Chroma-Sense: A Memory-Efficient Plant Leaf Disease Classification Model For Edge Devices

Kiran K. Kethineni*, Samuel Y. Wu*, Saraju P. Mohanty* and Elias Kougianos†
*Department of Computer Science and Engineering, University of North Texas, USA
†Department of Electrical Engineering, University of North Texas, USA
Email: kirankumar.kethineni@unt.edu, samuelwu@my.unt.edu, saraju.mohanty@unt.edu, elias.kougianos@unt.edu

***Abstract*—CNN models used in disease management systems for disease detection are typically stored and executed on remote cloud servers, communicating with sensors over the internet. This paradigm raises concerns about data security and reliability, especially in remote farmlands. Edge computing introduces a computing layer closer to sensory nodes, enabling more reliable communication and faster results. These edge devices are often resource constrained and cannot run CNN models that need high memory and computation. Since plant disease semantics are simple and consistent across channels, Chroma-Sense proposes processing the R, G, and B channels independently using the same feature extractor. Individual processing reduces the width of the feature extractor, lowering RAM usage and number of computations performed. In addition, reusing the same feature extractor minimizes the parameter count, the Flash memory required to store the model, enabling efficient edge deployment. Proposed Chroma-Sense was trained on a subset of the PlantVillage dataset and achieved a 25% reduction in peak RAM usage and a 60% reduction in flash memory when tested on three different edge devices with varying heap and storage capacities.**

***Keywords*—Smart Agriculture; Artificial Intelligence; Convolutional Neural Network (CNN); Edge Computing; Semantics; TinyML.**

I. Introduction

In the realm of smart agriculture [1], many agricultural practices have been automated with the help of Internet of Things (IoT) and Artificial Intelligence (AI). Subsequently, disease management systems capable of identifying diseases using computer vision techniques [2] were developed. Since these Convolutional Neural Network (CNN) models [3] are computationally intensive, they are traditionally hosted on high-performance computation devices in cloud, with the sensor devices communicate with them using internet technologies. In this architecture, sending the data over a network induces latency and security concerns and in rural scenarios with limited connectivity, it also hinders productivity. To handle these challenges, a new computing framework called "Edge Computing" [4] has been introduced, enabling computations to be performed locally, closer to the sensor devices as illustrated in Fig. 1.

While edge computing offers various benefits by eliminating the need for centralized servers, it also introduces new constraints, such as limited computational resources and low memory (RAM and Flash) availability, which hinder its adoption [5]. Therefore, the ML models developed to detect/classify plant diseases are supposed to satisfy technical constraints without compromising the advantages of edge computing. Many models, architectures have been designed to reduce the model size, computations performed and memory requirements, enabling them to operate efficiently in resource-constrained edge environments. In the same context, this paper "Chroma-Sense" presents a novel Sequential Multi-Channel CNN architecture designed for a plant disease classification, aimed at reducing the memory required to store and run the ML model on an edge device. The organization of rest of the article is as follows: Section II introduces the proposed solution and its novel contributions, while Section III offers a brief review of related research. The proposed method is detailed in Section IV, with experimental results presented in Section V. The article concludes with Section VI, which also outlines potential directions for future research.

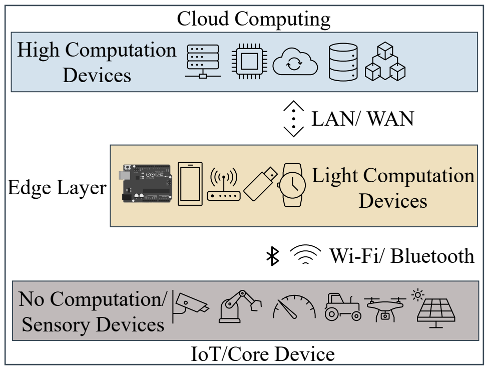


Figure 1: 3 Layered structure of IoT.

II. Novel Contributions of the Current Paper

A. Proposed Solution of the Current Paper

Plant diseases typically manifest as spots, patches, textures on leaves, often characterized by distinct colors. Table I provides an overview of several leaf diseases, compiled based on descriptions available on the PlantVillage website [6].

A detailed examination of features of plant leaf diseases reveals that their distinguishing features are relatively simple and include patterns such as rings, patches, circles, stripes (continuous patches), mosaics (alternating dark and bright regions), powdery textures (noise-like appearance), velvety surfaces, and leaf curls. The CNN architecture required to learn these patterns is relatively simple and does not need to be

Table I: Overview of symptoms for selected plant diseases.

Plant	Disease	Leaf symptoms	Features
Apple	Apple scab	Round yellow spots that enlarge and turn brown.	Semantics: Spots Colors: Yellow, Brown
Apple	Ceder rust	Bright orange or yellow spots surrounded by a red ring with a black spot at the center.	Semantics: Spots, Ring Colors: Yellow, Orange, Red, Black
Grape	Powdery mildew	White, powdery growth appears on leaves.	Semantics: Velvety Color: White
Grape	Esca	Appearance of interveinal chlorosis or necrotic strips	Semantics:Stripes Colors: Brown, Black
Tomato	Leaf mold	Yellow spots that enlarge and turn brown.	Semantics: Spots Colors: Yellow, Brown
Tomato	Mosaic virus	Infected leaves exhibit dark green mosaic	Semantics: Mosaic Color: Dark and light green
Tomato	Septoria spot	Circular spots with dark brown margins and gray centers	Semantics: Spots Colors: Gray, Brown

very deep [7]. Learning these patterns alone is insufficient for effective disease classification; the network must also identify features within the color space to achieve accurate results. In computer vision, images are represented using the Red (R), Green (G), and Blue (B) color channels, where different colors are more prominently expressed in specific channels. An example of a synthesized image of Apple Cedar Rust disease is presented in Fig. 2, with its R, G, and B channels represented in gray scale.

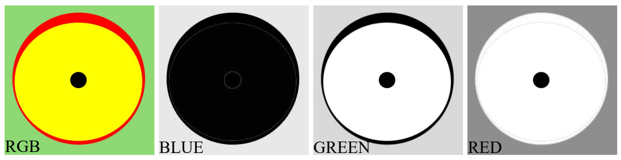


Figure 2: Illustration of features of Apple Ceder Rust disease.

The center of the Apple Cedar Rust disease symptom appears as a black spot, which is dark across all three channels. Since yellow is a combination of red and green, the yellow region surrounding the center appears bright in both the red and green channels. Additionally, the surrounding red ring is bright exclusively in the red channel. From this observation, it can be concluded that plant diseases can be effectively classified by analyzing simpler semantic features across different color channels as illustrated in Fig. 3 and inferring from these findings. This approach eliminates the need for complex CNN architectures designed to learn intricate concentric patterns that are sensitive to color differences.

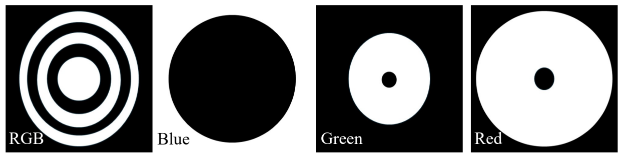


Figure 3: Illustration of features needed to be learned.

As the complexity of the features to be learned decreases, they can be achieved with fewer feature maps at each layer per color channel. From the prior discussion, it is evident that the feature maps to be learned across the R, G, and B color channels are identical. Processing the R, G, and B channels of an image using the same CNN network to extract features reduces the parameter count and the flash memory required to store the model. Furthermore, by processing the R, G, and B channels sequentially, the RAM needed to store intermediate activations and feature maps is significantly reduced. This approach also decreases the cumulative number of Multiply-Accumulate (MAdd) operations by 60%, thereby speeding up inference. Thus, the Serial Multi-Channel Processing proposed in this paper contributes to a memory-efficient CNN model for edge devices.

B. *Novelty and Significance of the Proposed Solution*

The novel contributions of Chroma-Sense are as follows.

1) Memory Efficiency: This article proposes processing the R, G, and B channels in the image individually and sequentially, which reduces the width of the CNN model by about two-third and significantly decreases the RAM required.
2) Reduction in parameter count: The plant leaf disease-specific design allows reuse of the CNN feature extractor across all color channels, reducing the parameter count while simplifying and speeding up inference.
3) Simplified Semantic Feature Extraction: As the CNN network processes grayscale images of each channel, it is compelled to learn shapes and textures [8], leading to more effective feature extraction.
4) Explainability: By extracting channel-specific features and stacking them for classification, the proposed architecture embeds spatial information, making the decision-making process transparent and explainable [9].

III. Related Prior Works

With the development of CNN architectures for effective image classification, such as LeNet-5 and AlexNet, researchers have increasingly utilized these models for classifying plant diseases. In [3], the authors presented the results of AlexNet and GoogLeNet on a plant disease dataset. Similarly, [10] provided results for ResNet on a similar dataset. Multi channel network ensemble in [11] processes features from R, G, B channels, Gabor filters, and PCA, to classify images through majority voting across these channels.

While the aforementioned methods achieve good accuracies, they are computationally intensive and are not specifically designed for edge devices, need lot of RAM to execute . The articles [12] presented a CNN model with very few layers to fit in edge devices but does not propose any architectural changes to optimize edge performance. The authors of [13] introduced an encoder-decoder-based classification model with reduced parameters, specifically designed for binary classification. In contrast, novel architectures with fewer parameters and reduced computational requirements, with moderate network width, were introduced in EfficientNet [14] and MobileNet [15]. The introduction of separable convolutions and squeeze-and-excite mechanisms made these architectures suitable for use on edge devices. The application of these methods for plant disease classification was presented in [16] and [17]. A brief overview of the related works is summarized in Table II. Along similar lines, the proposed method incorporates parameter reduction techniques from edge-optimized models and introduces an innovative serial multi-channel processing approach, designed for plant disease classification with a focus on reducing memory usage.

Table II: Relevant literature on plant disease detection.

Research	Method adopted	Remark
Mohanty et al. [3]	AlexNet, GoogLeNet	Models are generic and not optimized for edge devices.
Archana et al. [10]	ResNet-50	Models are generic and not optimized for edge devices.
Peker [11]	Multi channel network ensemble	Cumulative features across R,G,B channels are not considered.
Rakib et al. [12]	A reduced CNN model with layer quantization	Layers are reduced and optimized for edge devices, but not the architecture.
Bedi and Gole [13]	Convolutional auto-encoder	Optimized for plant diseases but limited to binary classification.
Chowdhury et al. [16]	EfficientNet	Models are optimized for edge devices but not for plant diseases.
Ashwinkumar et al. [17]	MobileNet	Models are optimized for edge devices but not for plant diseases.
Chroma-Sense	MobileNet style parameter reduction with serial multi channel processing	Models are optimized for edge devices and for plant diseases.

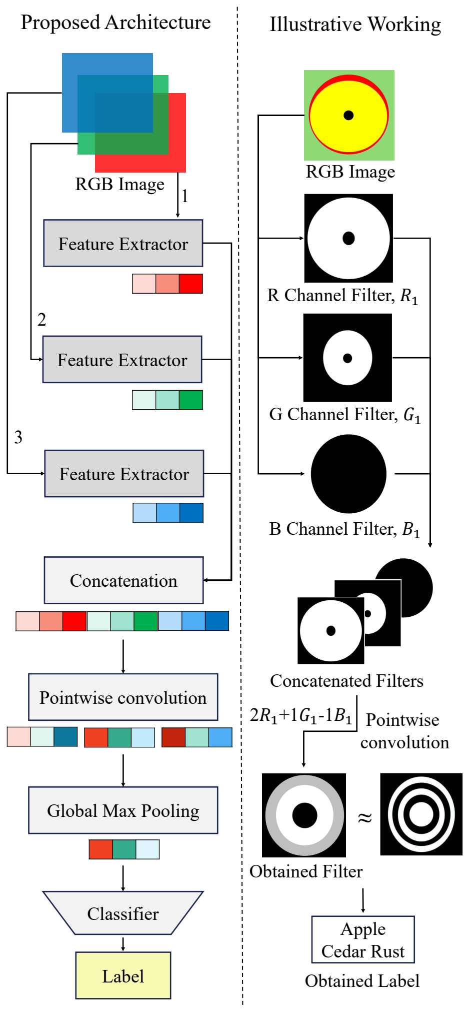


Figure 4: Proposed architecture for serial processing.

IV. Proposed Method

In the edge computing paradigm, computation is performed closer to sensory devices, resulting in limited computing resources being available. To operate within this constrained environment, the ML model's parameter count must be small enough to fit within the Flash memory. Additionally, the activations per layer should be minimized to ensure they fit within the available SRAM. Traditionally, input images of size 96x96x3 are used for edge applications. Additionally, all model weights and biases are quantized to Int8 instead of Float32 to enable efficient deployment on edge devices. This paper adopts Separable2D convolutions, as proposed by MobileNet [18], and proposes processing the R, G, and B channels individually using the same feature extractor, as illustrated in Fig. 4.

The disease semantics for each color channel are simple, such as circles, rings, dots, and patches, but at different scales. By processing the channels individually, feature maps can be obtained with reduced computational cost. Additionally, the semantics are the same across the channels, but their combination required for inference differs depending on the disease. So, this paper proposes use of same feature extractor across all channels as shown in the Fig. 4. Since the same model is used, the inputs are fed sequentially, resulting in "Serial Multi-Channel Processing". As mentioned, to perform detection, we need a mask that contains information from all channels. Therefore, we concatenate the feature maps from the R, G, and B channels and apply point-wise convolution. This generates new feature maps by performing a weighted linear sum of the concatenated features, with attention given to the most prominent channels, an example of such filter creation is presented in Fig. 4. Thus, by stacking and generating comprehensive filters with the help of attention, the correlation lost due to independent processing of channels is restored, along with the spatial correlation.

With the appropriate filter generated, the model will be able to identify all locations in the leaf with disease presence. Plant diseases do not have strict constraints on where they appear on the leaf or how many instances are present. They tend to appear randomly. Due to the nature of the diseases, the disease patterns are not strictly uniform. For example, the outer ring of a disease symptom may not always form a perfect circle or be evenly spaced from the inner circle. Sometimes, the borders may merge. Under these circumstances, applying Global Average Pooling would result in misclassifications as presented in Fig. 5. Image 1 contains multiple disease instances with varying activations, where average pooling resulted in 0.561 and max pooling resulted in 1. In contrast, Image 2 has only two instances, resulting in 0.2 for average pooling and 1 for max pooling. This variation in activations due to average pooling can lead to misclassifications, and when the model is quantized to Int8 as part of TinyML, the error is magnified. Therefore, we propose the use of "Global Max Pooling" to ensure that the model focuses on the spots that align well with the filter, thereby enhancing the model's accuracy.

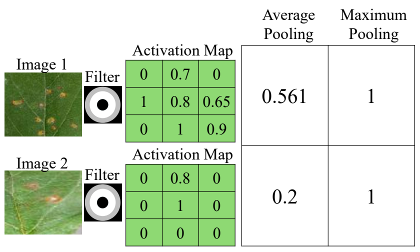
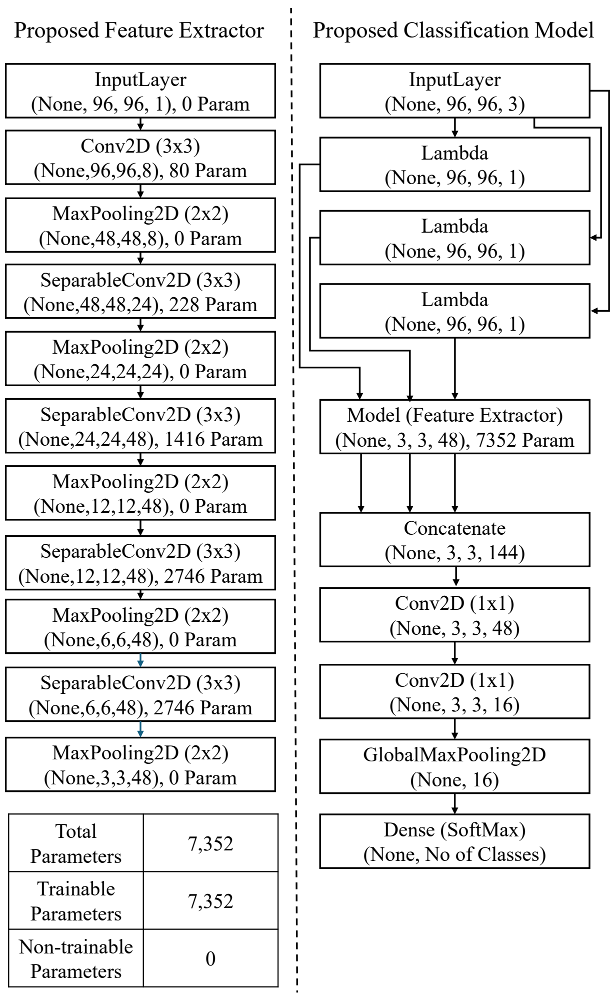


Figure 5: Comparison between average and max pooling.

Multiple networks were designed by reducing the depth and width of the layers while maintaining explainability. The optimal classification model, together with the proposed feature extractor, is shown in Fig. 6. For simplicity, batch normalization layers have been omitted from the diagram. While employing SeparableConv2D, the proposed model aims to reduce the number of parameters required. Since each color channel is processed individually, extracting just 32 features has provided satisfactory results, significantly reducing the RAM required to store intermediate activations. At this low number of features, the linear bottleneck with inverse residual connections would not provide significant benefits, and thus, they were not used in the model. Since the model uses the same feature extractor three times, the parameter count is significantly reduced, leading to a smaller model size and less Flash memory required to store the model.

V. Experimental Verification

The proposed Chroma-Sense was developed using Python and TensorFlow for the CNN classification model, with TensorFlow Lite used to quantize the model to Int8. It was individually validated on 4 plant types, using a total of 18,000 images across 18 classes from the PlantVillage dataset [19]. Images from the original dataset were cropped and split in a 80:20 ratio while ensuring that disease semantics were the only consistent feature across all images, enabling the model to effectively learn the intended patterns. The proposed model was trained on images of Apple, Tomato, Grape, and Corn plants. Grad-CAM results for the model trained on 96×96×3 images with F32 parameters are presented in Figs. 7 to 10, respectively. The trained models were converted to


Proposed Feature Extractor
InputLayer (None, 96, 96, 1), 0 Param
Conv2D (3x3) (None,96,96,8), 80 Param
MaxPooling2D (2x2) (None,48,48,8), 0 Param
SeparableConv2D (3x3) (None,48,48,24), 228 Param
MaxPooling2D (2x2) (None,24,24,24), 0 Param
SeparableConv2D (3x3) (None,24,24,48), 1416 Param
MaxPooling2D (2x2) (None,12,12,48), 0 Param
SeparableConv2D (3x3) (None,12,12,48), 2746 Param
MaxPooling2D (2x2) (None,6,6,48), 0 Param
SeparableConv2D (3x3) (None,6,6,48), 2746 Param
MaxPooling2D (2x2) (None,3,3,48), 0 Param
Proposed Classification Model
InputLayer (None, 96, 96, 3)
Lambda (None, 96, 96, 1)
Lambda (None, 96, 96, 1)
Lambda (None, 96, 96, 1)
Model (Feature Extractor) (None, 3, 3, 48), 7352 Param
Concatenate (None, 3, 3, 144)
Conv2D (1x1) (None, 3, 3, 48)
Conv2D (1x1) (None, 3, 3, 16)
GlobalMaxPooling2D (None, 16)
Dense (SoftMax) (None, No of Classes)

Total Parameters	7,352
Trainable Parameters	7,352
Non-trainable Parameters	0

Figure 6: Proposed model for classification.

Int8 parameters, quantized using TensorFlow Lite, a TinyML framework, to reduce the model size for efficient deployment on edge devices. The Int8 quantized model achieved validation accuracies of 93% for Apple while Tomato, Grape and Corn attained 89%, 96% and 93% , as shown in Table III.

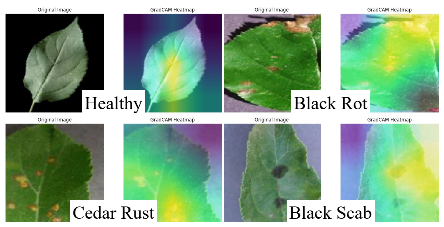


Figure 7: Grad-CAM results for the Apple dataset.

For testing the deployments, we chose the OpenMV H7 with minimal memory, the OpenMV H7 Plus with more memory, and the Arduino Nicla Vision with moderate memory and dual cores, as described in Table IV. These devices, initially load models from flash memory to frame buffer and utilize heap memory to perform inference. So, the size of the model that

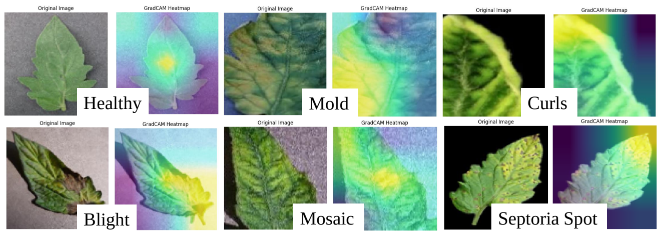


Figure 8: Grad-CAM results for the Tomato dataset.

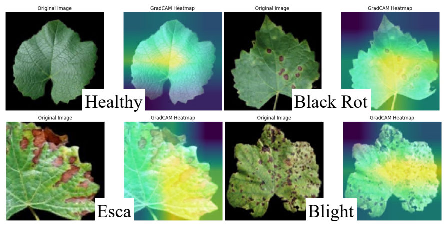


Figure 9: Grad-CAM results for the Grape dataset.

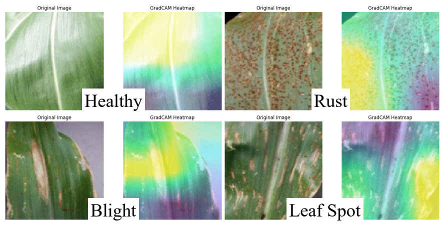


Figure 10: Grad-CAM results for the Corn dataset.

Table III: F1 Scores for different crop types.

Apple	F1 Score	Tomato	F1 Score
Healthy	0.93	Healthy	0.90
Black Rot	0.93	Mold	0.88
Black Scab	0.95	Curls	0.88
Cedar Rust	0.91	Blight	0.91
		Mosaic	0.90
		Septoria Spot	0.87
Grape	**F1 Score**	**Corn**	**F1 Score**
Healthy	0.97	Healthy	0.94
Black Rot	0.95	Rust	0.91
Esca	0.96	Blight	0.92
Blight	0.96	Leaf Spots	0.95

Table IV: Comparison of edge devices tested for the models.

Edge Device	Heap Memory	Flash Memory	Processor
OpenMV H7	256KB	128KB	Cortex-M7
OpenMVH7 Plus	4MB	32MB	Cortex-M7
Arduino Nicla Vision	256KB	16MB	Cortex-M7 + Cortex-M4

Table V: Enhancements incorporated into different models.

Model	Enhancement
Conv2D	Standard Conv2D layers with no optimization.
Grouped Conv	Reduced parameters through group convolution.
MobileNet [15]	Reduced parameters, RAM by using depthwise separable convolution.
MobileNetV2 [18] (Stride 2 used)	Improved expressive power while reducing feature map size and memory usage.
MobileNetV3 [20] (Stride 2 used)	Attention mechanisms helped reduce feature maps and memory usage, parameters.
EfficientNetV2 [14]	Fused-MBConv improved training speed and increased computations.
SqueezeNet [21]	Fire modules reduced parameters but required more memory for expansion.
ShuffleNet [22]	Enhanced feature cross-talk improved expressive power and reduced parameters, but did not save memory.
Squeeze and Excitation [23] (Stride 2 used)	Attention mechanisms enhanced expressive power and reduced memory usage, but increased computations.
Multi-Channel CNN	Serial processing helped reduce memory usage, parameters, and computational requirements.
Chroma-Sense (Stride 1 used)	Reduced memory usage, parameters, and computations with serial processing and separable convolutions.

can be run on the device is constrained by the available memory. Prior works discussed in Section III adopted prominent CNN architectures with varying depths and widths while this paper proposes a serial multi-channel processing architecture with a different depth and width. To make a fair comparison, rather than directly comparing the exact works, we developed models based on the architectures from the discussed works but with a CNN network that is three times the width of the network proposed in Section IV to accommodate the simultaneous processing of the R, G, and B channels, while maintaining the proposed depth. A brief description of each model, along with its proposed enhancements, is presented in Table V. The results of the developed models, tested on different edge devices, are presented in Table VI.

The presented models were tested on the same dataset for 30 epochs with batch normalization. For some models, the .tflite files generated were larger than the available flash memory, while for a few other models that fit within the flash memory, the heap memory required to run them exceeded the available heap memory. So, many models were unable to run on the H7 and Nicla Vision, and therefore, no frames processed per Second (FPS) were reported for them. The H7 Plus had ample memory and was able to run all the models, with Chroma-Sense achieving the highest FPS. In the case of Apple, the disease semantics exhibit considerable variation, while Tomato diseases have more subtle differences, which is reflected in the reported accuracies. ShuffleNet achieved higher accuracies on the Apple dataset but not on the Tomato dataset, indicating that the model struggles with generalization. The proposed model was able to strike the right balance between memory usage, computations, and generalization, demonstrating its suitability for edge deployments. MobileNetV3 is the model that most closely resembles the proposed model in terms of memory usage and generalizability, The proposed method achieved a 25% reduction in peak RAM needed for inference, 60% reduction in flash memory needed to store the .tflite file while maintaining similar accuracy and latency.

Table VI: Results comparison of models for edge deployments.

Model	Acc.Int8 Apple	Acc.Int8 Tomato	RAM	Flash Memory	Parameter Count	Madds	Frames/Sec: H7	Frames/Sec: H7 Plus	Frames/Sec: Nicla Vision
Conv2D	95.2	88.8	336KB	502KB	491K	130M	NA	1.7	NA
Grouped Conv	86.4	71.7	336KB	187KB	169K	43M	NA	4.5	NA
MobileNet [15]	86.7	86.6	341KB	99KB	66K	21M	NA	5.3	NA
MobileNetV2 [18]	95.4	90.4	271KB	203KB	153K	25M	NA	7	NA
MobileNetV3 [20]	92.8	92.6	216KB	138KB	97K	20M	NA	5.3	7.6
EfficientNetV2 [14]	91.8	88.6	240KB	160KB	110K	27M	NA	5.1	6.9
SqueezeNet [21]	97.2	83.5	338KB	80KB	56K	23M	NA	6.4	NA
ShuffleNet [22]	90.2	83	347KB	129KB	88K	13M	NA	2.3	NA
Sque. and Exci. [23]	82.2	79.1	149KB	550KB	527K	32M	NA	5.6	NA
Multi.Ch. CNN	88.8	86.1	160KB	87KB	62K	43M	4.7	4.7	4
Chroma-Sense	93.1	89.3	160KB	54KB	15K	8.3M	8.4	8.4	7.2

VI. Conclusion

This paper presented a novel serial multi-channel processing method for the effective classification of plant diseases on resource-constrained edge devices, it enables on-board disease classification in devices such as drones, facilitating disease containment and automated spraying [24]. The proposed model is designed for 96x96 resolution images and performs classification based on multi-channel feature extraction within the receptive field, considering the device's memory constraints. This low resolution may fail to capture fine details in images with multiple leaves or crops. Moreover, since the model focuses on localized features within the receptive field, it fails to capture global leaf-level features, leading to misclassification when multiple plant types are present in the dataset. To overcome these limitations, future work should focus on enhancing the feature extractor's representational power and designing efficient memory allocation algorithms, allowing the model to process higher-resolution images, such as 224x224, while maintaining efficiency on edge devices.

References

[1] A. Mitra, S. L. T. Vangipuram, A. K. Bapatla, V. K. V. V. Bathalapalli, S. P. Mohanty, E. Kougianos, and C. Ray, "Everything You wanted to Know about Smart Agriculture," 2022. [Online]. Available: 10.48550/ARXIV.2201.04754

[2] V. Singh, N. Sharma, and S. Singh, "A review of imaging techniques for plant disease detection," *Artificial Intelligence in Agriculture*, vol. 4, pp. 229–242, 2020.

[3] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[4] X. Zhang, Z. Cao, and W. Dong, "Overview of Edge Computing in the Agricultural Internet of Things: Key Technologies, Applications, Challenges," *IEEE Access*, vol. 8, pp. 141 748–141 761, 2020.

[5] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE International Conference on Smart Cloud (SmartCloud)*, 2016, pp. 20–26.

[6] "Crops," last accessed April 25 2024. [Online]. Available: https://plantvillage.psu.edu/plants

[7] Y. Li, J. Nie, and X. Chao, "Do we really need deep CNN for plant diseases identification?" *Computers and Electronics in Agriculture*, vol. 178, p. 105803, 2020.

[8] Y. Zhang and M. A. Mazurowski, "Convolutional neural networks rarely learn shape for semantic segmentation," *Pattern Recognition*, vol. 146, p. 110018, 2024.

[9] K. Wei, B. Chen, J. Zhang, S. Fan, K. Wu, G. Liu, and D. Chen, "Explainable Deep Learning Study for Leaf Disease Classification," *Agronomy*, vol. 12, no. 5, 2022.

[10] U. Archana, A. Khan, A. Sudarshanam, C. Sathya, A. K. Koshariya, and R. Krishnamoorthy, "Plant Disease Detection using ResNet," in *Proc. International Conference on Inventive Computation Technologies (ICICT)*, 2023, pp. 614–618.

[11] M. Peker, "Multi-channel capsule network ensemble for plant disease detection," *SN Applied Sciences*, vol. 3, no. 7, p. 707, 2021.

[12] A. F. Rakib, R. Rahman, A. A. Razi, and A. T. Hasan, "A lightweight quantized CNN model for plant disease recognition," *Arabian Journal for Science and Engineering*, vol. 49, no. 3, pp. 4097–4108, 2024.

[13] P. Bedi and P. Gole, "Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network," *Artificial Intelligence in Agriculture*, vol. 5, pp. 90–101, 2021.

[14] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," 2021. [Online]. Available: https://arxiv.org/abs/2104.00298

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[16] M. E. H. Chowdhury, T. Rahman, A. Khandakar, M. A. Ayari, A. U. Khan, M. S. Khan, N. Al-Emadi, M. B. I. Reaz, M. T. Islam, and S. H. M. Ali, "Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques," *AgriEngineering*, vol. 3, no. 2, pp. 294–312, 2021.

[17] S. Ashwinkumar, S. Rajagopal, V. Manimaran, and B. Jegajothi, "Automated plant leaf disease detection and classification using optimal MobileNet based convolutional neural networks," *Materials Today: Proceedings*, vol. 51, pp. 480–487, 2022.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2019. [Online]. Available: https://arxiv.org/abs/1801.04381

[19] A. Ali, "PlantVillage Dataset," 2019, last accessed April 25 2024. [Online]. Available: https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset

[20] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," 2019. [Online]. Available: https://arxiv.org/abs/1905.02244

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016. [Online]. Available: https://arxiv.org/abs/1602.07360

[22] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," 2017. [Online]. Available: https://arxiv.org/abs/1707.01083

[23] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," 2019. [Online]. Available: https://arxiv.org/abs/1709.01507

[24] K. K. Kethineni, S. P. Mohanty, E. Kougianos, S. Bhowmick, and L. Rachakonda, "SprayCraft: Graph-Based Route Optimization for Variable Rate Precision Spraying," 2024. [Online]. Available: https://arxiv.org/abs/2412.12176