

Browser-Deployed Lightweight YOLO for Banana Ripeness Classification on Commodity Edge Devices

Sumetee Jirapattarasakul*, Sirinya Thanyacharoen[†], Giuseppe Riccardo Leone[‡] and Thaweesak Yingthawornsuk[†]

*Dept. of Electrical and Information Engineering,
King Mongkut's University of Technology Thonburi (KMUTT), Bangkok, Thailand
e-mail: sumetee.jirap@kmutt.ac.th

[†]Department of Media Technology, King Mongkut's University of Technology Thonburi (KMUTT), Bangkok, Thailand
e-mail: {sirinya.than, thaweesak.yin}@kmutt.ac.th

[‡]Institute of Information Science and Technologies, National Research Council of Italy, Pisa, Italy
e-mail: giuseppericcardo.leone@cnr.it

Abstract—Automated banana ripeness inspection is increasingly important for agricultural sorting at small and medium scale. Existing solutions either rely on cloud computer-vision services, which raise concerns of cost, latency, and data privacy, or on dedicated edge accelerators (e.g. Raspberry Pi or Jetson devices) which carry a non-trivial hardware barrier. The goal of this study is to determine whether a modern lightweight image classifier can be deployed and run *entirely inside a web browser*, on the consumer hardware that producers already own, while retaining sufficient accuracy for practical quality grading.

We collected a longitudinal dataset of 24 unique banana fruits imaged over three days under controlled storage conditions, and trained a YOLOv8n-cls classifier (You Only Look Once, version 8, nano classification variant; 1.44 M parameters, 5.52 MB after export to the Open Neural Network Exchange, ONNX, format) under three class schemes (4-class fine-grained, 3-class quality grading, 2-class industrial binary).

We report results under three evaluation protocols including a Leave-One-Day-Out (LODO) cross-validation that controls for fruit-identity leakage. Under this leakage-controlled protocol the classifier achieves 0.818 ± 0.045 accuracy for 3-class quality grading, while on-device inference through `onnxruntime-web` runs at 17.8 ± 0.4 ms latency (56 frames per second) on a consumer laptop CPU. Beyond the accuracy numbers, we quantify the magnitude of banana-identity leakage in random cross-validation (a 20% accuracy gap for the binary task), highlighting a methodological pitfall in small longitudinal fruit datasets.

Index Terms—banana ripeness classification, YOLO, edge inference, ONNX runtime web, leave-one-day-out cross-validation, browser deployment

I. INTRODUCTION

BANANAS are among the most globally traded fruits, and their commercial value depends strongly on ripeness at the point of distribution. Manual inspection by trained operators remains the dominant practice in many producing regions, but it is slow, labour intensive, and prone to inter-operator disagreement. Automating this inspection step with computer vision—specifically by tracking visual features such as skin colour and texture changes—is therefore an active area of research. Most recent work relies on convolutional

neural networks or You Only Look Once (YOLO)-family detectors [1], [2].

Two deployment patterns dominate the literature. The first sends images to a remote cloud API, which works but introduces recurring costs, network latency, and privacy concerns for producers who do not wish to share their operational imagery. The second deploys models on dedicated edge devices such as a Raspberry Pi, NVIDIA Jetson, or Coral accelerator. These solutions are technically attractive but require producers to purchase and maintain specialised hardware, which presents a real barrier in small farms and cooperative settings [3].

To address these limitations, the primary purpose of this position paper is to propose and validate a “zero-infrastructure” edge computing paradigm for agricultural applications. We explore a third deployment option: running the trained classifier entirely inside the user’s web browser, using the WebAssembly backend of `onnxruntime-web` [4] (utilizing the Open Neural Network Exchange format). The browser thus becomes a universal edge device, and inference runs locally on whatever laptop, tablet, or smartphone the operator already owns. The complete pipeline—spanning from image acquisition and model training to model export and fully serverless client-side inference—is illustrated in Fig. 1.

Our specific contributions are threefold: (1) a longitudinal dataset of bananas under three storage conditions collected during a joint Thailand–Italy internship program; (2) a rigorous comparison of three class schemes (4, 3, and 2 classes) and three evaluation protocols, notably including a leakage-controlled Leave-One-Day-Out cross-validation; and (3) an end-to-end deployment pipeline that places the trained classifier into a single self-contained HTML file. Ultimately, we articulate a research agenda advocating for browser-based machine learning as a highly transferable solution that can be readily extended beyond banana inspection to other agricultural grading tasks, significantly lowering the barrier to AI adoption for small-scale producers.



Fig. 1. End-to-end pipeline. Bananas are imaged with a USB camera, used to train a YOLOv8n-cls classifier, exported to ONNX, and deployed through onnxruntime-web for client-side inference inside any modern browser. No server component is required at inference time.

II. RELATED WORK

A. Fruit ripeness classification

Early work on automated banana grading relied on hand-crafted colour and texture features fed into shallow classifiers [5]. With the rise of deep learning, recent literature has increasingly adopted convolutional neural networks and You Only Look Once (YOLO) variants for end-to-end ripeness prediction [1], [2], [6]. Methodologically, these studies typically frame the problem as either whole-image classification or bounding-box object detection, relying heavily on spatial features and colour transformations. While recent publications frequently report accuracies above 90% on highly curated datasets, there remains a critical methodological gap: very few studies explicitly control for fruit-identity leakage during cross-validation and evaluation, potentially inflating real-world performance metrics.

B. Edge inference for agricultural applications

Deploying inference at the edge is highly attractive in agriculture, particularly in rural settings where network connectivity is intermittent. Recent surveys document a wide breadth of machine-learning applications in this sector [3]. For computer-vision workloads, successful edge deployments are frequently reported on dedicated microcomputers such as the Raspberry Pi, Jetson Nano, and similar IoT accelerator boards [2]. However, a persistent methodological limitation in this literature is the assumption that producers possess the capital and technical expertise to acquire and maintain this dedicated hardware. This hardware barrier strongly motivates our proposed shift toward zero-infrastructure solutions.

C. In-browser machine learning

Browser-based inference has matured substantially over the last few years. Frameworks such as TensorFlow.js [7] and ONNX Runtime Web [4] expose WebAssembly and WebGL execution providers, enabling complex pre-trained models to run entirely client-side without relying on back-end server infrastructure. To our knowledge, while browser-based AI is gaining traction in general web development, this zero-infrastructure deployment pattern has not yet been systematically evaluated as a scalable methodology for fruit classification or broader agricultural grading tasks.

III. DATASET

A. Acquisition setup

Images were collected at the KMUTT Media Technology laboratory during a joint internship program with the Institute of Information Science and Technologies of the National Research Council of Italy. The acquisition platform consisted of a Specim Scanner systems base with controlled lighting and a fixed top-down camera mount. Each fruit was placed at the centre of a dark mat and imaged with two cameras concurrently: a wide-angle USB webcam and a Raspberry Pi camera module. This work uses only the USB-camera images; the Raspberry Pi images exhibited a severe magenta colour cast (consistent with a missing or saturated white-balance calibration) that prevented direct cross-camera training. Resolving this cross-sensor mismatch is left as future work.

B. Classes and image analysis methodology

The acquisition guide defined four ripeness classes: *Blue* (ready to eat), *Black* (almost ready, slight greenish tint), *Green* (raw, half green and half yellow), and *Small* (slightly raw fruit of a smaller cultivar, partially green and partially yellow). We

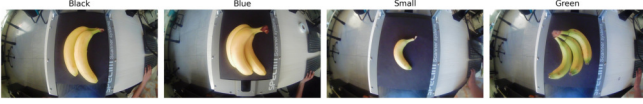


Fig. 2. One representative banana per ripeness class (USB camera, top-down view). From left to right: Black, Blue, Green, Small. The visual similarity between the Green and Small classes dominates fine-grained classification errors.

TABLE I
DATASET COMPOSITION (USB CAMERA, 4-CLASS BASELINE).

Date	Black	Blue	Green	Small
15 May	6	6	4	6
16 May	6	6	4	6
19 May	6	6	4	6
Total	18	18	12	18

note that the textual definitions of the Green and Small classes substantially overlap, a property we analyse in Section V. Representative images of each class are shown in Fig. 2.

To address the image analysis methodology, we emphasize that explicit semantic segmentation or manual background subtraction was not performed. Instead, the raw RGB images were resized and fed holistically into the YOLOv8n-cls network. While skin colour is naturally a dominant predictive feature for ripeness, the deep learning model also automatically extracts spatial and morphological features (e.g., curvature, dark spots, and textural changes) directly from the whole image to inform its classification boundary.

The 4-class composition is summarised in Table I. Initially, an acquisition of 24 unique banana subjects was planned. However, to clarify the sample distribution ($6+6+4+6 = 22$), 2 subjects were excluded from the final analysis due to severe framing and camera focus errors during the initial capture. Therefore, the refined dataset tracks exactly 22 unique banana subjects imaged at three time points (15, 16, and 19 May), yielding the 66 usable USB-camera images.

IV. METHODOLOGY

A. Model architecture and classification method

We adopt YOLOv8n-cls [2], the smallest classification variant of the YOLOv8 family of detectors and classifiers descended from the original YOLO architecture [1]. The model comprises approximately 1.44 M parameters and 3.3 GFLOPs at an input resolution of 224×224 pixels. The input pattern matrix (tensor) is defined as $X \in \mathbb{R}^{N \times 3 \times 224 \times 224}$, where N is the batch size, 3 represents the RGB colour channels, and the spatial dimensions are 224×224 . The target vector is formulated as $Y \in \{0, 1\}^{N \times C}$, where $C \in \{2, 3, 4\}$ depending on the selected class scheme. The final classification output is computed via a Softmax activation function applied to the unnormalised logits.

Weights are initialised from ImageNet pre-training. This choice reflects the deployment constraint of running inference in a browser through WebAssembly: larger models would

exceed reasonable download sizes and increase per-inference latency.

B. Computational environment

To ensure rapid prototyping and stable convergence during cross-validation, the model training and hyperparameter optimization phases were executed in a high-performance cloud computing environment using Google Colab Pro. This environment provided access to accelerated GPU instances, which were essential for handling the intensive data augmentation and repetitive fold-training required by our evaluation protocols.

C. Data augmentation

Because the training set is small (at most 36 images per fold), strong augmentation is required. We apply random horizontal and vertical flips (banana orientation on a conveyor is arbitrary), small rotation ($\pm 15^\circ$), translation and scaling, random erasing with probability 0.3, and HSV jitter. The hue shift parameter is kept deliberately small (0.01) because ripeness is signalled primarily by colour and a large hue perturbation would destroy the very feature the model must learn.

D. Class-merging ablation

We evaluate three class schemes. The 4-class scheme uses the original labels. The 3-class scheme merges Green and Small into a single *Unripe* category, motivated by the overlapping textual definitions of the two source classes. The 2-class scheme further merges Black and Blue into a single *Ripe* category, mirroring the practical decision boundary used in industrial sorting (ready to ship versus not yet ready).

E. Evaluation protocols

We report three protocols. The *temporal split* trains on 15 + 16 May and tests on 19 May; it is leakage-free by construction because each test image depicts a banana at a ripening state that does not appear in training. A *stratified random 5-fold cross-validation* pools all 66 images and shuffles. Finally a *Leave-One-Day-Out (LODO) cross-validation* trains on two of the three days and tests on the remaining day, cycling through all three combinations.

The three protocols differ in how strictly they control for identity leakage. Because the same 22 usable physical fruits were imaged on every day, a random k-fold split is permitted to place images of the same fruit at different ripening times into both train and test partitions. LODO eliminates this within-fold identity leakage while still allowing the model to see each fruit at some ripening states during training.

F. Software components and browser deployment

As our primary algorithmic and software contribution beyond the underlying neural network, we engineered a custom end-to-end client-side inference pipeline. Leveraging our expertise in modern website design and development, the front-end user interface was prototyped in Figma to ensure a professional, minimalist design suitable for non-technical

TABLE II
BANANA RIPENESS CLASSIFICATION UNDER THREE EVALUATION PROTOCOLS.

Scheme	Temporal split (15 + 16 → 19)	Random 5-fold CV (mean ± std)	LODO CV (mean ± std)
4-class	0.682	0.727 ± 0.039	0.667 ± 0.052
3-class	0.636	0.940 ± 0.034	0.818 ± 0.045
2-class	0.727	1.000 ± 0.000	0.803 ± 0.224

agricultural operators. The web application logic was then developed using React.

After training, we export the PyTorch checkpoint to ONNX (opset 20) and simplify the computational graph with `onnxslim`. The resulting model is loaded directly into the React-based browser environment by `onnxruntime-web` [4] using the WebAssembly execution provider. The entire pipeline (image upload, normalisation, forward pass, and result display) runs exclusively client-side; no inference data ever leaves the user’s device, preserving complete data privacy. A screenshot of the deployed application is shown in Fig. 4.

V. RESULTS

A. Classification performance

Table II reports accuracy under all three evaluation protocols and class schemes. Three observations follow.

First, classification accuracy increases monotonically as the class scheme coarsens, which is expected because merging confusable classes removes ambiguous decision boundaries. The pattern is stronger than a naive reading would suggest because the source ambiguity is concentrated in the Green–Small pair, as illustrated in the aggregate confusion matrix in Fig. 3: the model rarely confuses Black or Blue with either Green or Small.

Second, random 5-fold CV is substantially more optimistic than LODO CV for the coarser schemes, reaching 1.000 for the 2-class task with zero variance across folds. We attribute this gap to banana-identity leakage: because each of the 22 usable physical fruits was imaged on every day, random folds can place images of the same fruit on different days into both train and test partitions, and the model can exploit fruit-specific shape features rather than learning generalisable ripeness cues. The gap between random CV and LODO CV ($\Delta = 0.20$ for 2-class, 0.12 for 3-class) is a direct estimate of the magnitude of this leakage.

Third, the per-fold accuracies of the 2-class LODO experiment exhibit unusually high variance (0.955, 0.909, 0.545 for test days 15, 16, and 19 May respectively). The collapse on the latest day suggests that the bananas continued to ripen during the five-day observation window, so the model trained on the earlier two days encounters a shifted distribution at test time. Larger and more temporally diverse training data would be required to absorb this drift.

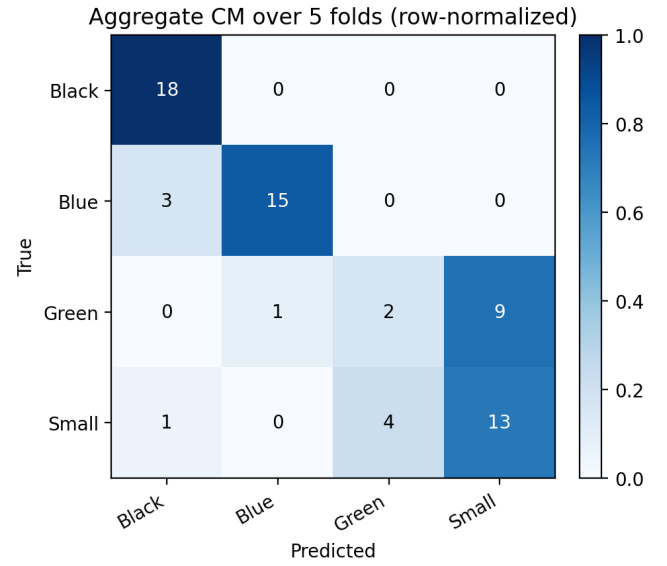


Fig. 3. Aggregate confusion matrix over 5 folds of stratified random cross-validation under the 4-class scheme. Confusions between Green and Small dominate the error budget; the Black/Blue ripe pair is essentially never confused with the Green/Small unripe pair.

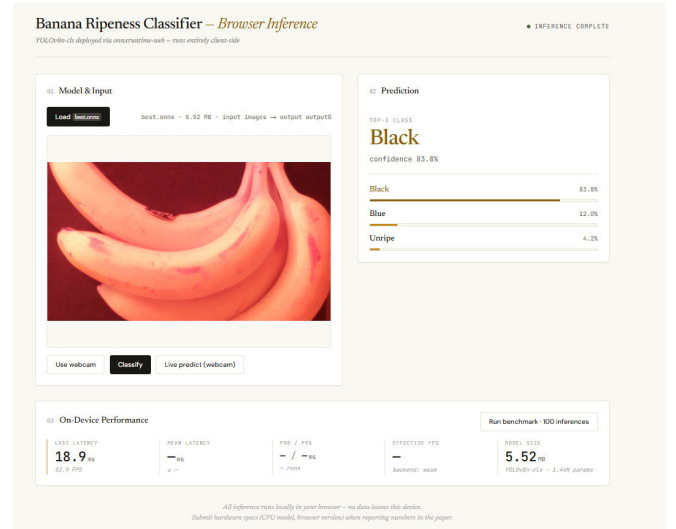


Fig. 4. Screenshot of the browser-deployed inference application after running the embedded 100-inference benchmark on a consumer laptop. The interface, the model, and all measurements run entirely client-side via the WebAssembly backend of `onnxruntime-web`; no server is involved.

B. On-device inference performance

Table III summarises the browser inference benchmark collected on a consumer laptop. All numbers are measured directly by the custom React-based web application itself (Fig. 4), deployed on the same consumer device that an end user would operate.

The mean throughput of 56 FPS exceeds the 30 FPS threshold that is typically taken to indicate real-time inspection on a sorting line, and the very tight $p95/p50$ ratio of 1.03 indicates

TABLE III
BROWSER INFERENCE PERFORMANCE VIA ONNXRUNTIME-WEB
(WEBASSEMBLY BACKEND).

<i>Hardware</i>	
CPU	AMD Ryzen 5 5600H
RAM	16 GB
Browser	Google Chrome
Backend	onnxruntime-web (WebAssembly)
<i>Model</i>	
Format	ONNX (opset 20, simplified)
Size	5.52 MB
Parameters	1.44 M
FLOPs	3.3 G
<i>Inference</i>	
Mean latency	17.8 ± 0.4 ms
p50 / p95 latency	17.8 / 18.3 ms
Effective throughput	56.1 FPS
Benchmark size	100 timed runs + 10 warm-up

that the latency distribution has no significant tail under sustained load. The GPU is not used by the WebAssembly backend; all timing reflects CPU performance only.

VI. DISCUSSION

A. Label granularity versus classification difficulty

The accuracy difference between the 4-class and 2-class schemes is larger than the difference between the 3-class and 2-class schemes, which is consistent with the Green–Small ambiguity dominating fine-grained errors. Practically, this suggests that an operational system should report whichever level of granularity is genuinely needed by the downstream sorting decision rather than attempting the most fine-grained classification by default.

B. Identity-aware evaluation for small fruit datasets

The discrepancy between random and LODO cross-validation in our experiments is a concrete warning about evaluation methodology in small longitudinal fruit datasets. Whenever the same physical fruit is imaged at multiple time points, a random k-fold split silently introduces identity leakage that inflates reported accuracy. We recommend that subsequent work on banana ripeness, and on similar fruit datasets, report results under at least one identity-aware protocol.

C. Browser deployment as a scalable zero-infrastructure paradigm

The browser is, in effect, a universal edge runtime. Any laptop, tablet, or smartphone manufactured in roughly the last five years can load our deployed React application from a static URL and run the classifier locally without further setup. The benefits are several: zero hardware investment, intrinsic privacy because images never leave the device, and one-step deployment to all platforms simultaneously. The principal disadvantage is dependence on the host’s CPU; throughput on a low-end phone will be lower than the numbers reported here.

Crucially, in aligning with the goals of this position paper, we argue that this zero-infrastructure paradigm extends far beyond banana ripeness classification. The software architecture we developed—combining a lightweight YOLO model, WebAssembly execution, and a user-friendly frontend—is highly transferable. It can be readily adapted for sorting other types of fruits, detecting crop diseases, or evaluating general product quality, fundamentally lowering the barrier to AI adoption for small and medium-scale agricultural enterprises globally.

VII. LIMITATIONS AND FUTURE WORK

The current dataset comprises only 22 unique usable fruits across 4 classes, which limits external validity. The large standard deviation of the 2-class LODO accuracy ($\sigma = 0.224$) indicates that the model is sensitive to which days are used for training; a longer observation window and a larger cohort of subjects would address this. Furthermore, the Raspberry Pi camera images were excluded because of an unresolved colour cast, and the USB-camera images have visible barrel distortion that we did not correct; implementing a proper camera-calibration pre-processing step would likely improve cross-camera generalisation.

Regarding future work and practical implementation, extending the system to multi-banana detection on a moving conveyor belt is a primary objective. This would require transitioning from a whole-image classifier to a YOLO-based object detection model trained on bounding boxes, coupled with a multi-object tracking algorithm to prevent double-counting fruits. Additionally, incorporating new analytical features—such as bruise/defect detection, physical size estimation, and peel texture analysis—would provide a much more comprehensive quality grading system.

Practically, the browser-deployed nature of our application allows for immediate integration into small-scale packing houses. Operators can simply mount a standard smartphone or tablet above a manual sorting table, access the web URL, and receive real-time visual overlays or audio cues to assist their sorting decisions, effectively digitizing quality control with zero upfront hardware investment.

VIII. CONCLUSION

In this position paper, we have presented a banana ripeness classifier trained on a longitudinal dataset of 22 unique fruits and deployed entirely inside a standard web browser. Under a strictly leakage-controlled Leave-One-Day-Out cross-validation, the classifier reaches 0.818 ± 0.045 accuracy on a 3-class quality-grading task while running inference at 17.8 ms latency (56 FPS) on commodity laptop hardware.

Beyond the accuracy results, we highlight two critical take-aways for the agricultural AI community. First, we quantify the magnitude of fruit-identity leakage that can inflate results in naive random cross-validation by up to 20% on binary tasks, emphasizing the need for rigorous, identity-aware evaluation methodologies in small datasets. Second, we demonstrate that browser deployment is a highly viable, scalable, and privacy-preserving zero-infrastructure alternative to dedicated edge

hardware. By eliminating the hardware barrier, this paradigm offers a practical pathway to bring advanced computer vision directly into the hands of small and medium-scale agricultural producers.

AI TOOL DISCLOSURE

The authors utilized a large language model assistant (Claude, Anthropic) to support the development process in the following capacities: code optimization and debugging assistance for the Python training and evaluation scripts; and generating boilerplate code to aid with the `onnxruntime-web` integration within the custom web application. Additionally, the assistant was used for copy-editing and refining portions of the initial manuscript text. The system overview figure (Fig. 1) was generated using Google's Gemini image generation model based on a detailed textual description provided by the authors. All experimental designs, core website design and development, and data analyses were conceptualized and executed by the authors. The authors independently verified all experimental results, reviewed and edited every section of the manuscript, and accept full responsibility for its content and conclusions. The use of AI assistance did not influence the evaluation methodology, the identification of the fruit-identity leakage issue, or the scientific findings of this work.

ACKNOWLEDGMENTS

The authors thank the Institute of Information Science and Technologies of the National Research Council of Italy (CNR-

ISTI) for hosting the internship during which the dataset was collected, and the Department of Media Technology at King Mongkut's University of Technology Thonburi (KMUTT) for providing the acquisition platform. The internship was made possible thanks to the collaborative framework provided by the CNR-ISTI and KMUTT Memorandum of Understanding (Prot. CNR n. 0204025, dated 2024-06-13).

REFERENCES

- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [2] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," <https://github.com/ultralytics/ultralytics>, 2023, software, version 8.x.
- [3] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018.
- [4] Microsoft, "ONNX Runtime Web," <https://onnxruntime.ai/docs/tutorials/web/>, 2024.
- [5] F. M. A. Mazen and A. A. Nashat, "Ripeness classification of bananas using an artificial neural network," *Arabian Journal for Science and Engineering*, vol. 44, no. 8, pp. 6901–6910, 2019.
- [6] I. A. Sabilla, C. S. Wahyuni, C. Faticah, and D. Herumurti, "Determining banana types and ripeness from image using machine learning methods," in *International Conference on Artificial Intelligence and Information Technology*, 2019.
- [7] D. Smilkov, N. Thorat, Y. Assogba, A. Yuan, N. Kreeger, P. Yu, K. Zhang, S. Cai, E. Nielsen, D. Soergel, S. Bileschi, M. Terry, C. Nicholson, S. N. Gupta, S. Sirajuddin, D. Sculley, R. Monga, G. Corrado, F. B. Viegas, and M. Wattenberg, "TensorFlow.js: Machine learning for the web and beyond," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 309–321, 2019.