CENTIVE
Conference on Electrical Engineering, Informatics
, Industrial Technology, and Creative Media
ISBN : 978-602-53004-0-0

Forest Fire Detection Leveraging Hybrid Convolutional-Recurrent Models

Raphon Galuh Candraningtyas[1*], Asyafa Ditra Al Hauna[2], Mohamad Nassar [3]
Mie University[1*], Telkom University[2], University of New Haven [3]
[1]1577 Kurimamachi Yachō, Tsu, Mie 514-0102, Japan
[2]Jl. DI Panjaitan No.128, Karangreja, Purwokerto Kidul, Kecamatan. Purwokerto Selatan, Kabupaten Banyumas, Jawa Tengah 53147
[3]300 Boston Post Road, New Haven, West Haven, CT 06516, United States
[*]425005c@m.mie-u.ac.jp

Abstract — Forest fires pose serious environmental and economic risks across tropical, temperate, and boreal regions. Traditional detection methods are often limited in accuracy and adaptability, motivating the use of deep learning for automated solutions. While Convolutional Neural Networks (CNNs) have shown promise, fewer studies have systematically examined hybrid models combining CNN feature extraction with recurrent layers such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU). This study compares CNN-MLP, CNN-RNN, CNN-LSTM, and CNN-GRU architectures on a public forest fire dataset, evaluating classification performance and computational efficiency. Results show that CNN-GRU offers the best trade-off, closely matching CNN-MLP in accuracy while requiring fewer resources. CNN-LSTM provides stable performance, whereas CNN-RNN underperforms and needs refinement. Computational analysis further indicates that CNN-MLP is the the most resource intensive models with over 1 millions parameter. These findings highlight CNN-GRU as a strong candidate for real-time forest fire detection, balancing accuracy and efficiency, and suggest future exploration of adaptive thresholds and transformer-based approaches.

Keywords – Forest fire detection, CNN, RNN, LSTM, GRU

I. INTRODUCTION

Fires are recognized as a major destructive force in forest ecosystems in tropical, temperate, and boreal regions [1], and are considered a global problem that affects vast areas of land [2]. Forest fires, largely driven by human activities, significantly disrupt forest ecosystems by altering nutrient cycles, soil properties, species composition, and biodiversity, while also contributing to soil degradation and nutrient loss [3]. In addition of being driven by human activity, forest fires can also be caused by a natural phenomenon which can often be called wildfires [5]. Fire characteristics such as frequency, size, intensity, and seasonal timing strongly influence forest regeneration and long-term stability of the ecosystem [4]. Given the rapid and destructive nature of forest fires, early and accurate detection is needed to enable early intervention and mitigate environmental and economic damages.

Early forest fire detection relied mainly on conventional ground-based methods such as guard towers placed at elevated locations [7] and the Osborne Fire Finder, a topographic disk tool used to visually detect smoke and flames [8]. However, these techniques were limited by the unreliability of human observers and the harsh conditions of remote fire-prone areas, which reduced their effectiveness [9]. To overcome these limitations, satellite-based systems such as the Moderate Resolution Imaging Spectroradiometer (MODIS) in Canada and the Advanced Very High Resolution Radiometer (AVHRR) in China were adopted [10]. While these systems provided broader coverage, they were constrained by environmental and technical factors, including terrain, cloud cover, and smoke from industrial activities, often resulting in reduced accuracy [7]. To address these challenges, an improved AVHRR-based method was later proposed, which automatically adapts threshold values using historical data to achieve higher fire discrimination with fewer false alarms, successfully detecting over 75% of significant fires in cloud-free regions with fewer than 20% false alarms [6].

Despite the progress of the traditional forest fire detection method by traditional means, their limitation in accuracy, time, and adaptability highlight the need for a more robust detection system. Therefore, re-

searchers have increasingly turned to machine learning and, more recently, deep learning approaches for forest fire detection. For example, UAV-based platforms combined with YOLOv3 have been successfully applied to real-time forest fire monitoring [11], similarly, transfer learning with CNNs has been explored for fire and smoke detection [12]. Furthermore, multimodal fusion methods such as Fire-Net have leveraged both optical and thermal bands in detecting active fires across diverse global regions [13]. Although these studies have established CNN-based models as highly capable for fire detection, most of them focused on static CNN architectures without deeply exploring the integration of recurrent layers or systematically addressing the trade-offs between detection performance and computational efficiency.

While prior studies have demonstrated the effectiveness of Convolutional Neural Networks (CNNs) in detecting forest fires, there has been limited exploration of hybrid architectures that combine CNN feature extraction with recurrent layers such as RNN, LSTM, and GRU. Moreover, few works have systematically compared these approaches in terms of both classification accuracy and computational efficiency, which is crucial for real-time forest fire detection scenarios. In this study, we investigate the extent to which CNNs can effectively extract discriminative features from forest fire datasets, examine how recurrent-based models such as RNN, LSTM, and GRU perform in distinguishing fire from non-fire cases, and analyze the trade-offs between predictive accuracy and computational complexity across the developed models. Building upon these objectives, the main contributions of this work are as follows:

a) Proposing a CNN-MLP baseline and recurrent-enhanced architectures (CNN-RNN, CNN-LSTM, and CNN-GRU).
b) Benchmarking these models on a binary forest fire classification dataset.
c) Analyzing the trade-offs between detection performance and computational cost.

Thereby, offering more insights into the practicality of recurrent-enhanced deep learning models for real-time fire detection applications.

II. LITERATURE REVIEW

A. Conventional and Remote Sensing Fire Detection Methods

Traditional fire detection relied on point-based observation systems such as guard towers and Osborne Fire Finders [7], [8]. While these techniques provided localized monitoring, they were prone to human error and unreliable under harsh conditions in remote fire-prone areas [9]. To expand monitoring coverage, satellite-based methods such as MODIS in Canada and AVHRR in China were employed [10]. These systems exploited thermal and mid-infrared bands for detecting active fires and hot spots, offering broader observation capabilities than ground-based approaches [14]. However, their effectiveness was constrained by factors such as limited revisit frequency, cloud cover, terrain, and smoke from industrial sources, which often reduced their operational utility [7].

In addition to long-range satellite monitoring, remote sensing instruments mounted on fixed fire-watch towers have been used to detect either hot spots or smoke plumes in surrounding areas [14]. While continuous in principle, these systems also suffered from interference and lacked adaptability across diverse environmental conditions. Conventional sensor-based technologies were similarly constrained. Point-type smoke and temperature detectors, for instance, rely on photoelectric or thermistor-based designs but are ineffective in large, high-ceilinged spaces where smoke disperses slowly and temperature gradients diminish before reaching sensors [15], [16]. Line-type beam smoke detectors improved coverage in open areas but remained vulnerable to environmental dust, shielding by equipment, and false alarms from installation conditions [17], [18]. Infrared and ultraviolet flame detectors enhanced detection sensitivity by capturing characteristic flame emissions, yet they were easily disrupted by radiation from electronic equipment, dust accumulation, and environmental interference [19]–[22]. Aspirating smoke detectors, designed to actively sample air for smoldering fire detection, offered quicker responses but struggled to localize fire sources in large buildings, delaying suppression efforts [23]–[25]. Comparative evaluations show that while each detector type has unique strengths, they are hindered by maintenance demands, susceptibility to false alarms, and limited adaptability in complex environments [26].

Overall, although conventional and remote sensing fire detection technologies have provided valuable tools for monitoring fire-prone regions, their reliance on static thresholds, environmental sensitivity, and operational inefficiencies have restricted their broader applicability. These shortcomings underscore the necessity of exploring advanced approaches, particularly machine learning and deep learning methods, which can provide greater adaptability, robustness, and accuracy in real-time forest fire detection.

B. Machine Learning and Deep Learning in Fire Detection

Recent research highlights the growing role of AI-based vision systems in automating fire monitoring. UAV platforms equipped with CNNs not only provide mobility but also enable cost-efficient, real-time surveillance of fire-prone regions [11]. Deep learning has further enabled more accurate detection of smoke and small-scale fires through transfer learning and novel training strategies [12]. Fire-Net exemplifies the integration of advanced CNN structures with remote

sensing data, demonstrating robust performance across geographically diverse forests [13].

Several studies have compared traditional machine learning models with deep learning frameworks in fire detection. Image-based fire detection demonstrates that CNNs significantly outperform conventional algorithms, achieving accuracy rates of up to 99.3% [27]. Other comparative studies involving support vector machines, decision trees, logistic regression, and multilayer perceptrons show that the MLP model performs best, with an accuracy of 0.997 and stable training dynamics [28]. The introduction of the DeepFire dataset enables systematic benchmarking across models, where VGG19-based transfer learning demonstrates superior performance with 95% accuracy, 95.7% precision, and 94.2% recall [29]. An improved Detectron2-based model using a custom dataset of 5,200 images achieved a precision of 99.3% and demonstrated effective detection of small fires over long distances [30]. Similarly, an optimized YOLO model for flame detection surpassed shallow learning approaches, achieving 76% accuracy [31]. A hybrid pipeline combining CNN feature extraction with logistic regression yielded 94.1% accuracy for daytime and 94.8% for nighttime scenarios while also estimating flame-affected areas with fewer false positives [32].

While these studies demonstrate significant progress across CNN-based, hybrid, and object detection frameworks, they primarily emphasize static spatial recognition. Since the dataset in this study consists of independent frames and does not exhibit temporal dependencies as found in video-based data, the analysis is limited to spatial representations within each frame. However, the temporal evolution of fire signals remains an underexplored aspect, highlighting the potential of integrating recurrent architectures such as RNNs, LSTMs, and GRUs with CNNs to capture spatiotemporal dynamics and enhance detection reliability in future research.

C. CNN in Image Classification

The emergence of deep learning, particularly Convolutional Neural Networks (CNNs), marked a significant advancement in fire detection. CNNs have proven highly effective in spatial feature extraction and image classification across various domains such as smoke detection and environmental monitoring. UAV-based fire detection systems, for example, achieved an 83% recognition rate using a YOLOv3-enhanced CNN for real-time aerial imagery analysis [11]. Similarly, transfer learning approaches combined with Learning without Forgetting (LwF), LwF itself is an approach that enables model to learn new tasks while retaining performance on previously learned tasks [39], thereby allowed CNNs to adapt to new datasets while retaining prior classification capabilities, thus enhancing robustness in smoke detection tasks [12]. Moreover, multimodal frameworks such as Fire-Net, which fused optical and thermal imagery, achieved detection accuracy exceeding 97%, demonstrating the potential of CNNs in large-scale monitoring across diverse ecosystems [13].

Recent studies further emphasize CNNs as the backbone of many high-performing fire detection systems. Detectron2-based CNN models achieved precision rates 99.3% in distinguishing small-scale fires even under challenging conditions, benefiting from custom datasets with thousands of annotated images [30]. Transfer learning with CNNs has also been shown effective in forest fire detection, with VGG19-based models trained on the DeepFire dataset reaching 95% accuracy, precision, and recall [29]. Additionally, CNN feature extraction combined with Logistic Regression achieved over 94% accuracy in both daytime and nighttime scenarios while reducing false positives in estimating flame-affected areas [32].

Collectively, these results demonstrate the versatility and effectiveness of CNNs in handling various fire detection tasks, from UAV-based surveillance and video flame detection to multimodal fusion and hybrid classification pipelines. Their proven success in spatial image analysis establishes CNNs as a robust foundation for more advanced architectures that integrate temporal modeling layers. In relation to our own study, this body work provides a reason for adopting CNNs as the primary component for spatial feature extraction from individual data sets. Since our dataset consists of independent frames without temporal dependencies, CNNs serve as an appropriate baseline to capture discriminative spatial patterns. Building upon this foundation, our research further explores how recurrent-based models such as RNN, LSTM, and GRU can be incorporated to model temporal dependencies and enhance classification performance, while also examining the trade-offs between predictive accuracy and computational complexity across the developed architectures.

III. RESEARCH METHOD

A. Dataset description

The dataset employed in this study was obtained from a publicly available repository on Mendeley Data [33], accessible at FireDataset.

It consists of labeled fire and non-fire images collected from diverse scenarios, including forested regions, open areas, and mixed vegetation. The dataset captures variations under both daytime and nighttime conditions, with environmental challenges such as fog, clouds, and human presence that may contribute to false alarms.

In total, the dataset comprises 2000 images, with a balanced distribution across the two classes (fire and non-fire). Each image was standardized to a fixed input resolution of 768 × 768 pixels, and pixel intensities

were normalized to the range [0, 1] for training stability.

For model evaluation, we adopted a 5-fold stratified cross-validation scheme, ensuring that each fold preserved the ratio of 80% Train and 20% Test samples. This approach reduces sampling bias and yields a more robust estimation of classification performance.

B. Model architecture

1) CNN-MLP baseline

The baseline model, illustrated in Fig. 1, follows a conventional CNN-MLP pipeline. The CNN component is responsible for spatial feature extraction, where convolutional layers capture local patterns such as edges, textures, and color variations by leveraging filters for corresponding RGB channels, followed by activation and pooling layers to reduce dimensionality while preserving key spatial information. The resulting feature maps are then flattened and passed into a fully connected multilayer perceptron (MLP), which performs binary classification into the two target classes: *fire* and *non-fire*. This architecture focuses solely on learning spatial representations, without accounting for the temporal dynamics present in sequential fire imagery. The absence of recurrent layers in this baseline design is intentional, as the model aims to isolate the performance of spatial feature extraction alone before incorporating temporal modeling in subsequent recurrent-based architectures.

2) CNN-recurrent model

To capture both spatial and temporal features, the CNN-MLP baseline is extended with recurrent layers, as shown in Fig. 2. The CNN block extracts spatial features, which are then passed to recurrent layers before the fully connected classifier. Depending on the configuration, this recurrent layer may consist of an RNN, an LSTM, or a GRU, each of which processes the sequential patterns differently:

- **CNN-RNN:** Incorporates a standard recurrent neural network layer. RNNs process input sequences step by step, passing hidden states across time. While effective for short-term dependencies, they suffer from vanishing gradients, which reduces their ability to model long-term temporal dependencies [35].
- **CNN-LSTM:** Replaces the RNN with long short-term memory (LSTM) units. LSTMs mitigate vanishing gradient issues by using memory cells and gating mechanisms (input, forget, and output gates). This design enables them to capture both short- and long-term temporal dependencies, making them highly effective for complex sequential tasks [36].
- **CNN-GRU:** Uses gated recurrent units (GRUs), which simplify the LSTM architecture by combining the forget and input gates into a single update gate. GRUs achieve comparable performance to LSTMs with fewer parameters, offering a computationally efficient alternative for modeling sequential dependencies [37].

These recurrent-enhanced models jointly leverage CNN-based spatial learning and recurrent-based temporal sequence modeling, thereby improving the system's ability to detect and classify fires in challenging scenarios.

C. Experimental setup

The proposed models were trained and evaluated under a consistent experimental setup to ensure fair comparison and reliable performance estimation across all architectures. The main configurations are summarized as follows:

- **Loss function:** Binary cross-entropy (BCE) loss was employed, as the task is a binary classification problem with two target classes: *fire* and *non-fire*. BCE effectively penalizes incorrect predictions by measuring the difference between predicted probabilities and ground-truth labels, making it a standard choice for binary classification tasks.
- **Optimizer:** The Adam optimizer was utilized to update network weights. Adam combines the advantages of adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp), offering efficient training with adaptive learning rates and momentum. This makes it suitable for complex models such as CNNs and recurrent networks.
- **Learning rate:** The initial learning rate was set to 1×10^{-3}, which balances convergence speed and training stability. A smaller learning rate would slow convergence, while a larger one could lead to instability or divergence.
- **Validation strategy:** To mitigate overfitting and ensure robust performance evaluation, a 5-fold cross-validation strategy was applied. The dataset was partitioned into five folds, with four folds used for training and one fold for validation in each iteration. Final performance metrics were averaged across all folds.
- **Training epochs:** The number of epochs was chosen empirically based on model convergence. The CNN-RNN model was trained for 80 epochs, while both CNN-LSTM and CNN-GRU models were trained for 50 epochs. This adjustment reflects differences in convergence behavior, as recurrent architectures with gating mechanisms (LSTM, GRU) typically converge faster and are less prone to vanishing gradients than vanilla RNNs.
- **Batch size:** A batch size of 512 was used across all experiments. This choice balances computational efficiency and gradient stability. Larger batch sizes improve parallelization on GPU hard-

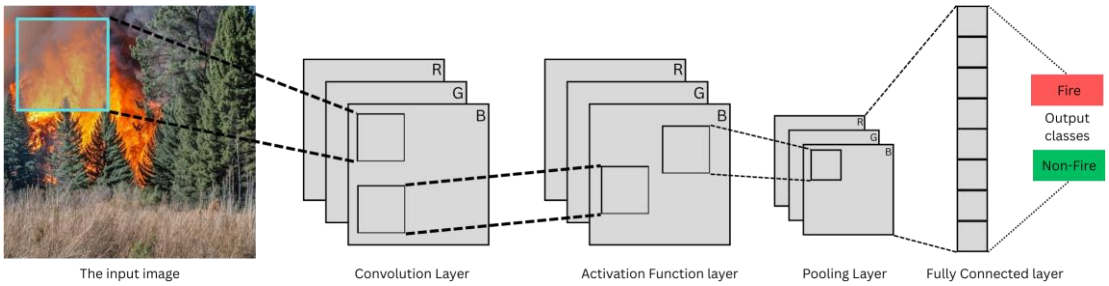

Fig. 1. CNN-MLP model architecture

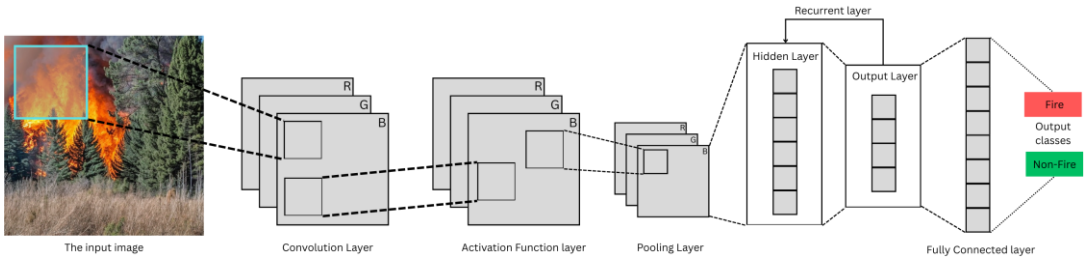

Fig. 2. CNN-recurrent model architecture

ware but may reduce generalization, whereas smaller batch sizes often lead to noisier updates.

This setup ensures consistent comparison between architectures while also leveraging optimization strategies commonly used in deep learning for image classification and sequence modeling tasks.

D. Evaluation metrics

To comprehensively assess the performance of the proposed models, we employ multiple evaluation metrics that capture classification quality, discriminative ability, regression-style explanatory power, and computational complexity.

1) Classification metrics

The confusion matrix is constructed using true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). From this, several performance measures are derived:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Accuracy provides an overall correctness measure, while precision highlights the proportion of correctly predicted fire instances among all predicted fires. Recall emphasizes sensitivity, i.e., the ability to detect actual fire occurrences, and the F1-score balances precision and recall.

2) AUC score

The area under the ROC curve (AUC) evaluates the trade-off between true positive rate (TPR) and false positive rate (FPR) across different thresholds:

$$\text{TPR} = \frac{\text{TP}}{\text{P}} \quad (5)$$

$$\text{FPR} = \frac{\text{FP}}{\text{N}} \quad (6)$$

$$\text{AUC} = \sum_{k=1}^{n-1} \frac{(\text{FPR}_{k+1} - \text{FPR}_k) \cdot (\text{TPR}_k + \text{TPR}_{k+1})}{2} \quad (7)$$

Here, P represents the total number of positive instances, while N denotes the total number of negative instances. According the AUC, k and $k + 1$ correspond to adjacent points on ROC curve, which are derived from consequtive classification thresholds. Higher AUC values indicate stronger discriminative capability of the model in distinguishing between fire and non-fire classes.

3) R-squared score

In addition to classification-based metrics, we also report the coefficient of determination (R^2), which measures how well the predicted outputs approximate the ground-truth values:

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{m} (Y_i - \bar{Y})^2} \quad (8)$$

Suppose y_i is the actual label e.g 0 or 1, $\hat{y}_i$ is probability predicted by the model for positive instances, and $\bar{y}$ is the mean of all actual labels. Although more common in regression, the R^2 score provides additional insight into the variance explained by the model predictions.

4) Model complexity

To complement predictive performance, we analyze the computational cost of each architecture. Model complexity is estimated using the theoretical

number of floating-point operations (FLOPs), multiply–accumulate operations (MACs), and trainable parameters across all neural network layers. For this purpose, we employ the open-source *Calflops* tool [38], which provides a reliable theoretical model complexity estimation for PyTorch-constructed models.

IV. RESULTS

Our extensive experiment on developing the four determined models yields a model architecture illustrated in Fig. 3. All models used the identical CNN block, which consists of a two-dimensional convolutional layer followed by ReLU as well as max pooling layer. We bifurcated the advance stages into two different computations exclusively to manage specific models input size.

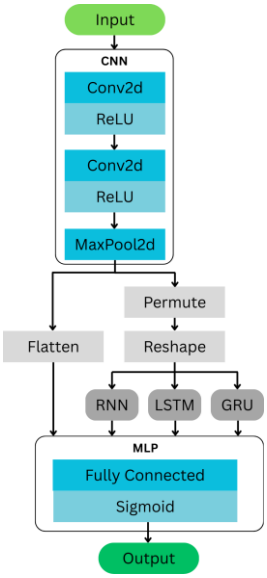

Fig. 3. Models Architectures

The first is merely one operation, flattening the extracted features from $R^{N\times C\times H\times W}$ to $R^{N\times F}$. The learned feature maps, height, and width relative to the image, which are the last three dimensions of the extracted features, are multiplied to obtain F, denoting flattened features. The N is preserved since it represents the batch size of the data. Subsequently, the flattened features are fed into the MLP for binary classification training.

The second is special to the recurrent-based models, dedicated to arranging extracted features in a sequential timestep fashion. The permute operation changed the order of extracted feature sizes from $R^{N\times C\times H\times W}$ to $R^{N\times H\times C\times W}$. The following operation is similar to the flatten operation, yet only multiplies the last two dimensions to produce $R^{N\times H\times T}$ feature size, acquisitioning T, which depicts all channel features. Those operations aim to exploit each row of pixels as a sequence, in which an individual row is a step. This approach enables the model to learn horizontal spatial continuity, leading to computationally more compomise compared to full 2D recurrent models that simultaneously process both row and column dependencies. In addition, the horizontally captured dependencies allows our model to discern deviations from a typical vertical forest composition, particularly in the presence of disruptive elements such as flame and smoke. Consequently, all transformed features are fed into the recurrent-based models for capturing feature patterns across rows. Finally, we passed MLP the last timestep of the recurrent layers, which contains a summary of the whole sequence.

The performance of all models in a stratified 5-fold cross validation setting is revealed by taking the average performance of the 5 folds. As shown in Table 1, the CNN-MLP model achieved the highest performance, and the CNN-RNN model achieved the lowest. CNN-LSTM and CNN-GRU, which contain more advanced recurrent layers, outperformed CNN-RNN, which contains ordinary recurrent layers. In addition, the R^2 of the CNN-RNN remained significantly different from other models, which means CNN-RNN is worse than the others when attempting to explain the proportion of variance in the target variable.

Table 2 captures the individual complexity of the models in Giga MACs (GMACs) and Giga Flops (GFLOPs) units of measure when done forward (fwd) and forward-backward (fwd+bwd) propagation for one step on batch size of 256, describing computational resource requirements during inferencing and training. CNN-MLP is the most complex model, with just over 1 million parameters and more than triple the computational cost of the recurrent-based models. CNN-RNN is the least complex model, while CNN-LSTM and CNN-GRU have three times and twice as many parameters. Nevertheless, all recurrent-based models had similar MACs throughout the forward and forward-backward passes.

V. DISCUSSIONS

Through rigorous experimentation and iterative refinement, we uncovered several critical insights that significantly contributed to the successful design of the proposed architecture. At the beginning of the experiment, we applied the first CNN block of the Tiny VGG architecture as well as the same parameters in [34]. Subsequently, we managed to assemble it with the four determined models. As the starting point we proceeded to train the CNN-MLP models. We found that as the hidden unit increased the performance got lower. Thus, we tried to low the hidden unit size up to 2 and adjusted the parameters of the CNN block. Finally, we revealed the best fit hidden size as 8 followed by the tuned CNN block parameters, which have kernel size = 3, stride = 1, padding = 0, and dilation = 1 for the convolution layers and kernel size = 3 as well as stride = 2 for the max pooling layer. Those parameters led the

Table 1. Models performance

Models	Acc	Prec	Rec	F1	AUC	R^2
CNN-MLP	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.77**
CNN-RNN	0.81	0.85	0.81	0.81	0.81	0.25
CNN-LSTM	0.91	0.92	0.91	0.91	0.91	0.65
CNN-GRU	0.93	0.93	0.93	0.93	0.93	0.70

Table 2. Models Complexity

Models	GMACs		GFLOPs		Parameters
	fwd	fwd + bwd	fwd	fwd + bwd	
CNN-MLP	118.81	356.44	243.61	730.84	1.16 M
CNN-RNN	**37.74**	**113.22**	**78.33**	**234.98**	**27.39 K**
CNN-LSTM	**37.74**	**113.22**	81.92	245.75	101.22 K
CNN-GRU	**37.74**	**113.22**	80.72	242.16	76.61 K

CNN-MLP to achieve such performance outperformed the recurrent-based models.

The same obstacle was also experienced by the recurrent-based models but easier for CNN-LSTM and CNN-GRU since we had obtained the baseline architectures from the CNN-RNN. We followed the same strategy for training the CNN-RNN, where we trained it with such a high hidden unit and the preserved CNN block parameters. Unfortunately, the setting led the models to underperform. To the end of this, we played along by preserving the CNN block parameters but tuned the hidden unit. Since the result was also unsatisfactory, we not only adjusted the CNN block parameters but also the hidden size as well. Ultimately, we revealed kernel size = 3, stride = 2, padding = 0, and dilation = 1 attributed the convolution layers and kernel size = 3 and stride = 2. In addition, the best fit hidden size was 16 for a RNN layer. The CNN-LSTM and CNN-GRU also attained their best performance in the same setting. We attempted to improve it by permuting the combination of the parameters and adding batch normalization layers after convolution layer, yet yielded decreased performance.

According to the revealed best fit parameters for individual models, it make sense for the CNN-MLP to require such theoretically expensive computational cost compared to the recurrent-based models. The CNN-MLP required a stride of 1 instead of 2 as in the recurrent-based models. The low number of strides did not allow the extracted features to be extremely reduced against the height and width of the original size. Therefore, the extracted features were reduced more in the CNN-RNN, CNN-LSTM, and CNN-GRU compared to the CNN-MLP.

Table 3. CNN-MLP Forward Pass Complexity Percentage

CNN-MLP			
Layers	**MACs**	**FLOPs**	**Parameters**
Conv2d	27.31	27.13	0.02
ReLU	0.00	0.49	0.00
Conv2d	72.44	71.15	0.05
ReLU	0.00	0.49	0.00
MaxPool2d	0.00	0.49	0.00
Linear	0.25	0.24	99.93
Sigmoid	0.00	0.00	0.00
Total	100%	100%	100%

Table 4. CNN-RNN Forward Pass Complexity Percentage

CNN-RNN			
Layers	**MACs**	**FLOPs**	**Parameters**
Conv2d	42.99	42.19	1.64
ReLU	0.00	0.77	0.00
Conv2d	57.01	55.13	8.47
ReLU	0.00	0.19	0.00
MaxPool2d	0.00	0.19	0.00
RNN	0.00	1.53	89.83
Linear	≈ 0	≈ 0	0.06
Sigmoid	0.00	0.00	0.00
Total	100%	100%	100%

Table 5. CNN-LSTM Forward Pass Complexity Percentage

CNN-LSTM			
Layers	**MACs**	**FLOPs**	**Parameters**
Conv2d	42.99	40.34	0.44
ReLU	0.00	0.73	0.00
Conv2d	57.01	52.72	2.29
ReLU	0.00	0.18	0.00
MaxPool2d	0.00	0.18	0.00
LSTM	0.00	5.84	97.25
Linear	≈ 0.00	≈ 0.00	0.02
Sigmoid	0.00	0.00	0.00
Total	100%	100%	100%

Table 6. CNN-GRU Forward Pass Complexity Percentage

CNN-GRU			
Layers	**MACs**	**FLOPs**	**Parameters**
Conv2d	42.99	40.94	0.58
ReLU	0.00	0.74	0.00
Conv2d	57.01	53.50	3.03
ReLU	0.00	0.19	0.00
MaxPool2d	0.00	0.19	0.00
GRU	0.00	4.45	96.36
Linear	≈ 0.00	≈ 0.00	0.02
Sigmoid	0.00	0.00	0.00
Total	100%	100%	100%

We systematically displayed the percentages of the individual constructing layers for every model as shown in Table 3, 4, 5, and 6. The percentages displayed in those tables originated from the forward pass phase, which represents the requirement of the model during classification. We did this to gain deeper insight about how each layer contributed to the complexity once the model was assigned. As shown in Table 3, CNN-MLP, had the resource-intensive layer at the second convolution layer by just over 71% for its MACs and FLOPs and possessed the largest parameters laid in the linear layer by 99.93%. Conversely, the recurrent-based models did not show the significant macs and flops gap between the first and the second

convolutional layer compared to the CNN-MLP, which means the recurrent-based models had more evenly distributed computation. In addition, their parameter number was centered around its recurrent layer, contributing more than 89 percent of all parameters for individual models.

We evaluated the storage size of the saved models and observed notable variations across architectures. Specifically, the CNN-GRU model required 306 KB, the CNN-LSTM model 403 KB, and the CNN-RNN model 114 KB. In contrast, the CNN-MLP exhibited a substantially larger size of 4.545 KB. This outcome aligns with expectations, as the MLP lacks the parameter-sharing mechanisms inherent to recurrent models, resulting in a significantly larger model size. The pronounced discrepancy underscores the structural differences between recurrent-based architectures and fully connected networks, particularly regarding parameter efficiency.

VI. CONCLUSIONS

We conclude our research by stating that for fire and non-fire forest classification task, CNN-MLP model is computationally heavier to achieve tremendous compared to the CNN-GRU, which approximately chases the performance of CNN-MLP by only one less difference at confusion matrix. Also, CNN-LSTM is a viable option for the implementation of the classification task since the performance remains stable. Meanwhile, the CNN-RNN requires more enhancement so it can better explain the variability of the target during predicting. We recommend further research to adjust the threshold of the binary classification since in this research we set it to be 0.5, which means if the probability is less than 0.5 it is considered non-fired and if it is greater or equal to 0.5 it is considered as fired. The goals of the trials may reveals underexplored insight to the real world application, although false alarm may be higher, but safer. Another recommendation is to explore state-of-the-art of transformer models, which possess attention behaviour combined with superpixel algorithms.

REFERENCES

[1] V. Ferna´ndez-Garc´ıa, E. Marcos, J. M. Ferna´ndez-Guisuraga, A. Taboada, S. Suarez-Seoane, and L. Calvo, "Impact of burn severity on soil properties in a *Pinus pinaster* ecosystem immediately after fire," Int. J. Wildland Fire, v

[2] A. Bento-Gonc¸alves, A. Vieira, X. U´ beda, and D. Martin, "Fire and soils: key concepts and recent advances," Geoderma, vol. 191, pp. 3–13, Jan. 2012.

[3] A. A. Agbeshie, S. Abugre, T. Atta-Darkwa, et al., "A review of the effects of forest fire on soil properties," J. For. Res., vol. 33, pp. 1419–1441, 2022. [Online]. Available: https://doi.org/10.1007/s11676-022-01475-4

[4] M. D. Flannigan, B. J. Stocks, and B. M. Wotton, "Climate change and forest fires," Sci. Total Environ., vol. 262, no. 3, pp. 221–229, 2000. [Online]. Available: https://doi.org/10.1016/S0048-9697(00)00524-6

[5] A. Mohapatra and T. Trinh, "Early wildfire detection technologies in practice—a review," Sustainability, vol. 14, no. 19, p. 12270, 2022. [Online]. Available: https://doi.org/10.3390/su141912270

[6] V. Cuomo, R. Lasaponara, and V. Tramutoli, "Evaluation of a new satellite-based method for forest fire detection," Int. J. Remote Sens., vol. 22, no. 9, pp. 1799–1826, 2001. [Online]. Available: https://doi.org/10.1080/01431160120827

[7] M. Hefeeda and M. Bagheri, "Forest fire modeling and early detection using wireless sensor networks," Ad Hoc Sensor Wireless Netw., vol. 7, no. 3–4, pp. 169–224, 2009.

[8] M. Bahrepour, N. Meratnia, and P. Havinga, "Automatic fire detection: A survey from wireless sensor network perspective," Centre Telematics Inf. Technol., Univ. Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-08-73, 2008. ISSN 1381-3625.

[9] Y. E. Asian, I. Korpeoglu, and O. Ulusoy, "A framework for use of wireless sensor networks in forest fire detection and monitoring," Comput., Environ. Urban Syst., vol. 36, no. 6, pp. 614–625, 2012. [Online]. Available: https://doi.org/10.1016/j.compenvurbsys.2012.03.003

[10] K. Bouabdellah, H. Noureddine, and S. Larbi, "Using wireless sensor networks for reliable forest fires detection," Procedia Comput. Sci., vol. 19, pp. 794–801, 2013. [Online]. Available: https://doi.org/10.1016/j.procs.2013.06.110

[11] Z. Jiao, et al., "A deep learning based forest fire detection approach using UAV and YOLOv3," in Proc. 1st Int. Conf. Ind. Artif. Intell. (IAI), Shenyang, China, 2019, pp. 1–5, doi: 10.1109/ICIAI.2019.8850815.

[12] V. E. Sathishkumar, J. Cho, M. Subramanian, et al., "Forest fire and smoke detection using deep learning-based learning without forgetting," Fire Ecol., vol. 19, p. 9, 2023. [Online]. Available: https://doi.org/10.1186/s42408-022-00165-0

[13] S. T. Seydi, V. Saeidi, B. Kalantar, N. Ueda, and A. A. Halin, "Fire-Net: A deep learning framework for active forest fire detection," J. Sensors, vol. 2022, no. 1, p. 8044390, 2022. [Online]. Available: https://doi.org/10.1155/2022/8044390

[14] J. San-Miguel-Ayanz, J. M. Moreno, and A. Camia, "Analysis of large fires in European Mediterranean landscapes: lessons learned and perspectives," For. Ecol. Manage., vol. 294, no. 1–3, pp. 11–22, 2005. [Online]. Available: https://doi.org/10.1016/j.foreco.2004.10.050

[15] Y. Cui, H. Zhang, Y. Wang, and X. Li, "An intelligent smoke detection system for forest fire prevention using multi-sensor data fusion," Sensors, vol. 24, no. 2, p. 567, 2024. [Online]. Available: https://doi.org/10.3390/s24020567

[16] A. Gaur, B. Singh, and P. Kumar, "Wireless sensor network: An approach for early forest fire detection," Int. J. Adv. Res. Comput. Sci., vol. 10, no. 5, pp. 57–61, 2019. [Online]. Available: https://doi.org/10.26483/ijarcs.v10i5.6473

[17] L. Yao, Z. Wang, and X. Chen, "A review of smoke detection technologies in wildland fire monitoring," Fire, vol. 6, no. 1, p. 25, 2023. [Online]. Available: https://doi.org/10.3390/fire6010025

[18] B. J. Meacham, "Fire detection technologies and applications: A review," Fire Technol., vol. 29, no. 1, pp. 5–29, 1993. [Online]. Available: https://doi.org/10.1007/BF01040264

[19] Y. Long, X. Wei, and H. Wang, "Flame detection based on image processing and support vector machines," Fire Saf. J., vol. 62, pp. 186–195, 2013. [Online]. Available: https://doi.org/10.1016/j.firesaf.2013.09.009

[20] C. C. Tsai and C. H. Huang, "A novel flame detection technique for video-based surveillance systems," Fire Saf. J., vol. 41, no. 6, pp. 512–522, 2006. [Online]. Available: https://doi.org/10.1016/j.firesaf.2006.05.001

[21] S. Bordbar, M. Ghasemi, and A. H. Sadeghi, "Outdoor flame detection in forest environments using infrared and ultraviolet sensors," Sensors, vol. 22, no. 14, p. 5280, 2022. [Online]. Available: https://doi.org/10.3390/s22145280

[22] Y. Pauchard, J. C. Borel, and J. L. Mermier, "Fire detection in natural environments using UV sensors," Fire Saf. J., vol. 34, no. 4, pp. 307–318, 2000. [Online]. Available: https://doi.org/10.1016/S0379-7112(99)00048-2

[23] X. Li, J. Zhao, and K. Wang, "Performance of aspirating smoke detectors in different environmental conditions," Fire Saf. J., vol. 120, p. 103087, 2021. [Online]. Available: https://doi.org/10.1016/j.firesaf.2021.103087

[24] H. Lee, D. Kim, and J. Park, "Application of aspirating smoke detection system in large open spaces," J. Fire Prot. Eng., vol. 32, no. 4, pp. 289–302, 2022. [Online]. Available: https://doi.org/10.1177/10423915221083742

[25] H. Visak and R. Sharma, "Review on aspirating smoke detection systems for fire safety," Int. J. Eng. Adv. Technol., vol. 10, no. 6, pp. 142–147, 2021. [Online]. Available: https://doi.org/10.35940/ijeat.F3250.089621

[26] X. Deng, Y. Sun, and M. Zhou, "Challenges and perspectives of fire detection technologies in wildland environments: A comprehensive review," Fire Technol., vol. 61, pp. 1123–1156, 2025. [Online]. Available: https://doi.org/10.1007/s10694-025-01321-9

[27] S. B. Kukuk and Z. H. Kilimci, "Comprehensive analysis of forest fire detection using deep learning models and conventional machine learning algorithms," Int. J. Comput. Exp. Sci. Eng., vol. 7, no. 2, pp. 84–94, 2021. [Online]. Available: https://doi.org/10.22399/ijcesen.950045

[28] A. Secilmis, N. Aksu, F. A. Dael, I. Shayea, and A. A. El-Saleh, "Machine learning-based fire detection: A comprehensive review and evaluation of classification models," JOIV: Int. J. Informatics Vis., vol. 7, no. 3–2, pp. 1982–1988, 2023.

[29] A. Khan, B. Hassan, S. Khan, R. Ahmed, and A. Abuassba, "DeepFire: A novel dataset and deep transfer learning benchmark for forest fire detection," Mobile Inf. Syst., vol. 2022, no. 1, p. 5358359, 2022. [Online]. Available: https://doi.org/10.1155/2022/5358359

[30] A. B. Abdusalomov, B. M. S. Islam, R. Nasimov, M. Mukhiddinov, and T. K. Whangbo, "An improved forest fire detection method based on the detectron2 model and a deep learning approach," Sensors, vol. 23, no. 3, p. 1512, 2023. [Online]. Available: https://doi.org/10.3390/s23031512

[31] D. Shen, X. Chen, M. Nguyen, and W. Q. Yan, "Flame detection using deep learning," in *Proc. 4th Int. Conf. Control, Autom. Robot. (ICCAR)*, pp. 416–420, 2018. [Online]. Available: https://doi.org/10.1109/ICCAR.2018.8384711

[32] J. Alves, C. Soares, J. M. Torres, P. Sobral, and R. S. Moreira, "Automatic forest fire detection based on a machine learning and image analysis pipeline," in A´. Rocha, H. Adeli, L. Reis, and S. Costanzo (eds.), *New Knowledge in Information Systems and Technologies (WorldCIST'19)*, pp. 259–269. Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-16184-2_24

[33] A. Rafi, S. A. Tanim, T. Biswas, F. Mridha, M. Safran, S. Alfarhood, and D. Che, "Fire and non-fire forest dataset," Mendeley Data, V2, 2025. [Online]. Available: https://doi.org/10.17632/5vb2kds993.2

[34] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Polo Chau, "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1396–1406, Feb. 2021. [Online]. Available: https://doi.org/10.1109/TVCG.2020.3030418

[35] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990. [Online]. Available: https://doi.org/10.1016/0364-0213(90)90002-E

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] K. Cho, B. Van Merrie¨nboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[38] X. Ye, "calflops: a FLOPs and Params calculate tool for neural networks in PyTorch framework," GitHub, 2023. [Online]. Available: https://github.com/MrYxJ/calculate-flops.pytorch

[39] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.