Advancing Wildfire Detection: A Comparative Study of Modern Image Recognition Architectures

Arnib Quazi
arquazi@ucdavis.edu

Cynthia Mascarenhas
cynmascarenhas@ucdavis.edu

1 Introduction

Existing wildfire detection models often lack the diversity and robustness needed to accurately identify fires in varied environments, leading to delays and false alarms. With wildfires becoming more frequent and severe in California due to climate change, early detection—especially in remote areas—is crucial to preventing catastrophic losses. This report explores enhancing wildfire detection by customizing MobileViT, EfficientNet V2, and EdgeNeXt models, along with developing custom CNN architectures, with the aim to outperform MobileNetV3 in both accuracy and latency.

2 Methodology

This study employed a wide set of state-of-the-art (SotA) deep learning architectures, all of which were adapted to incorporate a dual-head design for multi-task learning (MTL). MTL optimizes a single model for multiple related tasks simultaneously, leveraging shared representations to enhance generalization and robustness. This approach mitigates overfitting, reduces redundancy, and improves efficiency, making it particularly effective in resource-constrained scenarios.

This configuration simultaneously addressed two critical tasks: binary classification (fire/no-fire detection) and multiclass classification (subcategorization within fire/no-fire classes).

2.1 Models Chosen

The selected models were chosen based on their recent emergence in the literature and demonstrated efficacy in image recognition tasks. These include:

- **MobileNetV3**: Used as the baseline to benchmark performance due to its lightweight design and established results in similar applications.
- **MobileViT**: A hybrid model combining convolutional neural networks (CNNs) with vision transformers, selected for its balance between computational efficiency and accuracy (most notably used in the paper we based our project on).
- **EfficientNet V2**: Chosen for its scalable architecture and proven performance in scenarios requiring computational optimization.
- **EdgeNeXt**: Designed for resource-constrained environments, this model offered a compelling trade-off between accuracy and latency and is the most SotA of models in this class.
- **Custom CNN**: A novel architecture specifically tailored for this study, leveraging insights from the baseline models to refine performance under the MTL framework.

Each model's baseline performance provided insights into its limitations and strengths, guiding subsequent modifications aimed at improving overall binary classification accuracy and reliability for real-world deployment.

2.2 Dataset

The Wildfire Dataset, sourced from Kaggle, offers a wide-ranging collection of images for wildfire detection and classification, encompassing both fire and no-fire scenarios. It includes a variety of environmental conditions and perspectives, enabling models to generalize to real-world situations. Beyond binary classification, the dataset introduces subclasses within fire and no-fire categories, supporting fine-grained multi-class tasks. This design enhances model robustness and applicability, making it a valuable resource for training and advancing automated wildfire detection systems.

2.3 Data Preprocessing

The preprocessing pipeline was developed to standardize input data and enhance model generalization across different wildfire scenarios. Images were first resized to 224×224 pixels to align with the input dimensions required by the selected models. Pixel values were then normalized to match the channel-wise distribution of the ImageNet dataset, ensuring compatibility with pre-trained architectures. To improve robustness to environmental variability and mitigate overfitting, random data augmentation techniques, including horizontal flips, were applied during training. This comprehensive preprocessing strategy provided a consistent and optimized input representation for model training and evaluation.

2.4 Experimental Setup

Model development and evaluation were conducted using PyTorch on NVIDIA L4 GPUs, ensuring efficient training and testing. The dual-head architecture, implemented across all models, optimized for both binary and multiclass objectives simultaneously.

2.5 Training Process

The training process began with baseline training, where each model was implemented with a dual-head adaptation. The binary classification head utilized cross-entropy loss to identify fire/no-fire scenarios, while the multiclass head employed the same loss function to classify subcategories. To accelerate convergence and enhance initial performance, we leveraged each of the models' pretrained weights. Following baseline evaluation, the models were further optimized by manually adjusting key hyperparameters, including learning rates, the number of training epochs, network depth, and dropout rates, to enhance their overall performance.

2.6 Evaluation Metrics

Our goal was to maximize binary classification accuracy while minimizing inference latency. Model performance was evaluated using three metrics: binary classification accuracy (detecting wildfires), multiclass accuracy (discerning subcategories within fire/no-fire classes), and inference latency (assessing real-time deployment feasibility on edge devices). These metrics provided a comprehensive evaluation of the models.

Although multiclass classification was secondary, we were influenced by [1], which showed that improvements in multiclass accuracy within the MTL framework can also boost binary classification performance, guiding our optimization strategy.

3 Models

3.1 MobileNetV3

The baseline MobileNetV3 model, implemented as described in [1], served as the starting point for this study. While the model exhibited strong recall, it struggled with precision, leading to an overall performance that was lower compared to other models in the study (see Table 2). These observations highlight a tendency toward higher false positive rates and challenges in generalization, particularly for multi-class tasks.

The baseline results guided the development of subsequent variants designed to improve accuracy and achieve a better balance between precision and recall.

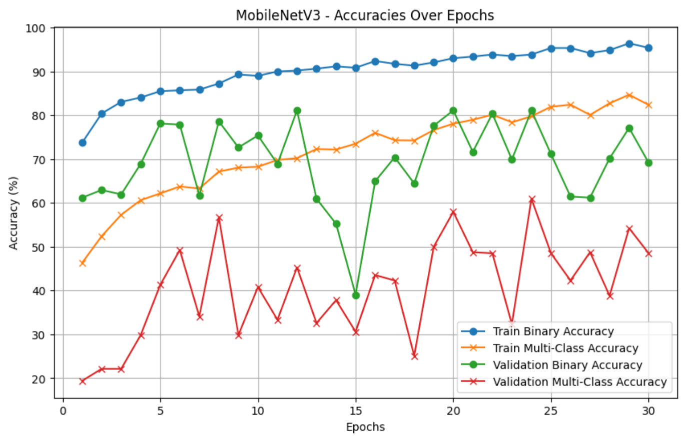

Figure 1: MobileNetV3 Training and Validation Metrics

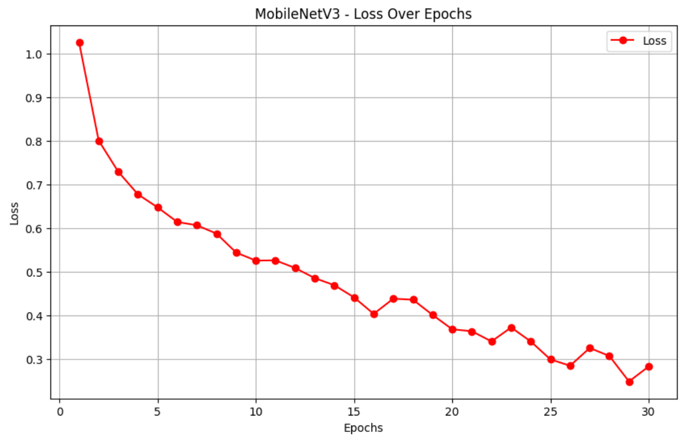

Figure 2: MobileNetV3 Training Loss

3.2 MobileViT

3.2.1 Baseline

MobileViT, a hybrid architecture combining CNNs and vision transformers, was used for wildfire detection and classification. Built on a pretrained MobileViTForImageClassification model, it included custom binary and multi-class classification heads. Trained for 30 epochs with the Adam optimizer (learning rate of 0.01), MobileViT outperformed MobileNetV3. As shown in Fig. 3, it achieved a peak validation binary accuracy of 83.83% (Epoch 26) and multi-class accuracy of 67.41%, demonstrating consistent improvement. Table 2 shows that MobileViT surpassed MobileNetV3 in key metrics: accuracy (85.12% vs. 70.98%), precision (80.25% vs. 57.75%), and F1-score (81.00% vs. 71.46%). MobileViT also achieved a higher ROC-AUC of 0.9257 vs. 0.8257, reflecting better class discrimination. Although MobileNetV3 had a higher recall (93.71% vs. 81.76%), MobileViT's balanced performance across all metrics makes it the more reliable model for wildfire detection and classification.

Models	Version	Learning Rate	Optimizer	Batch Size	Epoch
MobileNetV3	Baseline	0.01	Adam	32	30
MobileViT	Baseline	0.01	Adam	32	30
	Variant A	0.001	Adam	64	30
	Variant B	0.001	Adam	32	30
EfficientNetV2	Baseline	0.01	Adam	32	30
	Variant A	0.002	Adam	32	50
	Variant B	0.002	Adam	32	50
EdgeNeXt	Baseline	0.01	Adam	32	30
	Variant A	0.001	Adam	32	30
	Variant B	0.001	Adam	64	30

Table 1: Hyperparameter Configurations

Models	Version	Accuracy	Precision	Recall	F1-Score	ROC-AUC
MobileNetV3	Baseline	0.7098	0.5775	0.9371	0.7146	0.8257
MobileViT	Baseline	0.8512	0.8025	0.8176	0.8100	0.9257
	Variant A	0.8537	0.8898	0.7107	0.7902	0.9530
	Variant B	0.9439	0.9304	0.9245	0.9247	0.9894
EfficientNetV2	Baseline	0.8732	0.8242	0.8553	0.8395	0.9394
	Variant A	0.8927	0.8042	0.9560	0.8736	0.9773
	Variant B	0.9341	0.9177	0.9119	0.9148	0.9777
EdgeNeXt	Baseline	0.9220	0.9568	0.8365	0.8926	0.9785
	Variant A	0.9317	0.8970	0.9308	0.9316	0.9826
	Variant B	0.9146	0.9366	0.8365	0.8837	0.9801
Custom CNN	Standard	0.8805	0.8819	0.7987	0.8383	0.9523

Table 2: Performance Metrics for all models

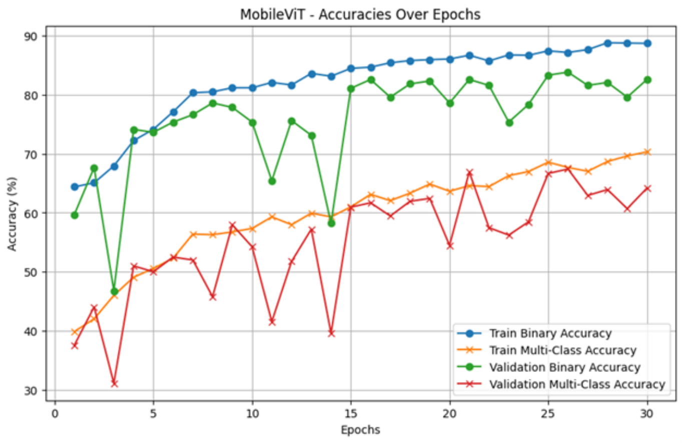

Figure 3: Baseline MobileViT Training and Validation Metrics

3.2.2 Variant A

In Variant A, hyperparameters were adjusted by increasing the batch size from 32 to 64 and reducing the learning rate from 0.01 to 0.001 (Table 1). These changes aimed to enhance stability and learning efficiency. The larger batch size improved gradient estimates, stabilizing training and boosting convergence, while the smaller learning rate ensured more gradual convergence without overshooting. These adjustments optimized learning and improved performance, as shown in Fig. 4.

Compared to the baseline, Variant A saw a slight decline in binary classification performance, with the F1 score dropping from 0.8100 to 0.7902. Precision increased from 0.8025 to 0.8898, while recall decreased from 0.8176 to 0.7107. Accuracy remained stable (0.8512 vs. 0.8537), but the ROC-AUC improved from 0.9257 to 0.9530, indicating better class discrimination. In multi-class classification, accuracy remained similar (0.68 vs. 0.65), with variations in precision, recall, and F1 scores across classes, reflecting a balance between stability and performance. The changes in performance can be attributed to the impact of the hyperparameter adjustments. Increasing the batch size likely led to more stable gradient estimates but also resulted in a slower adaptation to the data, which may explain the decrease in recall. The reduced learning rate promoted more gradual and stable learning, but this slower convergence likely contributed to the decrease in F1 score. However, the higher precision suggests that the model became more conservative in making positive predictions, reducing false positives. The improvement in ROC-AUC reflects better discrimination between classes, highlighting the trade-off between sta-

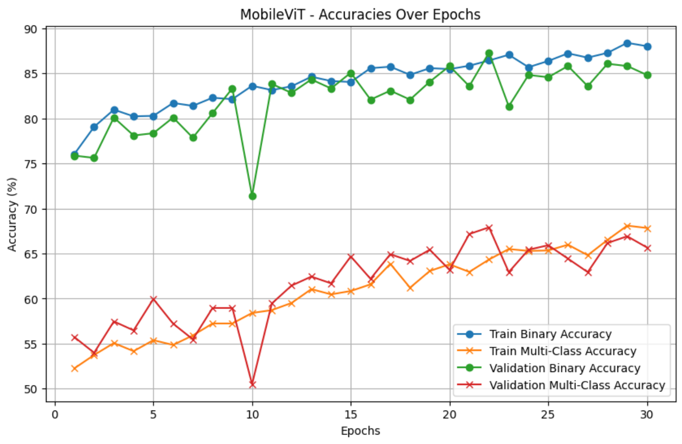

Figure 4: Variant A MobileViT Training and Validation Metrics

bility and sensitivity. These results demonstrate that while the adjustments improved model stability and discriminative power, they also affected the model's ability to identify all positive cases, leading to a nuanced shift in performance.

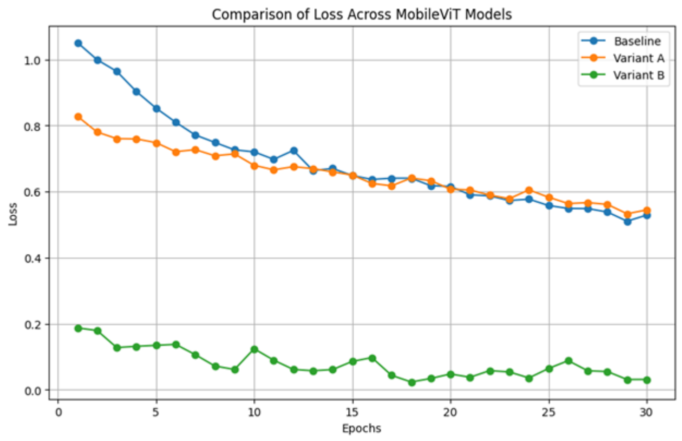

Figure 5: MobileViT Training Loss

3.2.3 Variant B

In Variant B, the only change made compared to the baseline was the reduction of the learning rate from 0.01 to 0.001. The reason for this change was to potentially stabilize training and prevent the model from overshooting the optimal weights, which can sometimes happen with a larger learning rate. When comparing the results with the baseline, we noticed that with the reduced learning rate, the model achieved slower but more stable convergence. This was expected, as the smaller learning rate leads to smaller weight updates, allowing the model to fine-tune its parameters over more iterations. Despite the slower training, the model was able to reach a higher level of accuracy and perform better in terms of generalization. This suggests that the adjustment in the learning rate helped the model refine its understanding of the data, improving its overall performance (see Fig.6).

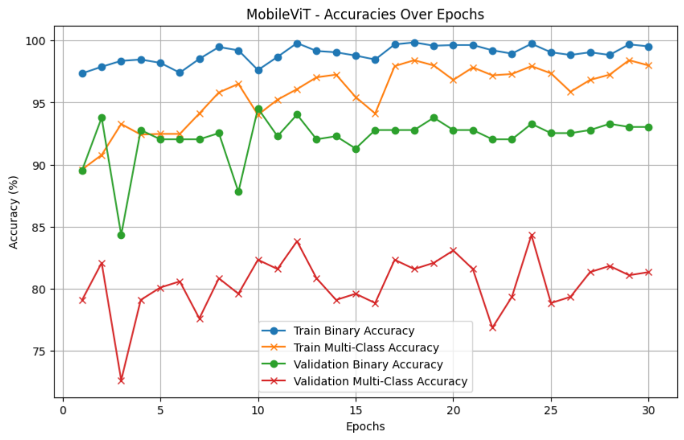

Figure 6: Variant B MobileViT Training and Validation Metrics

3.3 EfficientNetV2

3.3.1 Baseline

The baseline EfficientNetV2 model closely mirrors the MobileNetV3 baseline in hyperparameters and overall architecture, with the primary distinction being its use of the EfficientNetV2 backbone (see Table 1 for the hyperparameter configuration). The baseline EfficientNetV2 model demonstrated clear improvements over MobileNetV3 across key performance metrics. While MobileNetV3 exhibited strong recall, it struggled with precision, reflecting a higher rate of false positives. In contrast, EfficientNetV2 achieved more consistent and stable validation accuracies in both binary

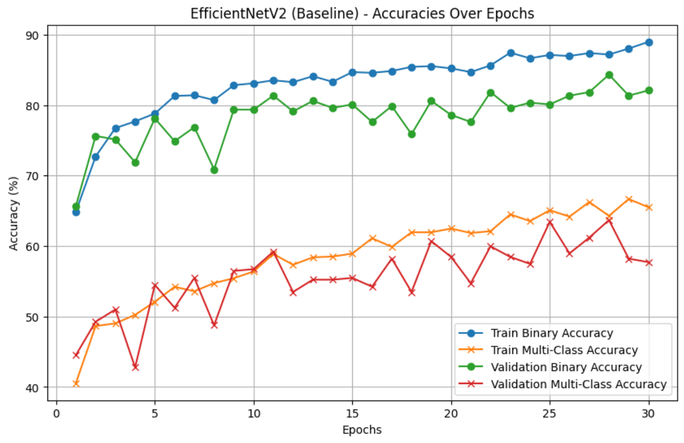

Figure 7: Baseline EfficientNetV2 Training and Validation Metrics

and multi-class tasks, indicating better generalization and reduced overfitting.

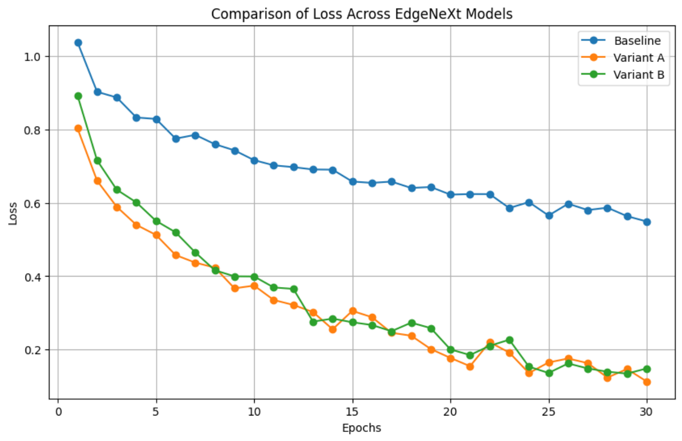

Figure 8: EfficientNetV2 Training Loss

3.3.2 Variant A

The baseline EfficientNetV2 model used a shallow architecture for both binary and multi-class tasks, limiting its capacity. Variant A addressed this by adding hidden layers with Leaky ReLU activations and dropout to both heads. This increased the model's representational power and reduced overfitting. To fully leverage these changes, the training epochs were increased to 50. As shown in Figure 9, Variant A has demonstrably better multi-class accuracy, resulting in further improvements for testing accuracy as seen in Table 2.

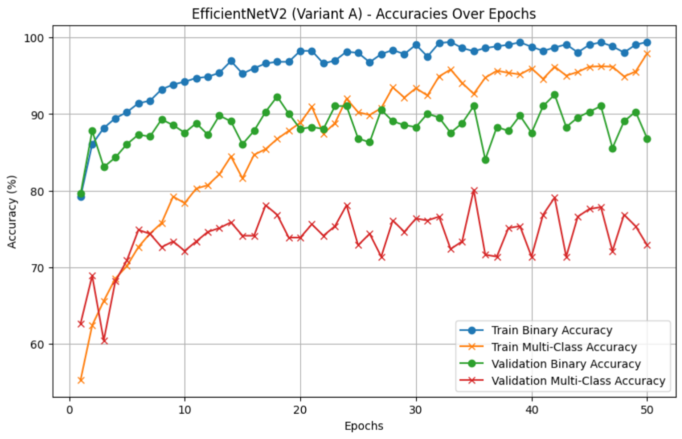

Figure 9: Variant A EfficientNetV2 Training and Validation Metrics

3.3.3 Variant B

Variant B introduced two key modifications on top of Variant A: increased dropout in the binary head for stronger regularization and a shifted loss weight to prioritize the multi-class task.

The increased dropout improved the model's generalization by reducing overfitting (seen in the later epochs of training – Figure 10), leading to even more stable validation accuracy. The re-weighted loss function, inspired by multi-task learning, enhanced the model's feature representations by adding more emphasis to the loss in multi-class predictions. This approach resulted in a more noticeable 4.4% improvement in accuracy.

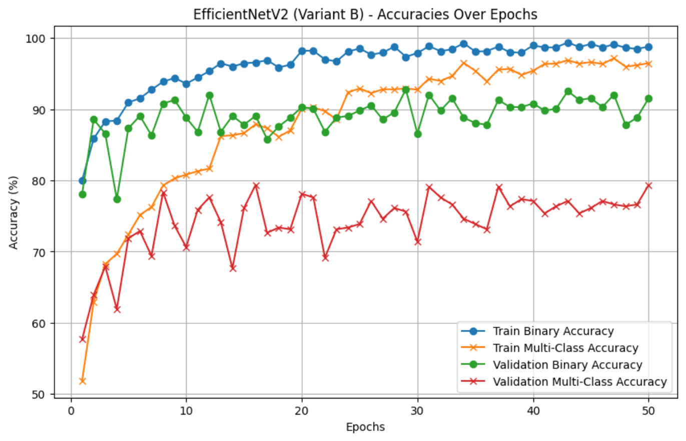

Figure 10: Variant B EfficientNetV2 Training and Validation Metrics

3.4 EdgeNeXt

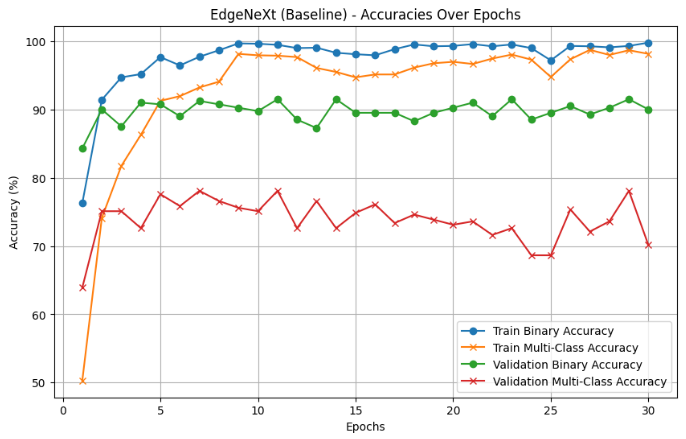

Figure 11: Baseline EdgeNeXt Training and Validation Metrics

3.4.1 Baseline

The baseline EdgeNeXt model was implemented with a lightweight yet powerful backbone designed for efficiency in both binary and multi-class tasks (see Table 1 for the

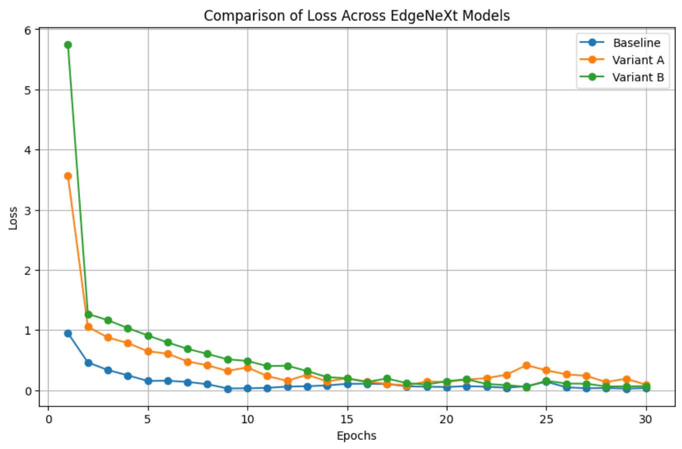

Figure 12: EdgeNeXt Training Loss

hyperparameter configuration). Its architecture builds on advancements in modern ConvNeXt-inspired designs, focusing on balancing between computational efficiency and representational capacity.

Compared to the other baseline models, EdgeNeXt consistently delivered higher performance across most metrics, particularly excelling in precision and overall stability during training. This highlights its ability to generalize effectively while maintaining robust predictions, making it a strong candidate for further refinement in subsequent variants.

3.4.2 Variant A

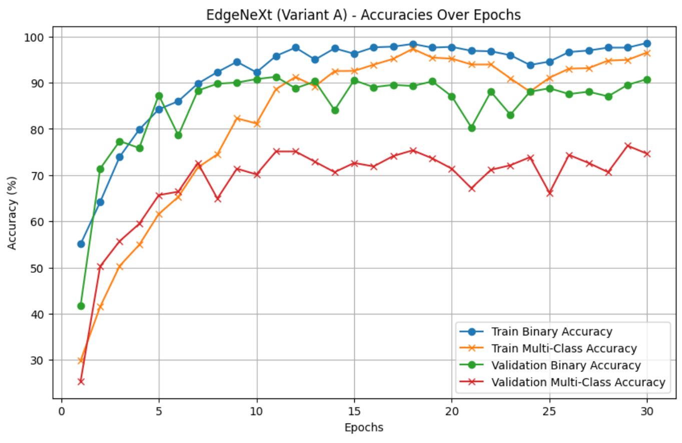

Figure 13: Variant A EdgeNeXt Training and Validation Metrics

Building on the baseline architecture, Variant A introduced additional hidden layers with Leaky ReLU activations and dropout to enhance the model's representational capacity and mitigate overfitting seen in the training accuracy of binary classification. Unlike EfficientNetV2 Variant A, the number of training epochs remained at 30, balancing computational efficiency with improved model performance.

These architectural enhancements resulted in a noticeable improvement in multi-class accuracy and a better balance across precision and recall metrics, as reflected in the performance metrics (see Table 2). Variant A demonstrated the model's ability to generalize more effectively without requiring extended training durations.

3.4.3 Variant B

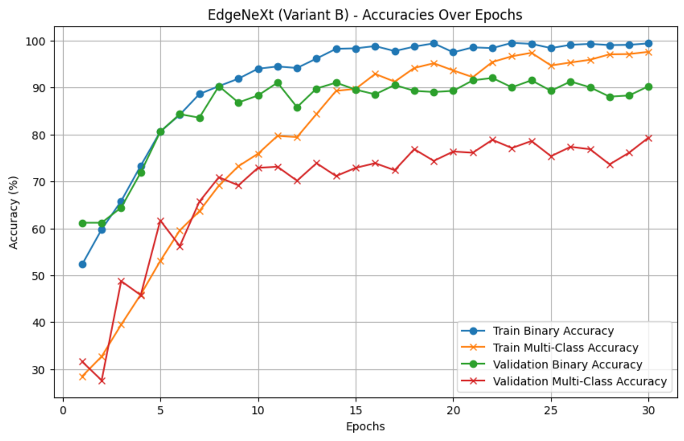

Figure 14: Variant B EdgeNeXt Training and Validation Metrics

Variant B aimed to improve multi-class generalization by making several key adjustments. The multi-class loss was weighted more heavily, the learning rate was increased to accelerate convergence, and a dropout layer was added to the multi-class head to mitigate overfitting, as the training accuracy for multi-class tasks was high, but validation accuracy lagged behind for Variant A.

Although studies were conducted with training epochs beyond 30, these were excluded from the final results due to poor performance and rapid loss increases in later epochs. Despite the modifications, Variant B underperformed compared to Variant A, likely due to a combination of instability from the higher learning rate and over-regularization from the additional dropout layer.

3.5 Custom CNN

The custom CNN features five convolutional blocks as a feature extractor, followed by separate classifiers for binary and multi-class outputs. Each block includes a convolutional layer, batch normalization, ReLU activation, and max pooling, enabling the model to learn hierarchical features from low-level patterns to high-level representations. Batch normalization stabilizes training and improves generalization, while dropout in fully connected layers reduces overfitting. The dual-output design allows shared feature learning, improving computational efficiency and potentially enhancing performance. A sigmoid activation is used for binary classification, while softmax is used for multi-class output across five categories. This architecture effectively addresses the complexity of wildfire detection, enabling accurate fire presence identification and scenario classification for real-world applications. The custom

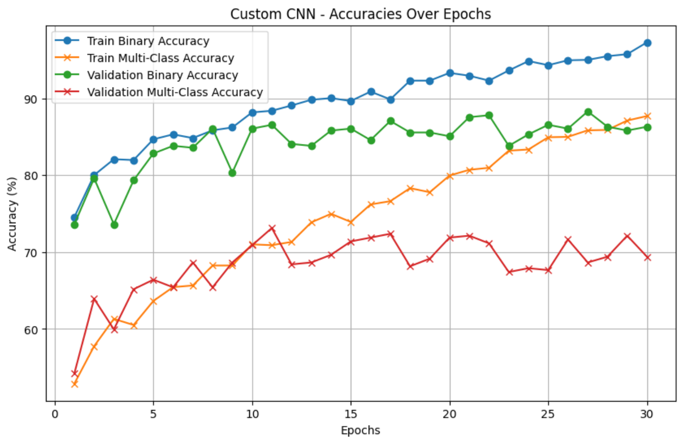

Figure 15: Custom CNN Training and Validation Metrics

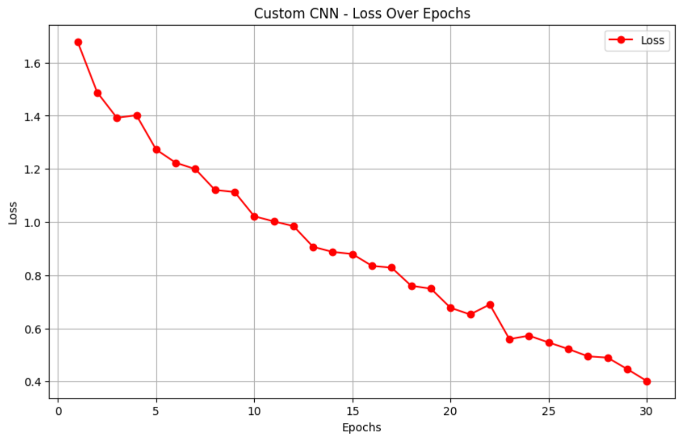

Figure 16: Custom CNN Loss

CNN model showed strong performance in both binary and multi-class wildfire detection tasks. As shown in Fig. 15, it achieved a final binary classification training accuracy of 95.50% and a peak validation accuracy of 88.31%, with an ROC-AUC score of 0.95. The confusion matrix indicated 234 true negatives, 127 true positives, 17 false positives, and 32 false negatives, resulting in high precision (88.19%) and recall (79.87%), with an F1-score of 83.83%. For multi-class classification, the model reached 85.90% training accuracy and 69.40% final validation accuracy, with F1-scores ranging from 0.56 to 0.82. Overall loss decreased from 1.6787 to 0.4898 over 28 epochs (Fig. 16), indicating effective learning. While the results validate the model's design, there is potential to improve multi-class performance and reduce false negatives in binary classification.

4 Inference Time Study

Inference time is a critical metric in model evaluation, as it directly impacts the practicality of deploying models in real-world settings. For wildfire detection, however, inference does not need to be instantaneous since decisions are not typically made on the order of milliseconds or hundredths of a second. In this context, we believe slightly longer inference times are acceptable if they result in improved accuracy and fewer false positives, which are more critical for minimizing resource deployment in the case of false alarms.

As shown in Table 3, the MobileNetV3 baseline outperformed all other models in terms of inference speed, achieving the lowest average and most consistent inference times. While other models, such as EdgeNeXt Variant A and EfficientNetV2 Variant B, demonstrated longer inference times, they offer trade-offs in terms of higher accuracy and better generalization. These results highlight the challenge of balancing inference efficiency with performance, emphasizing that for wildfire detection, the focus should remain on optimizing predictive accuracy and minimizing false positives rather than solely targeting faster inference speeds.

5 Quantization Study

We conducted a post-training dynamic quantization study to explore the feasibility of deploying our models in resource-constrained environments, such as edge devices without GPUs. Due to the custom and non-quantized nature of many layers in MobileViT and EdgeNeXt, we focused on quantizing linear layers only. This approach enabled partial quantization while preserving the core functionality of the models.

The results, shown in Table 4, reveal mixed outcomes. For MobileViT, quantization failed to yield any performance gains in inference time and severely degraded the model's ability to classify images. In contrast, EdgeNeXt showed a significant reduction in inference time, achieving a speedup of approximately 25% with only minimal degradation in accuracy and F1-score. These findings demonstrate the potential of dynamic quantization to make models like EdgeNeXt more viable for real-time applications on edge devices, where computational resources are limited, without requiring substantial sacrifices in predictive performance.

6 Discussion

This study sought to improve both accuracy and inference time for wildfire detection models. While accuracy improved significantly across architectures like EdgeNeXt Variant A and MobileViT Variant B, these gains came at the cost of longer inference times. Dynamic quantization showed promise with EdgeNeXt, achieving faster inference

Models	**Version**	**Average**	**Standard Deviation**	**Median**	**25th Percentile**	**75th Percentile**
MobileNetV3	Baseline	0.012264	0.012061	0.008798	0.008923	0.008594
MobileViT	Baseline	0.017939	0.009024	0.01454	0.017395	0.014233
	Variant B	0.015928	0.001418	0.015205	0.017612	0.014936
EfficientNetV2	Baseline	0.02704	0.012426	0.022856	0.023829	0.02277
	Variant B	0.026832	0.004226	0.025135	0.028681	0.023729
EdgeNeXt	Baseline	0.020121	0.002316	0.019226	0.020257	0.01877
	Variant A	0.020515	0.001725	0.020199	0.02116	0.019337

Table 3: Inference time metrics (measured in seconds) for all baseline models and best performing variants

Model	**Version**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Average Inference Time (s)**
MobileViT	Variant B	0.9439	0.9304	0.9245	0.9247	1.66
MobileViT	Quantized	0.3878	0.3878	1.0	0.5589	1.68
EdgeNeXt	Variant A	0.9317	0.8970	0.9308	0.9316	0.55
EdgeNeXt	Quantized	0.9293	0.8916	0.9308	0.9108	0.41

Table 4: Quantization Results for MobileViT and EdgeNeXt Models

with minimal accuracy degradation, but failed with MobileViT, which saw no runtime benefits and complete loss of classification capability. Expanding the quantization study to include static quantization and lower-precision formats (e.g., int8) could enable deployment on low-cost, resource-constrained edge devices such as Raspberry Pi or Arduino.

The study faced limitations, particularly with hyperparameter tuning, which relied on manual adjustments due to time and resource constraints. Employing Bayesian optimization in future work could further enhance model performance. Additionally, the quantization approach was limited to linear layers, and exploring quantization-aware training could improve both runtime and accuracy. Computational resources also restricted the ability to test deeper architectures or larger datasets, which could enhance model generalization.

Future efforts should focus on optimizing quantization techniques, leveraging hardware-specific tools like TensorRT or ONNX Runtime, and testing the models in real-world deployment scenarios. These improvements would help bridge the gap between accuracy and efficiency, making the models more practical for edge-based wildfire detection systems.

7 Conclusion

The study developed and evaluated state-of-the-art wildfire detection models on a diverse dataset, achieving significant improvements in precision and recall over the baseline with negligible inference time trade-offs. Among the models, EdgeNeXt Variant A emerged as the best performer, offering a strong balance of accuracy (93.17%) and inference time ($0.0205s$). Quantization demonstrated that EdgeNeXt could be effectively deployed on resource-constrained edge devices without significant performance loss. These advancements highlight the potential for scalable, real-time wildfire detection, with the ability to minimize false positives while maintaining efficient inference, making EdgeNeXt Variant A a highly suitable choice for deployment in practical applications. Additionally, the study underscores the importance of optimizing both model accuracy and inference speed to ensure effective wildfire detection in diverse, real-time operational environments. This research paves the way for more robust edge-based wildfire monitoring systems, which could be crucial for early detection and timely response in wildfire-prone regions.

8 Contribution & Acknowledgment

Arnib implemented the baseline for MobileNetV3 and completed the implementation of EfficientNetV2 and EdgeNext, utilizing GPU access. Cynthia implemented MobileViT and the Custom CNN model. The documentation was completed collaboratively. We also would like to thank our TA, Parsa, for always being open to our questions and providing valuable guidance throughout the project.

References

[1] El-Madafri, I.; Peña, M.; Olmedo-Torre, N. The Wildfire Dataset: Enhancing Deep Learning-Based Forest Fire Detection with a Diverse Evolving Open-Source Dataset Focused on Data Representativeness and a Novel Multi-Task Learning Approach. Forests 2023, 14, 1697. https://doi.org/10.3390/f14091697

A Appendix

All the code used for the experiments presented in this paper is available in the GitHub repository linked here. The video presentation can also be found here.