Meeting*, Vol. 2021. Society for Imaging Science and Technology, 40404–1.
- [10] Patrick Hansen, Alexey Vilkin, Yury Krustalev, James Imber, Dumidu Talagala, David Hanwell, Matthew Mattina, and Paul N Whatmough. 2021. ISP4ML: The role of image signal processing in efficient deep learning vision systems. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2438–2445.
- [11] Bashima Islam and Shahriar Nirjon. 2020. Zygarde: Time-Sensitive On-Device Deep Inference and Adaptation on Intermittently-Powered Systems. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 82 (sep 2020), 29 pages. https://doi.org/10.1145/3411808
- [12] Suyao Ji, Jing Pu, Byong Chan Lim, and Mark Horowitz. 2016. A 220pJ/pixel/frame CMOS image sensor with partial settling readout architecture. In *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*. 1–2. https://doi.org/10.1109/VLSIC.2016.7573545
- [13] Hongyang Jia, Murat Ozatay, Yinqi Tang, Hossein Valavi, Rakshit Pathak, Jinseok Lee, and Naveen Verma. 2021. 15.1 A Programmable Neural-Network Inference Accelerator Based on Scalable In-Memory Computing. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 64. 236–238. https://doi.org/10.1109/ISSCC42613.2021.9365788
- [14] Chi-Cheng Ju, Tsu-Ming Liu, Yung-Chang Chang, Chih-Ming Wang, Chang-Hung Tsai, Ying-Jui Chen, T.H. Wu, Hue-Min Lin, Han-Liang Chou, Abrams Chen, Andy-H.B. Wang, Wc Gu, Wayne Hsieh, Jing-Ying Chang, Shou-Chun Liao, C.T. Ho, Larry Chu, Sokonis Wei, Ch Wang, and Kevin Jou. 2020. 21.3 A 5.69mm² 0.98nJ/Pixel Image-Processing SoC with 24b High-Dynamic-Range and Multiple Sensor Format Support for Automotive Applications. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*. 326–328. https://doi.org/10.1109/ISSCC1994.2020.9063063
- [15] Edgar Liberis, Lukasz Dudziak, and Nicholas D Lane. 2021. μ NAS: Constrained Neural Architecture Search for Microcontrollers. In *Proceedings of the 1st Workshop on Machine Learning and Systems*. 70–79.
- [16] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. 2020. MCUNet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems* 33 (2020), 11711–11722.
- [17] Ekdeep Singh Lubana, Robert P. Dick, Vinayak Aggarwal, and Pyari Mohan Pradhan. 2019. Minimalistic Image Signal Processing for Deep Learning Applications. In *2019 IEEE International Conference on Image Processing (ICIP)*. 4165–4169. https://doi.org/10.1109/ICIP.2019.8803645
- [18] Demetris Marnerides, Thomas Bashford-Rogers, Jonathan Hatchett, and Kurt Debattista. 2018. ExpandNet: A deep convolutional neural network for high dynamic range expansion from low dynamic range content. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 37–49.
- [19] Alex Omid-Zohoor, David Ta, and Boris Murmann. 2014–2015. PASCALRAW: Raw Image Database for Object Detection. http://purl.stanford.edu/hq050zr7488
- [20] Aruni RoyChowdhury, Prakhar Sharma, and Erik G. Learned-Miller. 2017. Reducing Duplicate Filters in Deep Neural Networks. In *NIPS workshop on Deep Learning: Bridging Theory and Practice*.
- [21] Oindrila Saha, Aditya Kusupati, Harsha Vardhan Simhadri, Manik Varma, and Prateek Jain. 2020. RNNPool: efficient non-linear pooling for RAM constrained inference. *Advances in Neural Information Processing Systems* 33 (2020), 20473–20484.
- [22] Muhammad Shafique, Theocharis Theocharides, Vijay Janapa Reddy, and Boris Murmann. 2021. TinyML: Current Progress, Research Challenges, and Future Roadmap. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 1303–1306. https://doi.org/10.1109/DAC18074.2021.9586232
- [23] Qi She, Fan Feng, Xinyu Hao, Qihan Yang, Chuanlin Lan, Vincenzo Lomonaco, Xuesong Shi, Zhengwei Wang, Yao Guo, Yimin Zhang, et al. 2020. OpenLORIS-Object: A robotic vision dataset and benchmark for lifelong deep learning. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 4767–4773.
- [24] Jack Tumblin, Amit Agrawal, and Ramesh Raskar. 2005. Why I want a gradient camera. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. 103–110 vol. 1. https://doi.org/10.1109/CVPR.2005.374
- [25] Bram-Ernst Verhoef, Nathan Laubeuf, Stefan Cosemans, Peter Debacker, Ioannis Papistas, Arindam Malik, and Diederik Verkest. 2019. FQ-Conv: Fully Quantized Convolution for Efficient and Accurate Inference. arXiv:1912.09355 [cs.LG]
- [26] Christopher Young, Alex Omid-Zohoor, Pedram Lajevardi, and Boris Murmann. 2019. A Data-Compressive 1.5/2.75-bit Log-Gradient QVGA Image Sensor With Multi-Scale Readout for Always-On Object Detection. *IEEE Journal of Solid-State Circuits* 54, 11 (2019), 2932–2946. https://doi.org/10.1109/JSSC.2019.2937437
- [27] Yuhan Zhu, Anand Samajdar, Matthew Mattina, and Paul Whatmough. 2018. Euphrates: Algorithm-SoC Co-Design for Low-Power Mobile Continuous Vision (ISCA '18). IEEE Press, 547–560. https://doi.org/10.1109/ISCA.2018.00052

A IMAGE RECONSTRUCTION FROM $\log \nabla$

To reconstruct a RAW image from its $\log \nabla$, we train a two-layer network (2conv, see Figure 11) in TensorFlow, employing padding to ensure that the feature maps after the Conv2D layers have the same dimensions as the original input images. In the Conv2D layers, the channel counts are 10 and 1 respectively; filter sizes are chosen to be large to learn the global correlations in each training image (kernel size of 16 for 96x96x1 images). The loss is set as the mean square error between the RAW prediction and the ground truth (the original RAW images). The Adam optimizer is used with learning rate=0.0002. The batch size is 128 and the total number of epochs is 150 with the subset split of train : validation : test = 70 : 15 : 15 (%).

B VWW+EXPANDNET EXPERIMENTS

To approximate RAW data, we convert VWW JPEGs back to high dynamic range RGB images with the pretrained CNNs from [18]; the simulated RAW dataset is denoted by “VWW+ExpandNet.” Note that, even though the conversion yields higher dynamic range, it cannot recover all the information lost during JPEG compression. In other words, “VWW+ExpandNet” is just an approximation, not real RAW data. Images in the “person” class are cropped according to bounding boxes [11] to avoid the random cropping and resizing problems when “person” occupies a small area in an image. In other words, the dataset used is not the same as other works like MicroNets [2], and yields higher accuracy.

Similar to the experiments in Section 3, we train the 2conv1fc CNN with $c_1 \in \{2, 4, 8, 16, 32\}$, $c_2 = 8$ on four inputs $I \in \{8\text{-bit JPEG}, 16\text{-bit simulated RAW}, \text{FP } \log \nabla, 1.5\text{-bit } \log \nabla\}$, and plot the test accuracy in Figure 12. The $\log \nabla$ models are less accurate than 8-bit JPEG models because “VWW+ExpandNet” is converted from non-invertible JPEGs, resulting in inaccurate $\log \nabla$ information. Nonetheless, the results are comparable for higher channel counts, indicating that our pipeline may generalize to other datasets. True RAW images would be required to come to a full conclusion. Compared with 16-bit simulated RAW, both FP and 1.5-bit $\log \nabla$ are more accurate with the same c_1 and need a smaller c_1 to achieve the same accuracy, which is consistent with our findings from Section 3.

C TRAINING SCHEDULES

The training details for the μ NAS and the fixed CNN (2conv1fc) models are as given below. They were implemented in TensorFlow and PyTorch, respectively.

C.1 μ NAS training schedule

We train on images with dimensions 96x96x1 and the subset split is train : validation : test = 70 : 15 : 15 (%). Aging evolution was used together with structured pruning. The batch size is 128 and the total number of epochs is 80 with pruning from 40th epoch to 75th; the minimum sparsity and maximum are set to 0.1 and 0.85, respectively. The TensorFlow SGDW optimizer is used with learning rate=0.03, momentum=0.9, and weight decay=0.0001. To limit the resources, the bounds are configured to: error bound=0.05, peak memory bound=10000, model size bound=20000, and mac bound=1000000. GPU info: 4x Nvidia GeForce rtx6000 @ 24GB GDDR6/module.

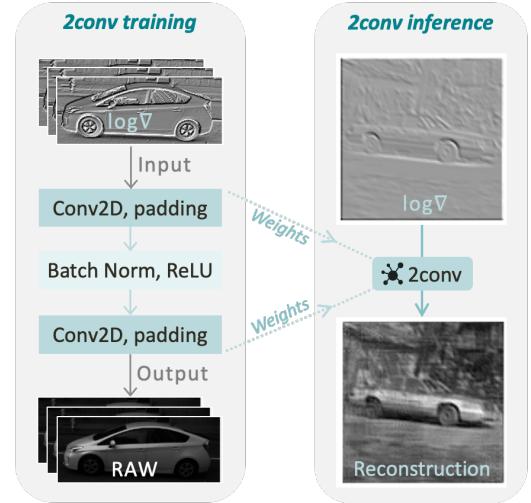


Figure 11: 2conv training and inference for reconstruction.

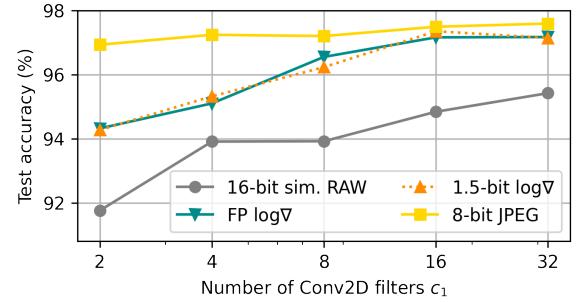


Figure 12: VWW test accuracy with $c_1 \in \{2, 4, 8, 16, 32\}$, $c_2 = 8$.

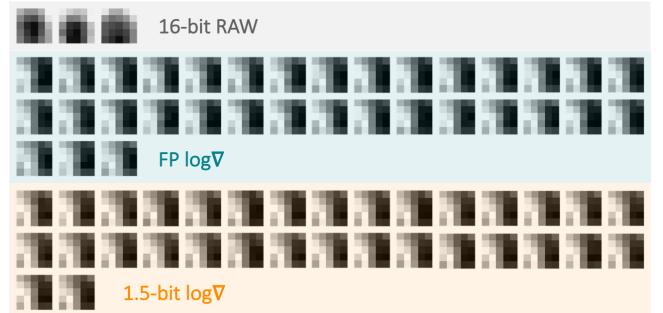


Figure 13: Visualization of similar filters.

C.2 Fixed CNN training schedule

We train on images with dimensions 224x224x1 and the subset split is train : validation : test = 70 : 15 : 15 (%). The total number of epochs is 40 with a batch size of 64. The Adam optimizer is used with learning rate=0.001, scheduled to decay by gamma=0.95 every epoch. GPU info: 1x Nvidia GeForce rtx6000 @ 24GB GDDR6/module.

D VISUALIZATION OF SIMILAR FILTERS

Figure 13 shows filters of high similarity ($s_{i,j} > 0.98$). For both FP and 1.5-bit log ∇ , we find a larger number of filters that are close to identical (about 11x more than for RAW).