ORIGINAL RESEARCH Open access

Lightweight Deep Learning and Edge Computing for Real-Time Tomato Disease Detection

Lina Hassan[1*], Omar Khalaf[2], Reem Jaber[3], Rania Mostafa[4]

Abstract

The global tomato supply chain is constantly imperiled by diseases that drastically reduce yields and pose persistent production threats. Enabling early, automated disease recognition would empower growers to rapidly implement containment strategies, curb propagation, and increase productivity. Convolutional neural network (CNN) techniques based on deep learning have been extensively adopted for identifying tomato diseases. Yet, the intensive computational requirements of deep learning models pose a significant obstacle to their practical use in agricultural contexts such as disease surveillance and detection. Recently, the evolution of open-source Edge hardware has unlocked the possibility of practical, low-cost, energy-sipping deep learning applications in agriculture. Accordingly, the central aim of this work was to assess the efficacy of self-contained Edge-AI systems for detecting tomato leaf diseases. Toward this end, the study first utilized computationally efficient deep learning architectures to identify and distinguish among multiple tomato leaf conditions (bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, two-spotted spider mites, target spot, and yellow leaf curl virus). These architectures were subsequently implemented on affordable, energy-frugal Edge platforms to probe their operational viability as autonomous Edge-AI tools for prompt disease identification. The resource-conscious CNN-based models GoogleNet and MobileNetV2 achieved classification accuracies of 98.25%, compared with 98.13%, 98.13%, 94.62%, and 90.67% for EfficientNetB0, ResNet-18, SqueezeNet, and NasNetMobile, respectively. NVIDIA Jetson ORIN AGX and Nano substantially eclipsed the Raspberry Pi and Raspberry Pi augmented with a Google Coral AI accelerator for image classification speed, processing images in 3.5 ms and 5.2 ms, respectively, with SqueezeNet, compared to 15.3 ms and 10.5 ms on the Raspberry Pi-based configurations. Moreover, the Raspberry Pi paired with Google Coral yielded the strongest cost-to-FPS metric at 0.14, whereas the NVIDIA Jetson AGX Orin and Jetson Nano Orin registered 0.7 and 0.26, respectively. These outcomes underscore the potential of independent Edge-AI frameworks that leverage low-cost, energy-efficient software and hardware for early-stage detection of tomato diseases.

Keywords Deep learning, CNN, Edge devices, Plant disease, Disease scouting, Edge-AI

*Correspondence:
Lina Hassan
lina.hassan@ju.edu.jo
[1] Department of AI for Tomato and Cucumber Greenhouse Yield Prediction, Faculty of Agriculture, University of Jordan, Amman, Jordan
[2] Department of Machine Learning for Pest and Beneficial Insect Classification, Faculty of Agriculture, Jordan University of Science and Technology, Irbid, Jordan
[3] Department of AI for Lavender and Rose Oil Production Optimization, Faculty of Agriculture, Agricultural University of Plovdiv, Plovdiv, Bulgaria
[4] Department of Smart Nutrient Film Technique Monitoring, Faculty of Agriculture, Al-Balqa Applied University, Salt, Jordan

Introduction

This is an open access article distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0), which permits non-commercial use, distribution, reproduction, and adaptation in any medium or format, provided that appropriate credit is given to the original author(s) and source, a link to the license is provided, and any changes made are indicated. Adapted material must be distributed under the same license.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Ranking among the world's most important crops, tomato cultivation generates over 189 million metric tonnes each year [1]. Nonetheless, diseases remain a relentless challenge, consistently undermining production. The ability to pinpoint diseases accurately and without delay could help halt their advance, thereby raising yields and stabilizing market values. Minimizing the fallout from disease episodes allows farmers to preserve dependable supply volumes, a cornerstone of effective price forecasting [2, 3]. In customary practice, tomato growers diagnose diseases by visually scrutinizing symptoms. Because these observation-driven approaches rely on human know-how, they tend to be labor-intensive, slow, and prone to delays in information delivery. Automated, early-stage disease identification would enable producers to immediately deploy countermeasures to limit the spread of outbreaks, ultimately boosting tomato output. While computer vision is powerful at autonomously detecting and categorizing diseases, it falls short on its own for real-time deployments in resource-strapped environments like farms. The barrier lies in the typically hefty computational footprint of computer vision models, which may be unattainable in field conditions. Edge-AI resolves this by analyzing data locally on endpoints such as cameras or sensors, enabling instantaneous analysis without continuous reliance on the cloud. Such an architecture slashes latency, bolsters system robustness, and raises efficiency, making it exceptionally suitable for on-the-spot disease diagnosis. Marrying Edge-AI with computer vision, therefore, yields speedier, more resilient, and more streamlined disease detection right where data originates, enabling swift interventions and reduced crop forfeitures. Hence, the primary aim driving this study is to examine how standalone Edge-AI configurations perform in the task of tomato leaf disease detection.

Related work

In recent years, methodologies grounded in computer vision have surged in popularity owing to their potency and proficiency in swift, early disease detection [4, 5]. Established computer vision paradigms have been widely deployed to identify plant ailments [6]. These techniques have drawn upon support vector machines (SVM) [7-9], fused Haar-like attributes with the Viola-Jones algorithm [10], harnessed discrete wavelet transforms and gray-level co-occurrence characteristics for SVM and K-Nearest Neighbor (K-NN) [11, 12], combined local binary pattern (LBP) descriptors with SVM [13], and blended Haralick textural measures, Hu moments, and chromatic features with SVM [14] as well as random forest [15] to pinpoint and separate tomato diseases. Yet, the dependence of these conventional strategies on manually crafted disease signatures impedes their success in realistic diagnostic scenarios. For hand-designed features, the capacity to represent attributes is limited, which hampers the refinement and broad applicability of these approaches for disease recognition.

To remedy this shortcoming, deep learning-powered convolutional neural networks (CNNs) deliver elevated hierarchical capacities for extracting and encoding features, culminating in greater detection precision and resilience [16]. Over the years, a variety of researchers have developed CNN-based deep learning approaches for recognizing and categorizing tomato diseases. For example, Thangaraj *et al.* [17] proposed a multi-tiered feature fusion within a CNN-based Xception architecture to detect tomato diseases. Bora *et al.* [18] applied a multivariate normal deep learning model to tomato disease classification. Along similar lines, Kaur *et al.* [19] utilized a tailored InceptionResNetV2 to detect tomato foliage afflictions. Nawaz *et al.* [20] leveraged the Faster R-CNN framework to detect and localize tomato diseases. Fuentes *et al.* [21] harnessed both Faster R-CNN and SSD networks for this purpose. Wang *et al.* [22] resorted to vision transformer (ViT) rooted techniques for detection. Ferentinos [23] explored transfer learning across various CNN-based networks—AlexNet, AlexNetOWTBn, GoogleNet, Overfeat, and VGG—to detect plant diseases. Roy *et al.* [24] formulated PCA DeepNet by fusing CNN, principal component analysis (PCA), and a generative adversarial network (GAN) to flag tomato leaf diseases. Correspondingly, Chen *et al.* [25] architected a tomato disease diagnosis pipeline merging ABCK-BWTR with B-ARNet. That said, deep learning models are voracious consumers of computational capability, ballooning in breadth and depth, encompassing millions or billions of tunable parameters and millions of floating-point operations (FLOPs). Vaster and more elaborate models promise increased fidelity but command considerable processing wherewithal [26, 27]. This computational load poses a bottleneck to the practical adoption of deep learning for agrarian tasks such as disease tracking and detection.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Over the last several years, the maturation of open-source Edge platforms has paved the way for practical, low-cost, energy-efficient deep learning deployments across sectors such as E-Healthcare [28], Smart Cities [29], and Smart Farms [30]. Breakthroughs in hardware, spanning system-on-a-chip (SoC) accelerators, application-specific integrated circuits (ASICs), graphics processing units (GPUs), and central processing units (CPUs), have helped narrow the computational gap between heavyweight deep learning architectures and the constrained resources of Edge hardware. These Edge ecosystems enable inference procedures to run natively on the device with minimal or no internet connectivity, a trait that makes them well-suited for realistic deep learning deployments in farm settings, including tomato disease scouting [31]. In recent times, a handful of investigations have documented the embedding of deep learning networks on Edge appliances for Edge AI. For instance, Anh and Duc [32] operationalized MobileNetV3 on a Raspberry Pi 3 to survey plant leaf maladies. Gonzalez-Huitron *et al.* [33] rolled out and benchmarked a quartet of CNNs—MobileNetV2, NasNetMobile, Xception, and MobileNetV3—on a Raspberry Pi 4 for tomato disease identification. Similarly, Tarek *et al.* [34] put MobileNetV3 onto a Raspberry Pi 4 for the same detection task. Moreira *et al.* [35] trialed assorted CNN-based algorithms, namely AlexNet, DenseNet, and SqueezeNet, for mobile-oriented plant disease surveillance. Zhang and Chen [36] conceived an Edge cloud-mediated solution for detecting tomato leaf curl disease. In a similar vein, Nag *et al.* [37] proposed a mobile application that uses the DenseNet-121 network for tomato disease diagnosis. Nonetheless, these endeavors either presuppose internet access for data offloading/communication [35-37] or exclusively compare disparate deep learning models on a singular device (most commonly the Raspberry Pi) [32-34].

Goals and objectives

The broad intent of this study is to gauge the operational effectiveness of standalone Edge-AI solutions for detecting disorders in tomato leaves. To meet this aim, the investigation began by deploying lightweight deep learning networks to identify and sort tomato leaf diseases. Afterward, these networks were ported to low-cost, low-energy-demand Edge devices to evaluate their functional performance as self-reliant Edge-AI instruments for early disease detection. The findings from this research are expected to deliver crucial performance metrics for standalone Edge-AI configurations that leverage affordable, energy-efficient software and hardware to enable early detection of tomato diseases.

Materials and Methods

The architecture of our Edge-AI detection pipeline is shown in **Figure 1**, outlining the sequence by which a tomato leaf disorder is recognized on an Edge appliance. The sections that follow unpack the steps for assembling the dataset, the lightweight CNN designs, porting to Edge hardware, and the evaluation metrics.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

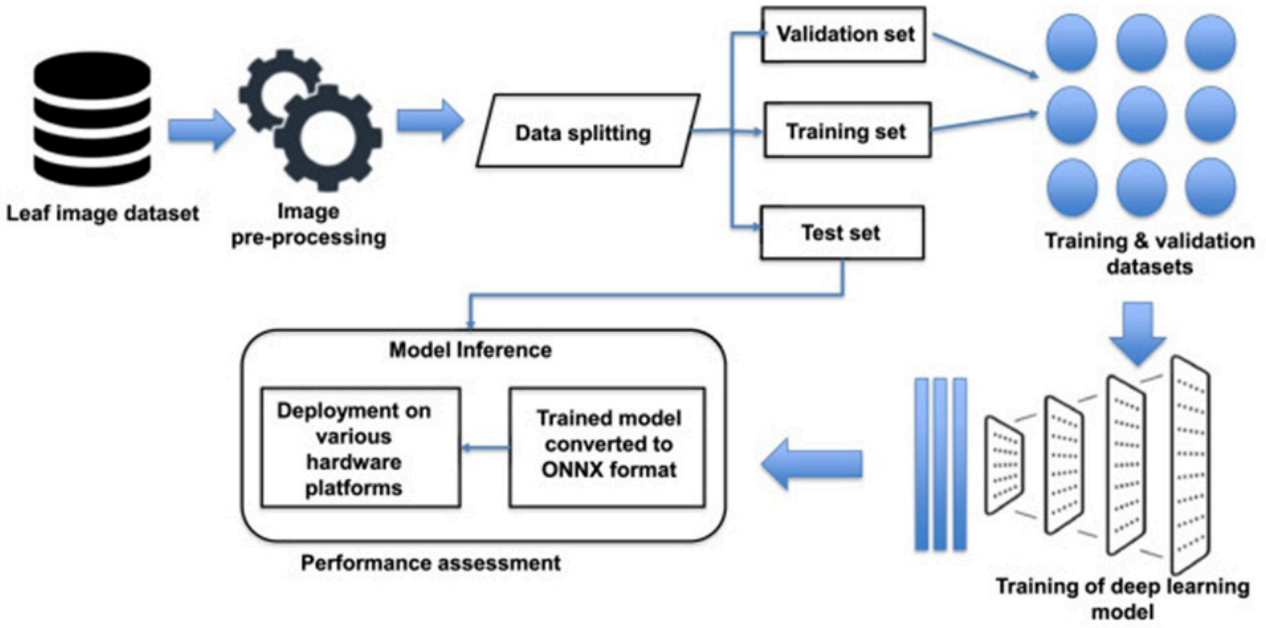


Figure 1. Schematic representation of the Edge-AI-based system for spotting tomato leaf ailments, depicting the journey from raw data handling through to on-device inference.

Dataset preparation

Because the overriding purpose of this investigation was to benchmark how well self-sufficient Edge-AI configurations perform at recognizing diseases on tomato foliage, the image bank was drawn from the PlantVillage repository [38]. A total of 8568 tomato leaf snapshots, distributed across 9 separate categories (bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, two-spotted spider mites, target spot, and yellow leaf curl virus), were pulled from PlantVillage. Every single picture carries a pixel dimension of 256 × 256. The pool was split at random into a training portion (roughly 80% of the overall count: 6856 samples), a validation portion (roughly 10%: 857 samples), and a testing portion (roughly 10%: 855 samples). To guard against overfitting and boost the models' generalization, synthetic data diversification was undertaken. Three augmentation operations were applied: stochastic horizontal mirroring, stochastic rotation, and stochastic resizing. Horizontal mirroring flipped images left-to-right with a 50% chance. Rotation picked an angle at random from the interval 90° to 90°, while resizing multiplied dimensions by a factor drawn randomly between 1 and 2. These transformations were imposed on-the-fly and in random combination on each training sample before it entered the network.

Lightweight CNN architectures

Convolutional operations in deep learning help tease out feature representations of target entities in an image. The size of the resulting feature maps is determined by several choices: stride length, depth, filter variety, filter dimensions, and padding scheme. **Eq. 1** captures the mechanics of how a convolutional layer produces a feature map:

$$F_j^n = b_j^n + \sum_k w_{jk}^n \bigstar x_k^{n-1} \quad (1)$$

In this expression, F_j^n stands for the feature map emerging from the nth convolutional layer, built from the preceding layer's output x_k^{n-1} after being convolved with the weight tensor w_{jk}^n. The additive term b_j^n injects non-linearity through

© 2024 The Author(s). | Published by Agro Intelligence Research Press

bias, enabling the model to latch onto complex input patterns.

Capitalizing on the multi-level feature abstraction capabilities of CNNs, our approach leveraged CNN-grounded deep learning models to achieve precise and resilient identification of tomato leaf disorders. Moreover, given the goal of embedding these models into Edge-AI contexts, the candidate pool was restricted to architectures with fewer than 12 million parameters. Six compact CNN designs were ultimately enacted for the task: ResNet-18 [39], EfficientNetB0 [40], NasNetMobile [41], SqueezeNet [42], MobileNetV2 [43], and GoogleNet [44], which respectively encompass 11.7 million, 5.3 million, 5.3 million, 1.24 million, 3.5 million, and 7 million parameters, each having been pre-trained on the ImageNet corpus [45]. These backbones initially learned general visual concepts from large-scale datasets such as ImageNet for broad-spectrum object categorization. To redirect them toward tomato pathology recognition, their input interfaces, internal weights, and final decision layers had to be swapped out and retuned to align with the exact number of target classes and the subtle visual signatures of tomato leaf afflictions. During retuning, the macro-architecture and parameter budget of the CNNs remain mostly unchanged. What retuning accomplishes is allowing the networks to absorb domain-specific cues pertinent to the new assignment and to mitigate issues like vanishing gradients, culminating in superior fidelity and robustness. The following section compares the structural DNA of each network and itemizes the targeted layer modifications implemented to enable repurposing for effective tomato leaf disease detection. **Figure 2** encapsulates the end-to-end procedure for wielding CNN-centric deep learning models in this diagnostic context.

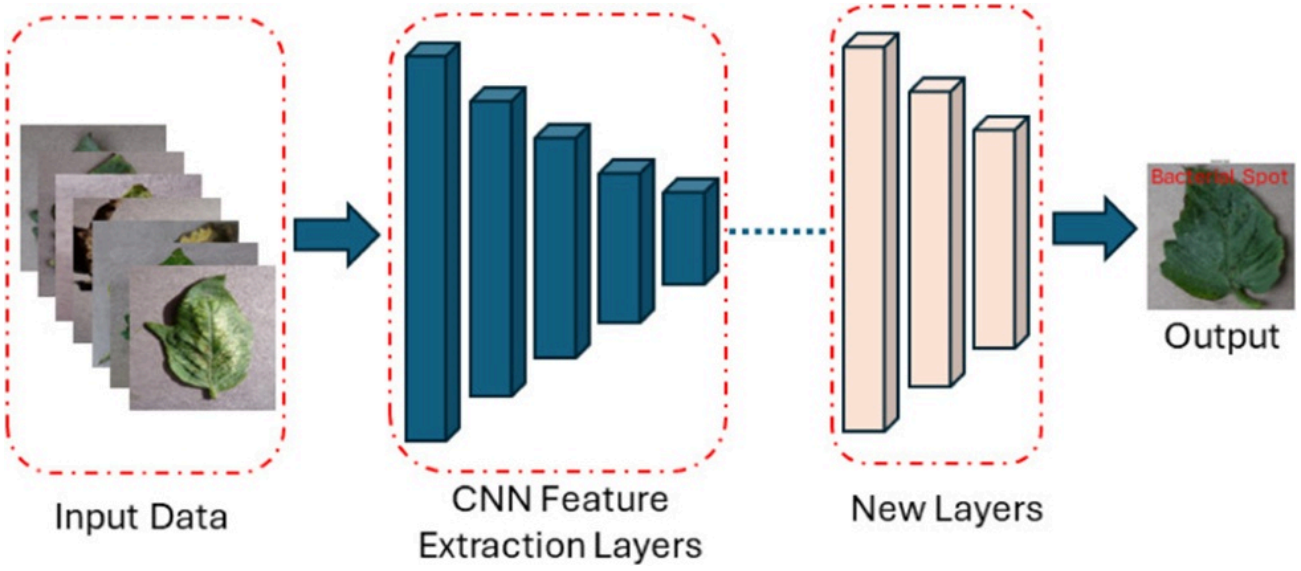


Figure 2. High-level depiction of the methodology for leveraging different CNN-based deep learning architectures (isolating CNN-derived features and grafting new layers) to pinpoint tomato leaf ailments.

ResNet-18 is structured around residual units that integrate shortcut connections with 18 convolutional layers for distilling high-level descriptors. EfficientNet is built from MBConv (mobile inverted bottleneck) blocks that funnel narrow input and output tensors through an expanded intermediate layer. The EfficientNet family spans seven scales, labeled B0 through B7, which vary by block configuration. EfficientNetB0 was chosen for this work because it offers the lowest latency and the most frugal parameter count. NasNetMobile operates based on the NasNet search space paradigm, which interleaves convolutional layers and filter banks. To match NasNetMobile's required input footprint, training images were rescaled to 224 × 224 pixels. MobileNetV2 relies on depthwise separable convolutions, which factor the convolution into a pointwise and a depthwise stage, sharply reducing both computational load and model complexity. GoogleNet pursues an expansive yet parameter-conscious design by blending deeper and wider convolutional stacks with the Hebbian learning principle and multi-resolution processing to maximize output quality while restraining the parameter budget. ResNet-18, EfficientNetB0, NasNetMobile, MobileNetV2, and GoogleNet all expect an input resolution of 224 × 224 pixels, so the training imagery was standardized to that size.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Additionally, the decision heads of these architectures are composed of fully connected layers. Accordingly, their original dense layers were stripped out and replaced by a freshly initialized fully connected layer, with both the weight and bias learning rate multipliers set to 10. Following this, a new 9-way classification layer was swapped in to generate the class-membership probabilities. SqueezeNet, by contrast, relies on Fire modules, squeeze convolutional layers, and a deliberate deferral of downsampling to attain competitive accuracy with an ultra-lean parameter profile. For this network, the training images were resized to 227 × 227 pixels to conform to its input geometry. The classification head of SqueezeNet uses 2D convolutional layers rather than dense layers. Hence, a novel 2D convolution layer, configured with weight and bias learning factor values of 10 each, was crafted and substituted for the original to handle disease labeling. A new 9-class classification layer was then introduced to output the probability distribution over classes.

To steer the convergence behavior of these models, stochastic gradient descent (SGD), formalized in **Eq. 2**, was deployed as the optimizer, with cross-entropy loss, given in **Eq. 3**, serving as the minimization target:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L\left(\theta_t; x_i, y_i\right) \quad (2)$$

$$\text{Loss} = -\sum_k^{i=1} y_i \log\left(p_i\right) \quad (3)$$

In these equations, θt denotes the model parameters at iteration t, η sets the learning rate, and η∇θL(θt; xi,yi) is the gradient of the cost function L with respect to θ, evaluated on the data pair (xi,yi). In the loss formula, k is the total number of categories, yi is a binary indicator of the true class, and pi is the model's predicted probability for class i.

To facilitate rapid convergence while retaining a low loss profile, the starting learning rate and mini-batch size were fixed at 0.0001 and 10, respectively, values determined after multiple rounds of empirical adjustment. The training data order was also randomly permuted at the start of each epoch to reduce the risk of overfitting. All network training runs were conducted on a workstation outfitted with an NVIDIA® RTX™ A6000 GPU possessing 48 GB of VRAM, complemented by an Intel® Xeon® Gold 6246R CPU and 256 GB of system RAM. The RTX A6000 packs state-of-the-art RT Cores, Tensor Cores, and CUDA cores, furnishing exceptional throughput for rendering, AI inferencing, graphics, and high-performance computing. It interfaces with memory across a 384-bit bus, delivering 768 GB/s of memory bandwidth. In terms of raw compute capability, it reaches peak performance of 38.7 TFLOPS for standard operations, 75.6 TFLOPS for single-precision floating-point operations, and 309.7 TFLOPS for RT Core and Tensor workflows.

CNN model deployments on edge devices

Key commercial players driving the edge of AI include NVIDIA, Qualcomm, and Google. Google Coral [46], for example, provides a cost-effective pathway for deploying complex algorithms. In a related vein, Gabdullin and Raskovalov [47] harnessed multiple CNN variants and operationalized them with Google Coral for the task of person re-identification (re-ID). On another front, NVIDIA [48] stands as the preeminent force driving Edge-AI innovation and offers a broad product portfolio. The NVIDIA Jetson line, including the Nano Orin and AGX Orin modules, balances affordability with a compact footprint and lightweight profile, while still capable of executing sophisticated, multi-layered CNN workloads. A further strength of the NVIDIA Jetson ecosystem is its fused CPU-GPU architecture [49], where the CPU handles system bootstrapping and the CUDA-capable GPU shoulder-taps to accelerate heavy machine learning pipelines.

Within the scope of this investigation, a set of Edge computing platforms was put to the test: the Raspberry Pi 4 Model B (RPI 4) both standalone and paired with a Google Coral accelerator, the NVIDIA Jetson Orin Nano, and the NVIDIA Jetson AGX Orin, as visualized in **Figure 3**. **Table 1** provides a head-to-head summary of the core technical specifications of these shortlisted hardware options.

© 2024 The Author(s). | Published by Agro Intelligence Research Press



Figure 3. The Edge computing hardware under examination for the Edge-AI framework. Top left: NVIDIA Jetson Orin Nano, Top right: NVIDIA Jetson AGX Orin, Bottom left: RPI 4 Model B coupled with Google Coral, Bottom right: RPI 4 Model B.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Table 1. The fundamental specifications of the Edge platforms scrutinized in this research are listed, along with their prevailing commercial retail prices at the time of publication.

Feature	NVIDIA Jetson Orin Nano	NVIDIA Jetson Orin AGX	Google Coral	Raspberry Pi 4 Model B (RPI 4)
Size	100 mm × 79 mm × 21 mm	110 mm × 110 mm × 71.65 mm	65 mm × 30 mm	95 × 70 × 24.9 mm
CPU	6-core Arm Cortex-A78AE v8.2 64-bit CPU (1.5 MB L2 + 4 MB L3)	12-core Arm Cortex-A78AE v8.2 64-bit CPU (3 MB L2 + 6 MB L3)	NXP i.MX 8M SoC (quad Cortex-A53 + Cortex-M4F)	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC
Hardware Accelerator	1024-core NVIDIA Ampere GPU with 32 Tensor Cores	2048-core NVIDIA Ampere GPU with 64 Tensor Cores	Integrated GC7000 Lite graphics	Not available
Memory	8 GB 128-bit LPDDR5 (68 GB/s)	64 GB 256-bit LPDDR5 (204.8 GB/s)	4 GB LPDDR4	8 GB LPDDR4
Power Consumption	7–15 W	15–60 W	15 W	15 W
Weight	173 g	880 g	19.6 g	45.5 g
Peak Performance	40 TOPS	275 TOPS	4 TOPS	Not available
Price	$499	$1999	$60	$75

Google Coral operates as an external USB processing dongle that supplements a host machine with an Edge tensor processing unit (TPU) co-processor. Connection is via a USB-C interface, linking it to a primary computer to speed up model inference. Housed within is Google's proprietary application-specific integrated circuit (ASIC), the Edge TPU, which races through TensorFlow Lite workloads while sipping electrical energy, topping out at 4 trillion operations per second (4 TOPS) yet drawing merely 2 W. Routing machine learning computation through Google Coral trims latency and fortifies the safeguarding of private data.

Within NVIDIA's portfolio is a family of boards that carry embedded computers equipped with discrete GPUs for hardware-level acceleration. The two NVIDIA Edge platforms examined here are the NVIDIA Jetson Orin Nano and the NVIDIA Jetson Orin AGX.

Arriving in March 2023, the NVIDIA Jetson Orin Nano packs a space-efficient layout teeming with interface connectors and musters up to 40 TOPS, readily servicing a diverse set of AI demands. Built around an Ampere GPU and a six-core ARM CPU, the Orin Nano juggles several simultaneous AI workflows alongside high-throughput inferencing. Its modules deliver up to an 80-fold performance improvement over the prior Jetson Nano, granting it the muscle to tackle modern model types, including transformer networks and advanced robotic perception stacks.

Stepping up in capability, the NVIDIA Jetson Orin AGX debuted in May 2022 and delivers up to 275 TOPS of AI horsepower, with power draw adjustable across a 15 W to 60 W range. This unit shines when deploying bulky, sophisticated models for domains like semantic language interpretation, three-dimensional scene understanding, and the fusion of multiple sensor streams. Packing up to 8 times the punch of the Jetson AGX Xavier in a similarly compact enclosure, the Orin AGX is well-suited for engineering a wide variety of AI products. At its heart lies the NVIDIA Ampere architecture, housing 2048 CUDA cores and 64 tensor cores, paired with a 12-core Arm Cortex-A78AE v8.2 64-bit

© 2024 The Author(s). | Published by Agro Intelligence Research Press

processor with 3 MB of L2 and 6 MB of L3 cache. It is engineered to sustain numerous concurrent inference streams and integrates 64 GB of soldered eMMC storage, 204 GB/s of memory throughput, and 64 GB of DRAM. High-bandwidth I/O for an array of sensors comes standard, encompassing 22 PCIe Gen4 lanes, a single 10GbE interface, DisplayPort, 16 MIPI CSI-2 lanes, USB 3.2 connectivity, and a 40-pin expansion header.

By comparison, the RPI 4 stands as one of the most ubiquitous embedded computing platforms, serving an exceptionally wide range of use cases. This inexpensive, miniature board integrates onboard electronics that support an expansive set of input/output peripherals. A 64-bit Broadcom BCM2711 quad-core Cortex-A72 SoC clocked at 1.8 GHz provides the horsepower, and memory options range from 1 to 8 GB. The board features Gigabit Ethernet, dual-band (2.4 GHz and 5 GHz) IEEE 802.11b/g/n/ac Wi-Fi, a pair of USB 2.0 ports, a pair of USB 3.0 ports, two micro-HDMI outputs, and a microSD slot. Critically, it omits a dedicated GPU entirely.

Once the model assortment had been trained, the focus shifted to porting them to tangible hardware. Within the experimental framework, six architectures—ResNet-18, EfficientNetB0, NasNetMobile, SqueezeNet, MobileNetv2, and GoogleNet—were run across four distinct hardware setups.

The initial grouping uses the NVIDIA Jetson Orin Nano as the central compute engine, as shown in **Table 1**. The second grouping is built around the NVIDIA Jetson Orin AGX, whereas the third relies on the RPI 4 Model B. The fourth arrangement pairs an RPI 4 Model B as the host with a Google Coral accelerator connected via a USB 3.0 port. Pictorial representations of each arrangement are given in **Figure 4**.

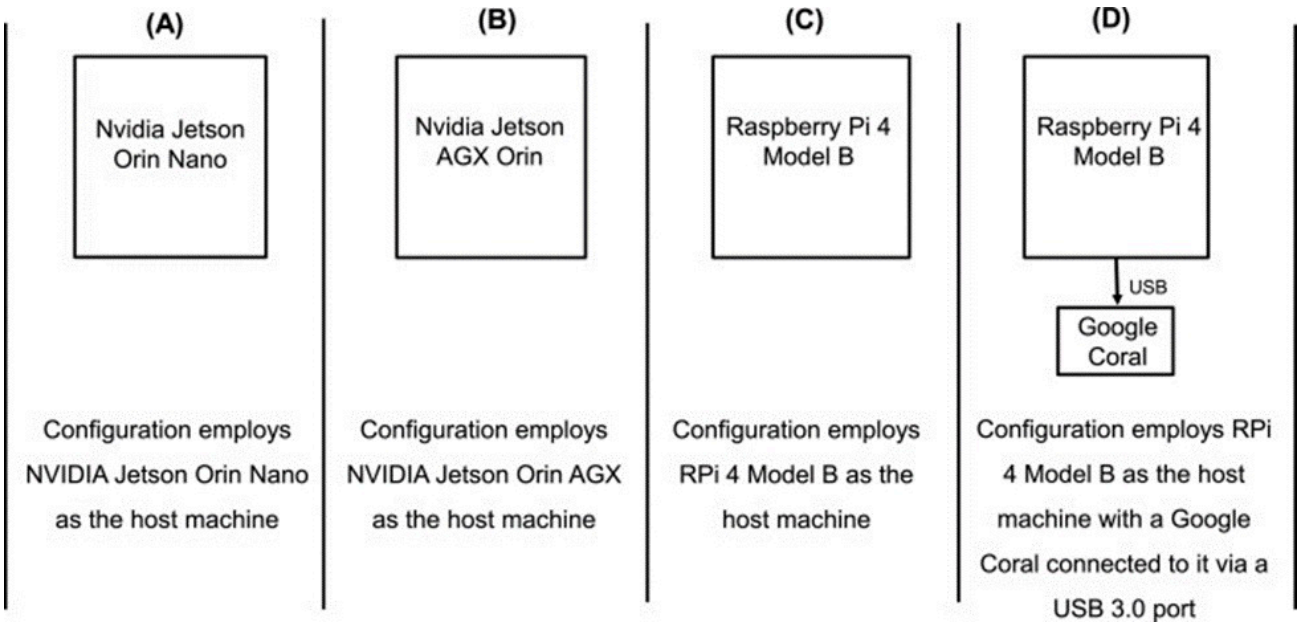


Figure 4. Schematic of the Edge-AI categorization pipeline when realized on (a) NVIDIA Jetson Orin Nano, (b) NVIDIA Jetson Orin AGX, (c) RPI 4 Model B, and (d) Raspberry Pi 4 Model B coupled with a Google Coral module.

To circumvent the restrictions imposed by any single platform, a conscious decision was made not to translate the models into TensorRT, a format that works exclusively on NVIDIA hardware. Because the evaluation required a fair head-to-head comparison across all platforms, including the Raspberry Pi family, for which TensorRT offers no support, an alternative path was necessary. TensorRT typically serves as a supporting role alongside training frameworks, leveraging GPU resources to accelerate network inference.

The chosen alternative was the Open Neural Network Exchange (ONNX) format, which supports the entire set of target platforms and thereby enables a rigorous comparative analysis. ONNX occupies an intermediary position, serving as a

© 2024 The Author(s). | Published by Agro Intelligence Research Press

conduit between PyTorch models and TensorRT representations. Because model training was originally conducted in MATLAB and the artifacts were stored with the ".mat" extension, the conversion to ONNX format was straightforward.

Beyond format choices, Docker containers were uniformly used across devices to ensure a consistent runtime environment. Each container image bundles all necessary libraries and software dependencies, making deployment smoother and more repeatable.

Performance evaluation

Quantifying the capability of the trained CNN designs relied on the metrics of accuracy, precision, recall, and F1-score. Accuracy captured the fraction of samples for which the tomato disease was correctly assigned out of the entire pool of samples **(Eq. 4)**. Precision gauged the proportion of instances tagged as a particular positive category that genuinely belonged to that category **(Eq. 5)**. Sensitivity, also termed recall, assessed the proportion of true positive instances that the model successfully retrieved **(Eq. 6)**. The F1-score was deployed to represent overall model quality as the harmonic mean that balances precision against recall **(Eq. 7)**.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad (4)$$

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (5)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (6)$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (7)$$

The suite of indicators marshalled to contrast and interpret inference behavior across the four Edge device configurations included the following:

· CPU utilization: The share of available CPU capacity consumed while performing inference, presented as a percentage obtained by averaging over all accessible CPU cores. To illustrate, on the 12-core NVIDIA Jetson Orin AGX, the value reflects the mean across those dozen cores. The scale runs from 0 to 100.

· GPU utilization: The proportion of GPU capacity tapped during inference, likewise expressed on a 0-to-100 percentage scale.

· Average power consumption: The mean electrical wattage drawn by the hardware throughout the inference period, given in watts (W).

· Classification time: The span of time needed to assign a label to one individual image.

· Frames per second (FPS): The count of images handled per unit of time while the classifier operates. It is mathematically equivalent to the inverse of the classification time.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Results and Discussion

Tomato leaf disease detection with lightweight CNN architectures

By the terminal epochs, both the training and validation loss curves, along with their associated accuracy levels, had converged for all architectures, indicating that the models had reached stable solutions without latching onto spurious patterns in the training corpus. This property was critical for enabling correct disease identification on the previously unseen test data.

Table 2 compiles the averaged accuracy, precision, recall, and F1-score of the trained CNN models when evaluated on the test partition. GoogleNet and MobileNetV2 produced the top-tier outcomes. The maximum average values for accuracy, precision, recall, and F1-score registered at 0.9825, 0.9825, 0.9829, and 0.9825, respectively, for GoogleNet, and at 0.9825, 0.9825, 0.9827, and 0.9825, respectively, for MobileNetV2, outstripping the remaining contenders. ResNet-18 and EfficientNetB0 turned in roughly equivalent figures but lagged behind GoogleNet and MobileNetV2 on every metric. NasNetMobile ranked last across all four measures relative to the other architectures. The observation that NasNetMobile's recall (0.9347) exceeded its accuracy (0.9076) suggests a significant false-positive issue. **Figure 5** presents the per-class accuracies, precisions, recalls, and F1-Scores for each distinct tomato leaf pathology. It becomes evident that NasNetMobile encountered pronounced difficulty with the target spot and two-spotted spider mite categories, leading to a sharp drop in recall and a higher false-negative rate. GoogleNet, in contrast, posted strong results across all categories, demonstrating steady, trustworthy behavior irrespective of disease type. It is worth emphasizing that ResNet-18, despite having the highest parameter count (11.7 million), underperformed both GoogleNet (7 million parameters) and MobileNetV2 (3.5 million parameters). Moreover, GoogleNet and MobileNetV2 tied for average accuracy, even though MobileNetV2 used only half the parameters of GoogleNet. Possessing the leanest parameter budget (1.24 million), SqueezeNet still managed an accuracy of 0.9462, a profile that could prove attractive in deployments where economizing on computation is the overriding concern. Because the tallies of training, validation, and test samples remained identical across all architectures, and the training logs showed no indication of overfitting, the evidence points to the absence of a positive relationship between parameter volume and accuracy. This insight holds practical significance when picking a model suited for Edge deployment (an expanded treatment of this matter appears in Section 3.2).

Table 2. Synopsis of performance measures (Accuracy, Precision, Recall, and F1-Score) belonging to the CNN architectures tasked with tomato leaf disease recognition.

Model	F1-score	Recall	Precision	Accuracy
GoogleNet	0.9825	0.9829	0.9825	0.9825
MobileNetV2	0.9825	0.9827	0.9825	0.9825
SqueezeNet	0.9463	0.9477	0.9462	0.9462
NasNetMobile	0.9119	0.9347	0.9076	0.9076
EfficientNetB0	0.9812	0.9817	0.9813	0.9813
ResNet-18	0.9812	0.9816	0.9813	0.9813

© 2024 The Author(s). | Published by Agro Intelligence Research Press

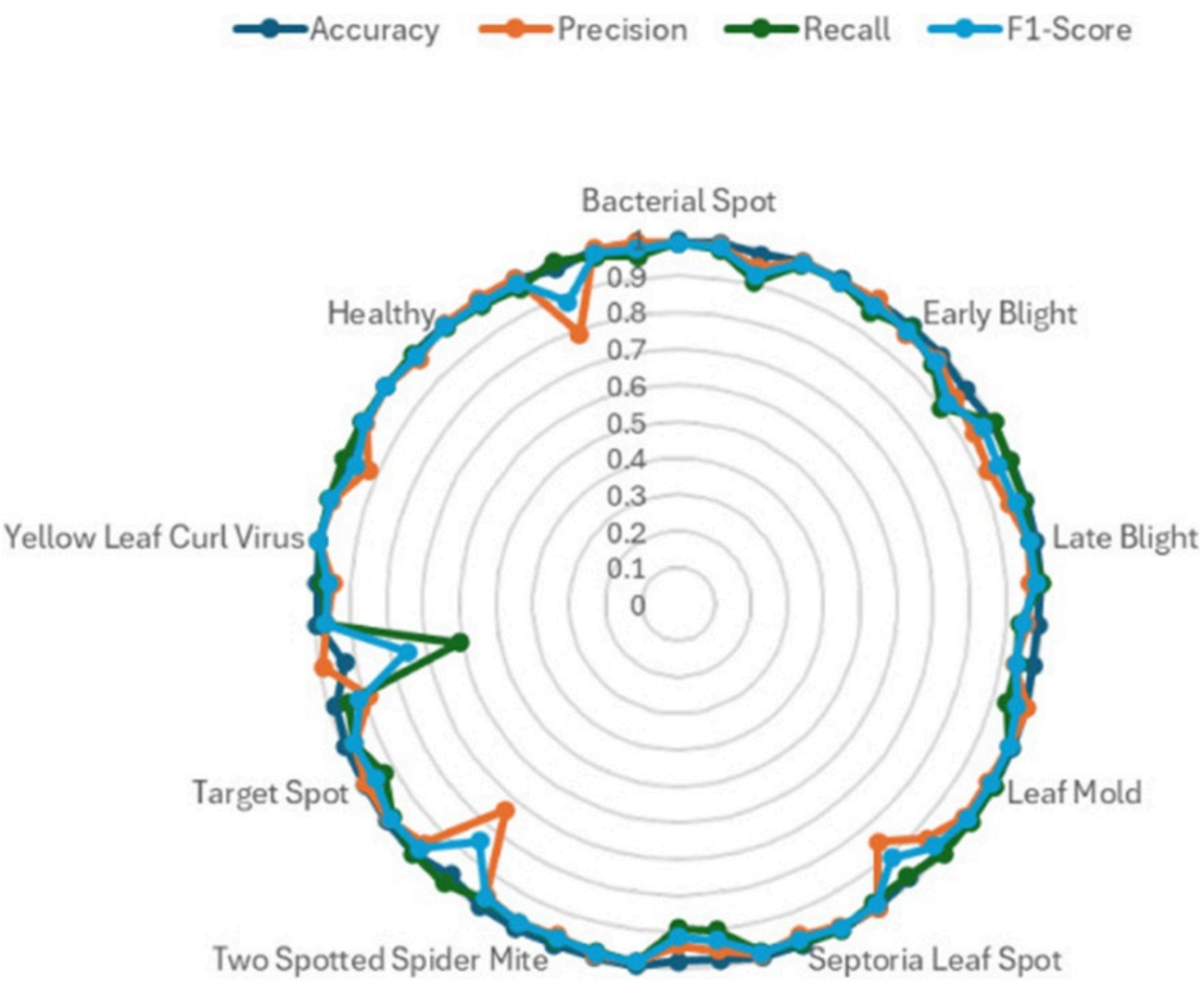
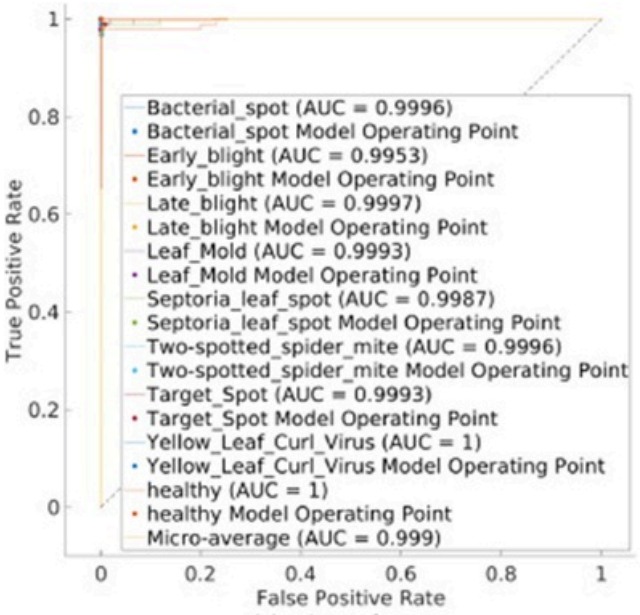


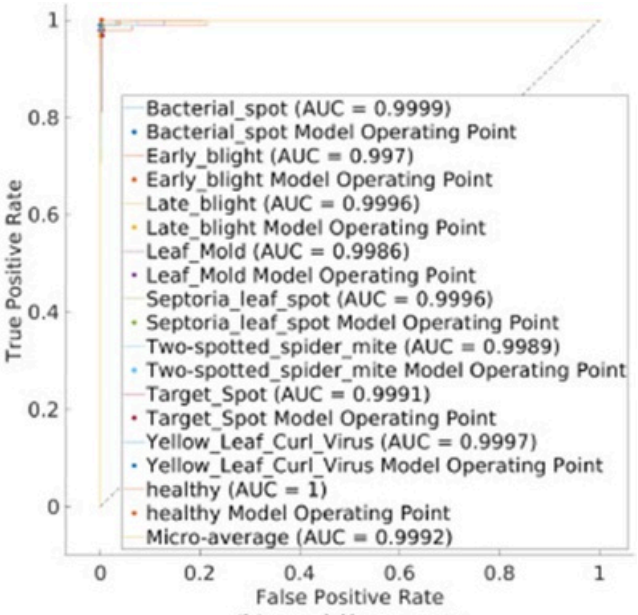
Figure 5. Class-by-class breakdown of accuracy, precision, recall, and F1-score for each tomato leaf disease category. The networks are arranged clockwise following the sequence: GoogleNet, MobileNetV2, SqueezeNet, NasNetMobile, EfficientNetB0, and ResNet-18.

Sensitivity-versus-specificity trade-offs for each network across the spectrum of tomato diseases were examined using receiver operating characteristic (ROC) curves, which are shown in **Figures 6a–6f**. A consistent finding was that the operating coordinates for every disease class across all architectures lay above the diagonal chance line (depicted as a dashed black segment), confirming that model behavior for each condition was more than adequate—genuine positives substantially outnumbered false positives. Area-under-curve (AUC) scores were uniformly high, with every disease-model pairing achieving values of 0.99 or higher (tightly approaching the perfect score of 1), indicating broadly competent overall performance. The pronounced upward-leftward bowing of each curve additionally signals robust model behavior across a range of classification cutoffs. For NasNetMobile, whose average accuracy ranked lowest among the cohort as illustrated in **Figure 6c**, the principal drag on performance originated from target spot and two-spotted spider mite, where the operating thresholds dipped below a true positive rate of 0.8. This pattern indicates that the model regularly assigned these two conditions to incorrect categories. To dig deeper into the misclassification patterns, confusion matrices were scrutinized, as set out in **Figures 7a–7f**.

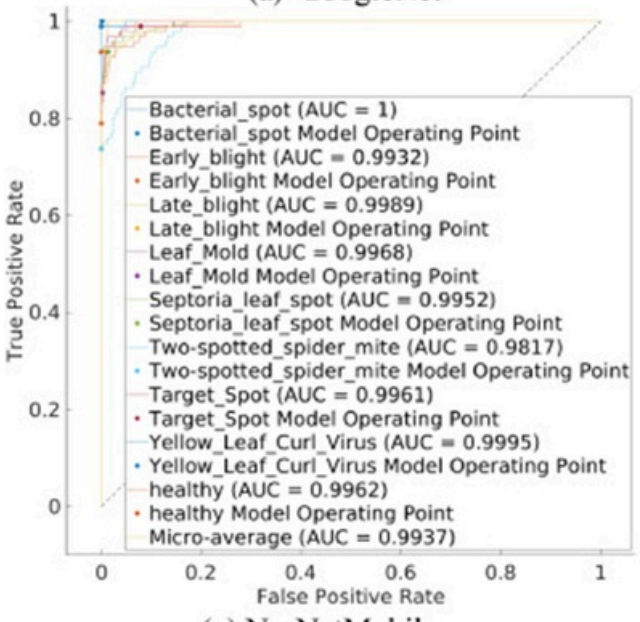
© 2024 The Author(s). | Published by Agro Intelligence Research Press



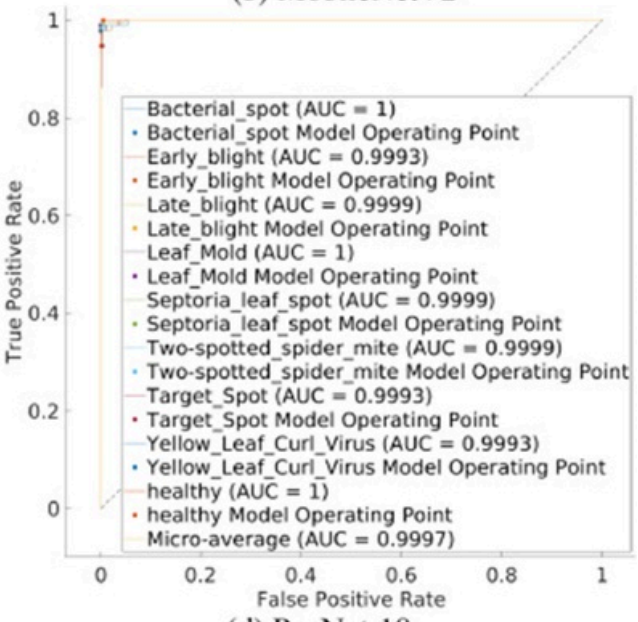
(a) GoogleNet



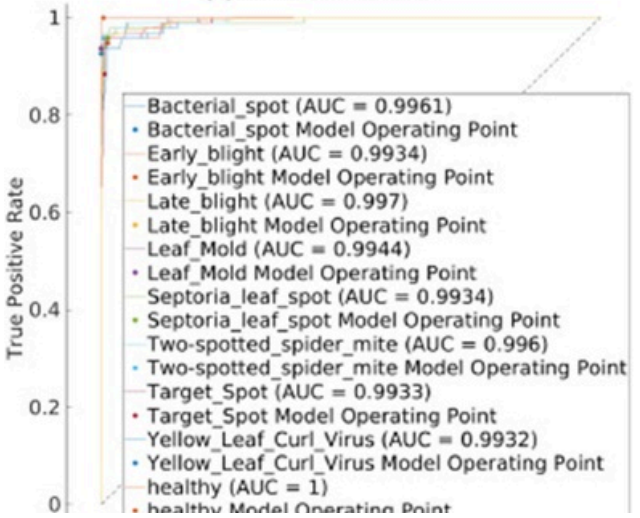
(b) MobileNetV2

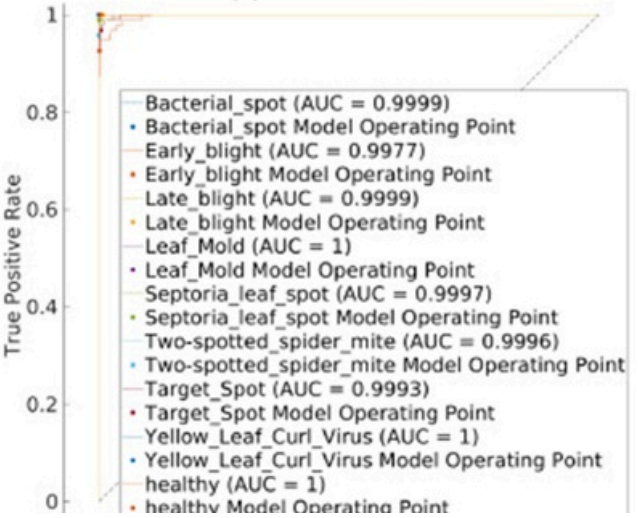


(c) NasNetMobile



(d) ResNet-18





© 2024 The Author(s). | Published by Agro Intelligence Research Press

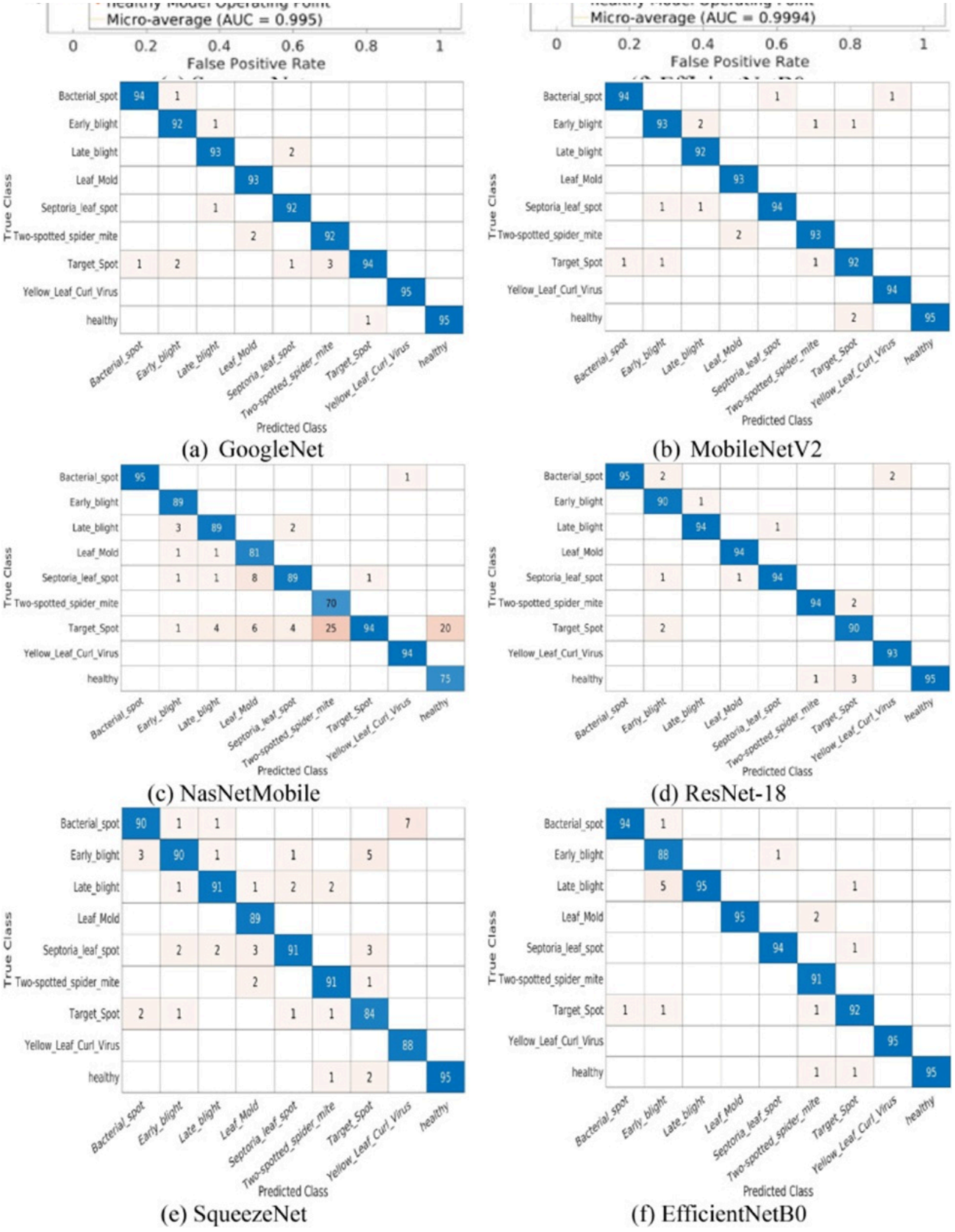


Figure 7. Confusion matrices cross-referencing predicted class labels against actual ground-truth categories for (a) GoogleNet, (b) MobileNetV2, (c) NasNetMobile, (d) ResNet-18, (e) SqueezeNet, and (f) EfficientNetB0.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

The NasNetMobile confusion matrix **(Figure 7c)** immediately reveals that target spot disease was the single largest contributor to the architecture's diminished accuracy. Instances of target spot were incorrectly shunted into the two-spotted spider mite bucket (25 erroneous predictions) and the healthy bucket (20 erroneous predictions). The most likely interpretation is that NasNetMobile was unable to extract sufficiently distinctive hierarchical representations of target spot symptoms, instead merging its visual signature with those of two-spotted spider mite and unblemished foliage. The other architectures proved considerably more adept at encoding the unique features of the target spot and keeping it isolated from competing classes. For instance, GoogleNet **(Figure 7a)** and MobileNetV2 **(Figure 7b)** produced correct predictions in the overwhelming majority of cases. That said, GoogleNet misdirected 7 target spot images to alternate categories, predominantly two-spotted spider mite (3 occurrences), early blight (2 occurrences), bacterial spot (1 occurrence), and septoria leaf spot (1 occurrence). Although SqueezeNet **(Figure 7e)** had the sparsest parameter budget of all evaluated models, it nonetheless showed a reasonable ability to separate the target spot. Its misclassification errors, however, were thinly spread across numerous disease classes, a dispersion that pulled its overall accuracy below those of GoogleNet, MobileNetV2, ResNet-18, and EfficientNetB0.

Performance of edge-AI systems

For each hardware arrangement, the time required to classify a randomly selected image from the full set of disease categories was recorded. A further 95 images per class were subsequently processed, bringing the total to 855 across all categories. To ensure a level playing field, the same image set used earlier to evaluate model performance in Section 3.1 was reused here without alteration.

The data assembled in **Table 3** highlight the decisive lead held by the NVIDIA Jetson ORIN AGX operating with SqueezeNet. That specific combination sliced single-image classification latency down to a mere 3.5 milliseconds and tore through the full 855-image batch in a scant 0.3 seconds. Across every platform included in the comparison, SqueezeNet's inference throughput eclipsed all competing classifiers.

Table 3. Classification speed outcomes across the CNN models and the four hardware configurations deployed in this work.

Model/Images		RPI 4 with Google Coral (in seconds)	RPI 4 (in seconds)	NVIDIA Jetson NANO ORIN (in seconds)	NVIDIA Jetson AGX ORIN (in seconds)
ResNet-18	1 image	0.00168	0.00291	0.000895	0.000489
	855 images	1.436	2.486	0.766	0.418
EffcientNetB0	1 image	0.00147	0.00252	0.000821	0.000546
	855 images	1.259	2.208	0.702	0.467
NasNetMobile	1 image	0.00171	0.00282	0.000997	0.000681
	855 images	1.46	2.408	0.853	0.582
SqueezeNet	1 image	0.00105	0.00153	0.000522	0.000351
	855 images	0.898	1.311	0.446	0.300
MobileNetv2	1 image	0.00116	0.00184	0.000630	0.000437

© 2024 The Author(s). | Published by Agro Intelligence Research Press

	855 images	0.9918	1.576	0.539	0.374
GoogleNet	1 image	0.00163	0.00289	0.000901	0.000501
	855 images	1.39	2.474	0.770	0.428

The analysis was extended to capture GPU and CPU duty cycles **(Figure 8a)** and CPU temperature measurements **(Figure 8b)** for models running on power-conscious hardware. To capture richer operational snapshots, static frames were stitched into a six-minute video. These frames were selected at random from the collective disease pool, and each frame was held on screen for 2 seconds before advancing. Power consumption reached 45 W for the Jetson Orin AGX and 15 W for the Jetson Orin Nano. The RPI 4 similarly drew 15 W. **Figures 8a and 8b** map out the collected inference-performance signals—average CPU load, average GPU load, and mean CPU temperature expressed in degrees Celsius—as recorded on the NVIDIA Jetson Orin Nano, the NVIDIA Jetson Orin AGX, and the RPI 4 linked to Google Coral.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

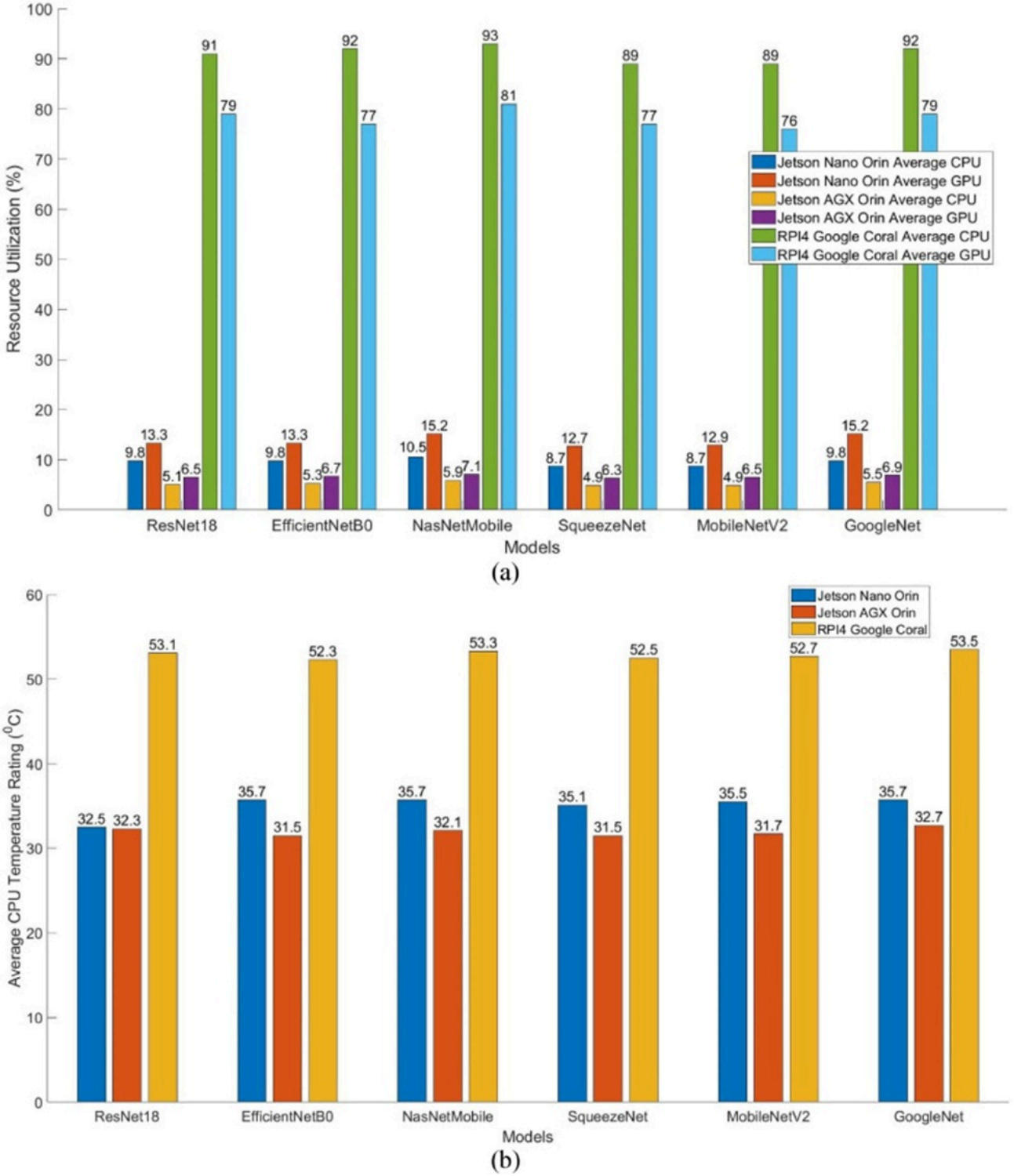


Figure 8. (a) CPU and GPU utilization and (b) average CPU temperature readings for all CNN architectures executing on Edge hardware (NVIDIA Jetson Nano Orin, NVIDIA AGX Orin, and RPI 4 with and without Google Coral).

Table 3 also documents inference throughput figures for the NVIDIA Jetson Orin Nano across all candidate models. SqueezeNet again rose to the top, chalking up 1915 FPS on the Jetson Nano and besting the rest of the field. Differences across performance gauges, such as average CPU utilization, average GPU utilization, and average CPU temperature, were trivial between models, and both CPU and GPU utilization stayed comfortably low on the Jetson Nano. CPU

© 2024 The Author(s). | Published by Agro Intelligence Research Press

temperatures remained well within acceptable limits, ruling out a thermal bottleneck. Turning to the NVIDIA Jetson Orin AGX, SqueezeNet once more claimed the fastest speed at 2849 FPS, with MobileNetV2 placing second at 2288 FPS—both figures reflecting outstanding throughput. Paralleling the Nano results, model-to-model variation in performance indicators was negligible, and temperatures stayed in safe territory. On the RPI 4 Model B outfitted with Google Coral, SqueezeNet achieved 952 FPS, once again outperforming all other architectures. However, the CPU load across all models on this platform fluctuated between 89% and 92%, approaching the upper limit and underscoring the heavy computational burden these models place on the RPI 4. GPU utilization patterns remained stable across models, reflecting the persistent heavy demand for GPU-accelerated computation during object classification. CPU temperatures, however, remained safely below critical thresholds, a benefit enabled by the PWM fan, which maintained stable thermal conditions.

Figure 9 plots the mean electrical power drawn by each device during the algorithms' run. The theoretical power absorbed under different operating modes does not necessarily match the actual energy dissipated during execution. The recorded numbers confirm the higher energy appetite of the NVIDIA Jetson Orin AGX, an outcome fully in line with its vastly superior classification throughput in FPS. Across the tested platforms, power draw showed only minor variability between models, with measured values clustering tightly for each model.

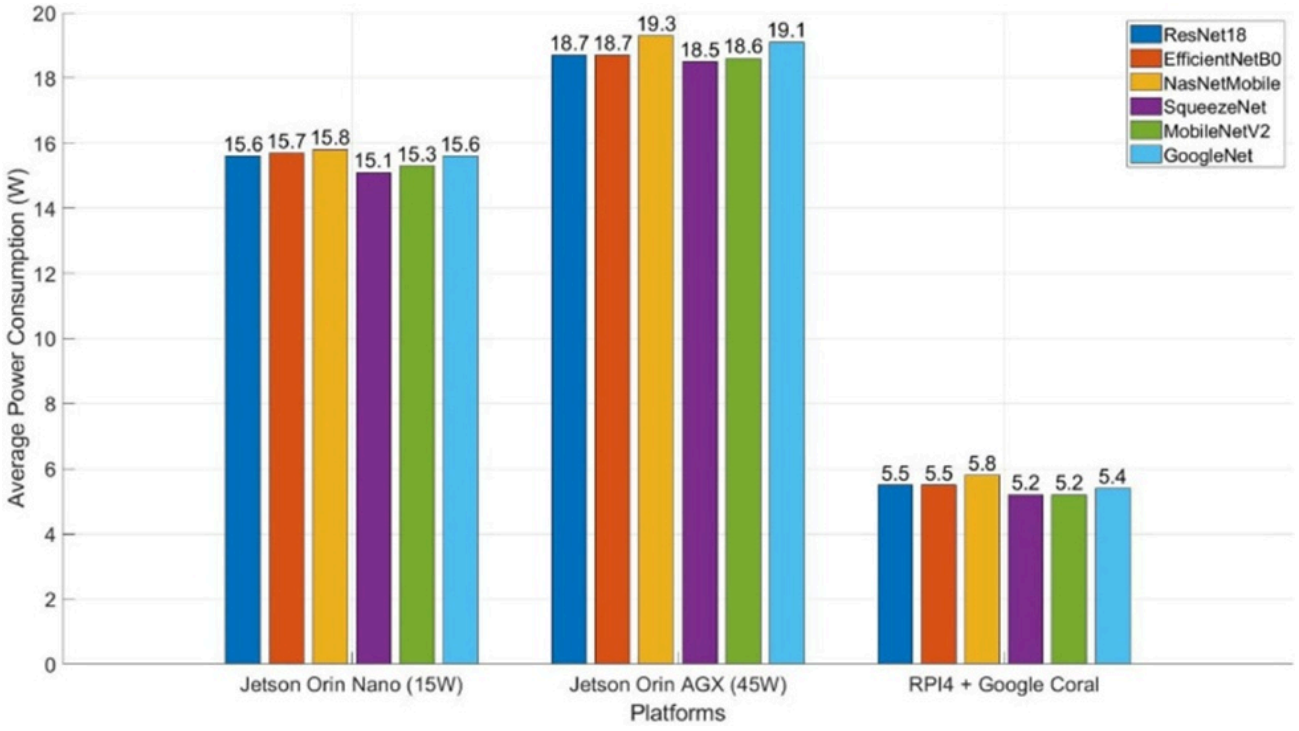


Figure 9. Mean power consumption observed on each Edge device during algorithm execution, broken down by CNN model.

A further angle of comparison, cost relative to FPS, is explored in **Table 4** with SqueezeNet serving as the standard. When the additional expense of the Google Coral accelerator module is folded in, the RPI 4 paired with Google Coral still emerges as the most well-rounded option in the cost-versus-speed calculus. In actual deployment, Google Coral serves as a USB-tethered acceleration peripheral and cannot operate independently as a real-time object detection engine. Consequently, its operation requires the procurement of a companion embedded platform [50], such as an RPI, which increases the aggregate system price. The NVIDIA Jetson Orin AGX, while delivering best-in-class performance, commands a price point that puts it out of reach for strictly cost-limited solutions. The NVIDIA Jetson Orin Nano, by contrast, occupies a more attractive position on the cost-to-FPS curve, making it a highly competitive alternative.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Table 4. Cost-efficiency comparison across the embedded platforms when running the SqueezeNet classification model.

Device	Cost/FPS	FPS	Cost ($) starting at
NVIDIA Jetson AGX Orin	0.7	2849	1999
NVIDIA Jetson Nano Orin	0.26	1915	499
RPI 4 Model B (8GB) with Coral	0.14	952	135

A deeper dive into the performance data presented in **Tables 3 and 4** reveals a substantial performance gap between the tested configurations. The NVIDIA Jetson AGX Orin handily surpassed both the NVIDIA Jetson Orin Nano and the RPI 4 with Google Coral, a margin attributable to the premium-grade hardware foundation of the AGX Orin. When the full spectrum of yardsticks—resource occupancy (CPU and GPU), CPU thermals, FPS, and mean power consumption—is weighed together, the following assessments emerge regarding the suitability of the deep learning platform. Where the paramount concern is maximizing classification velocity while keeping resource draw proportionate to expenditure, the NVIDIA Jetson AGX Orin stands out as the platform of choice. In extensive farming operations where the capital cost of Edge infrastructure can be absorbed, the NVIDIA Jetson Orin AGX represents a well-matched option. The NVIDIA Jetson Orin Nano, for its part, posts solid scores in resource efficiency and processing speed, alongside moderate average power usage and a more affordable price, giving it a distinct competitive edge. The RPI 4 augmented with Google Coral yields slightly diminished performance but remains functional for classification tasks, offering acceptable throughput and fidelity—though it incurs heavier resource consumption and carries potential reliability risks during extended operation. For short-term classification requirements and small-scale farming where budgetary constraints dominate, the RPI 4 with Google Coral may thus be a sensible choice. In sum, the optimal platform flows from the specific demands of the deployment scenario: classification throughput, resource consumption, average power draw, acquisition cost, and the magnitude of the farming enterprise. Each candidate platform offers a unique blend of benefits and concessions, enabling purpose-fit solutions tailored to heterogeneous application requirements.

Zooming out, this study drew on a publicly accessible dataset assembled under controlled, unchanging background conditions that covered multiple tomato leaf disease types. Achieving reliable disease detection for genuine in-field surveillance, however, will require the compilation of a more heterogeneous image corpus spanning diverse backgrounds and lighting conditions, with deep learning models subsequently retrained on that richer collection. Because the evidence points to a direct coupling between network parameter footprint and Edge hardware performance—with slimmer parameter counts yielding higher FPS and diminished CPU/GPU load—there is a strong case for engineering bespoke lightweight architectures calibrated specifically to the demands of tomato disease identification. Such custom networks, inherently simpler and with fewer parameters than off-the-shelf pre-trained models, promise to improve both the speed and reliability of Edge-AI deployments. Lightweight models purpose-tuned for the constrained computational envelopes of Edge devices will unlock faster inference turnaround, curbed energy consumption, and greater operational resilience. Looking forward, upcoming studies will prioritize the development of networks capable of localizing and/or segmenting diseased tissue. This capability could open the door to precision-guided spraying strategies for disease management.

Edge platforms typically furnish considerably less computational horsepower—both CPU and GPU—than cloud-based server infrastructure. Although certain high-end exceptions exist, the NVIDIA Jetson Orin AGX, among them, with its powerful but costly silicon, the vast majority of Edge devices operate under stringent computational ceilings. Deep CNN architectures impose heavy arithmetic demands during inference, which can lead to sluggish response times and place a hard cap on achievable model complexity in resource-constrained settings. Attempting to field large CNN architectures on Edge hardware may therefore be unrealistic, forcing a compromise between accuracy and efficiency. Memory ceilings and the imperative of energy thriftiness place additional shackles on the deployment of CNN models at the Edge. Bulky models with enormous parameter counts consume substantial memory to remain resident. Power drain, meanwhile, is an acute concern, as running computationally intensive CNN workloads can rapidly deplete the batteries that supply most Edge devices.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Scalability presents yet another obstacle, owing to the logistical complexity of provisioning, managing, and maintaining a large distributed population of Edge nodes executing CNN pipelines. These constraints require careful forethought when deploying CNNs on Edge hardware, particularly for missions such as early-stage tomato disease detection. High-performance deep learning CNNs lean heavily on substantial computing infrastructure—GPU-equipped machines—to reach top-tier accuracy. But their voracious processing appetite renders them a poor match for embedded systems starved of computational headroom. This mismatch poses a challenge for deploying CNNs on the kinds of Edge devices commonly used in precision agriculture, which often lack the processing capacity to run such elaborate models. Lightweight CNNs, engineered from the ground up for efficiency, are thus a far more natural fit for these platforms. The present investigation placed four hardware platforms, each varying in computational endowment and purchase price, under empirical scrutiny to determine how lightweight CNN models fare on each. The results throw into sharp relief the performance-accuracy profile of each hardware platform when paired with compact models, delivering actionable decision support for growers across the size spectrum—small, medium, and large—who must select a platform by triangulating speed, power, and cost. One pivotal insight from this work concerns safeguarding data confidentiality and integrity. Because Edge hardware collects and processes information locally, mitigating privacy and security risks becomes non-negotiable. Compact CNNs reduce the need for data transmission across networks, thereby bolstering data protection. Moreover, the findings generated here are relevant to a broad range of agricultural applications, including crop health surveillance, pest identification, yield estimation, and livestock tracking. This study emphasizes the critical importance of identifying optimal hardware-software pairings for such tasks. Edge computing, in conjunction with IoT technologies, dovetails effectively with AI-powered applications and delivers meaningful economic benefits. Still, even though Edge-AI technology has attained a level of maturity, the rollout of novel applications remains a demanding endeavor. Collectively, our work shows that while Edge-AI is sufficiently mature for real-world deployment, unlocking its full agricultural value and navigating the implementation hurdles requires careful specification and tuning of hardware and software configurations.

Conclusion

The central objective driving this research was to gauge how effectively standalone Edge-AI configurations can detect diseases in tomato foliage. In pursuit of that objective, compact deep learning networks trained to recognize and discriminate among tomato leaf diseases were ported onto inexpensive, energy-frugal Edge hardware, and their functional viability as fully autonomous Edge-AI detection units was probed. The lightweight CNN-based deep learning architectures attained peak disease identification accuracies of 98.25 %. When comparing economic efficiency, the RPI 4 paired with a Google Coral AI accelerator posted the lowest cost-to-FPS ratio of 0.14, compared with the NVIDIA Jetson AGX Orin and NVIDIA Jetson Nano Orin, which registered cost-to-FPS ratios of 0.7 and 0.26, respectively. Although the NVIDIA Jetson Orin AGX supplies best-in-class performance, its higher acquisition cost renders it less viable for strictly budget-constrained deployments. The NVIDIA Jetson Orin Nano, meanwhile, delivers a more balanced profile on cost/FPS ratio, resource occupancy (CPU and GPU), CPU thermal behavior, average energy consumption, and deep learning framework compatibility. These factors position it as an attractive candidate for sustained standalone Edge-AI operation in agricultural contexts, even given potential long-term reliability considerations. This study confirms that Edge-AI technology has reached a sufficient level of maturity for practical application; however, navigating deployment complexities and fully tapping its agricultural potential requires deliberate coordination of hardware and software components. Upcoming research will focus on developing custom lightweight CNN algorithms trained on real-world disease datasets and on testing Edge-AI performance under field conditions. Bespoke resource-frugal models, specifically tuned to the constrained compute budgets of Edge devices, will guarantee faster inference, lower energy draw, and superior dependability. Concurrently, extensive on-farm validation trials will be conducted to substantiate the effectiveness of these Edge-AI tools across a range of agricultural environments. Finally, work will focus on crafting intuitive user-facing interfaces that put these technological capabilities within the practical reach of farmers, regardless of operational scale.

Acknowledgements

© 2024 The Author(s). | Published by Agro Intelligence Research Press

This research was funded by the Texas Department of Agriculture (Specialty Crop Block Grant Agreement # GSC2022105). This research is partially supported by the United States Department of Agriculture (USDA)'s National Institute of Food and Agriculture (NIFA) Federal Appropriations under TEX09954 and Accession No. 7002248. This publication is also supported by Texas A&M AgriLife Research, Vegetable and Fruit Improvement Center (VFIC).

Conflict of interest

None

Financial support

None

Ethics statement

None

Received: 17 Feb 2023 Revised: 24 Apr 2023 Accepted: 10 May 2023
Published online: 25 May 2024

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at https://creativecommons.org/licenses/by-nc-sa/4.0/.

References

1. FAO (Food and Agriculture Organization of the United Nations), FAOSTAT Database. 2022. https://www.fao.org/faostat/en/#home.

2. Jin B, Xu X. Forecasting wholesale prices of yellow corn through the Gaussian process regression. Neural Comput Appl. 2024;36(15):8693-710.

3. Xu X, Zhang Y. Corn cash price forecasting with neural networks. Comput Electron Agric. 2021;184:106120.

4. Abade A, Ferreira PA, de Barros Vidal F. Plant diseases recognition on images using convolutional neural networks: A systematic review. Comput Electron Agric. 2021;185:106125.

5. Jin B, Xu X. Price forecasting through neural networks for crude oil, heating oil, and natural gas. Measurement. 2024;1(1):100001.

6. Pardede HF, Suryawati E, Krisnandi D, Yuwana RS, Zilvan V. Machine learning based plant diseases detection: A review. 2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET). IEEE. 2020. p. 212-7.

7. Gadade HD, Kirange DK. Machine learning based identification of tomato leaf diseases at various stages of development. 2021 5th International Conference on Computing Methodologies and Communication (ICCMC). IEEE. 2021. p. 814-9.

8. Kovvuri RR, Kaushik A, Yadav S. Disruptive technologies for smart farming in developing countries: Tomato leaf disease recognition systems based on machine learning. Electron J Inf Syst Dev Ctries. 2023;89(6):e12276.

9. Mokhtar U, Ali MA, Hassenian AE, Hefny H. Tomato leaves diseases detection approach based on support vector machines. 2015 11th International Computer Engineering Conference (ICENCO). IEEE. 2015. p. 246-50.

10. Anubhove MST, Ashrafi N, Saleque AM, Akter M, Saif SU. Machine learning algorithm based disease detection in tomato with automated image telemetry for vertical farming. 2020 International Conference on Computational Performance Evaluation (ComPE). IEEE. 2020. p. 250-4.

11. Harakannanavar SS, Rudagi JM, Puranikmath VI, Siddiqua A, Pramodhini R. Plant leaf disease detection using computer vision and machine learning algorithms. Global Transit Proceed. 2022;3(1):305-10.

12. Nyasulu C, Diattara A, Traore A, Ba C, Diedhiou PM, Sy Y, Raki H, Peluffo-Ordóñez DH. A comparative study of Machine Learning-based classification of Tomato fungal diseases: Application of GLCM texture features. Heliyon. 2023;9(11):e21697.

13. Kumar SA, Sasikala S. Disease detection in tomato leaves using machine learning and statistical feature fusion. 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing, and Automation (ICAECA). IEEE. 2021. p. 1-6.

14. Kiran SM, Chandrappa DN. Plant leaf disease detection using efficient image processing and machine learning algorithms. J Robotics Control (JRC). 2023;4(6):840-8.

15. Govardhan M, Veena MB. Diagnosis of tomato plant diseases using random forest. 2019 Global Conference for Advancement in Technology (GCAT). IEEE. 2019:1-5.

16. Chen M, Chen Z, Luo L, Tang Y, Cheng J, Wei H, et al. Dynamic visual servo control methods for continuous operation of a fruit harvesting robot working throughout an orchard. Comput Electron Agric. 2024;219:108774.

17. Thangaraj R, Pandiyan P, Anandamurugan S, Rajendar S. A deep convolution neural network model based on feature concatenation approach for classification of tomato leaf disease. Multimed Tools Appl. 2023:1-25.

18. Bora R, Parasar D, Charhate S. A detection of tomato plant diseases using deep learning MNDLNN classifier. Signal Image Video Process. 2023:1-9.

19. Kaur P, Harnal S, Gautam V, Singh MP, Singh SP. A novel transfer deep learning method for detection and classification of plant leaf disease. J Ambient Intell Humaniz Comput. 2023;14(9):12407-24.

20. Nawaz M, Nazir T, Javed A, Masood M, Rashid J, Kim J, et al. A robust deep learning approach for tomato plant leaf disease localization and classification. Sci Rep. 2022;12(1):18568.

21. Fuentes A, Yoon S, Kim SC, Park DS. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors. 2017;17(9):2022.

22. Wang Y, Chen Y, Wang D. Convolution network enlightened transformer for regional crop disease classification. Electronics. 2022;11(19):3174.

23. Ferentinos KP. Deep learning models for plant disease detection and diagnosis. Comput Electron Agric. 2018;145:311-8.

24. Roy K, Chaudhuri SS, Frnda J, Bandopadhyay S, Ray IJ, Banerjee S, et al. Detection of tomato leaf diseases for agro-based industries using novel PCA DeepNet. IEEE Access. 2023;11:14983-5001.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

25. Chen X, Zhou G, Chen A, Yi J, Zhang W, Hu Y. Identification of tomato leaf diseases based on combination of ABCK-BWTR and B-ARNet. Comput Electron Agric. 2020;178:105730.

26. Meng F, Li J, Zhang Y, Qi S, Tang Y. Transforming unmanned pineapple picking with spatio-temporal convolutional neural networks. Comput Electron Agric. 2023;214:108298.

27. Ye L, Wu F, Zou X, Li J. Path planning for mobile robots in unstructured orchard environments: An improved kinematically constrained bi-directional RRT approach. Comput Electron Agric. 2023;215:108453.

28. Ray PP, Dash D, De D. Edge computing for Internet of Things: A survey, e-healthcare case study and future direction. J Network Comput Appl. 2019;140:1-22.

29. Badidi E, Moumane K, El Ghazi F. Opportunities, applications, and challenges of edge-AI enabled video analytics in smart cities: A systematic review. IEEE Access. 2023.

30. Abraham A, Dash S, Rodrigues JJ, Acharya B, Pani SK, editors. AI, edge and IoT-based smart agriculture. Academic Press. 2021.

31. Sanchez-Iborra R, Skarmeta AF. TinyML-enabled frugal smart objects: Challenges and opportunities. IEEE Circuits Syst Mag. 2020;20(3):4-18.

32. Anh PT, Duc HTM. A benchmark of deep learning models for multi-leaf diseases for edge devices. 2021 International Conference on Advanced Technologies for Communications (ATC). IEEE. 2021:318-23.

33. Gonzalez-Huitron V, León-Borges JA, Rodriguez-Mata AE, Amabilis-Sosa LE, Ramírez-Pereda B, Rodriguez H. Disease detection in tomato leaves via CNN with lightweight architectures implemented in Raspberry Pi 4. Comput Electron Agric. 2021;181:105951.

34. Tarek H, Aly H, Eisa S, Abul-Soud M. Optimized deep learning algorithms for tomato leaf disease detection with hardware deployment. Electronics. 2022;11(1):140.

35. Moreira R, Moreira LFR, Munhoz PLA, Lopes EA, Ruas RAA. AgroLens: A low-cost and green-friendly Smart Farm Architecture to support real-time leaf disease diagnostics. Internet Things. 2022;19:100570.

36. Zhang Y, Chen M. An IoT-enabled energy-efficient approach for the detection of leaf curl disease in tomato crops. Wireless Networks. 2023;29(1):321-9.

37. Nag A, Chanda PR, Nandi S. Mobile app-based tomato disease identification with fine-tuned convolutional neural networks. Comput Electr Eng. 2023;112:108995.

38. Hughes DP, Salathé M. An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint. 2015;arXiv:1511.08060.

39. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016:770-8.

40. Tan M, Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks. International Conference on Machine Learning (ICML). PMLR. 2019:6105-14.

41. Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018:8697-710.

42. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. arXiv preprint. 2016;arXiv:1602.07360.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

43. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018:4510-20.

44. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015. p. 1-9.

45. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis. 2015;115:211-52.

46. Winzig J, Almanza JCA, Mendoza MG, Schumann T. Edge AI - use case on Google Coral Dev Board mini. 2022 IET International Conference on Engineering Technologies and Applications (IET-ICETA). 2022. p. 1-2. https://doi.org/10.1109/IET-ICETA56553.2022.9971614.

47. Gabdullin N, Raskovalov A. Google Coral-based edge computing person reidentification using human parsing combined with analytical method. Internet Things. 2023;22:100701.

48. Mittal S. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. J Syst Archit. 2019;97:428-42. https://doi.org/10.1016/j.sysarc.2019.01.011.

49. Mittal S, Vetter JS. A Survey of CPU-GPU heterogeneous computing techniques. ACM Comput Surv. 2015;47(4):1-35. https://doi.org/10.1145/2788396.

50. Ojo MO, Giordano S, Procissi G, Seitanidis IN. A review of low-end, middle-end, and high-end IoT devices. IEEE Access. 2018;6:70528-54. https://doi.org/10.1109/ACCESS.2018.2879615.

© 2024 The Author(s). | Published by Agro Intelligence Research Press