AI-Powered Fire Surveillance for Greece: Real-Time Wildfire and Smoke Detection using Neural Networks

Vasilis Tsavalias(ics21083)
Ioannis Kontaxis(ics21105)

An assignment for the course "Neural Networks"
8th Semester
Instructor: Prof. Protopapadakis

University of Macedonia

30–06–2024

Abstract

Wildfires pose a significant threat that keeps deteriorating due to human activity. They are among the most devastating events caused by the climate crisis, impacting ecosystems, populations, and infrastructure on a global scale. Swift action is essential to minimizing the catastrophic consequences of this phenomenon. This study introduces four different architectures for convolutional neural networks (CNNs) to improve the ability to detect and classify wildfires. The study specifically focuses on two specific tasks: classifying fires and detecting smoke.

The proposed system proposes three pretrained CNN models (ResNet50,Inception V3,Xception) on the ImageNet dataset, used to classify images as either fire or non-fire, as well as a YOLOv8 model for detecting smoke in real-time. After training on a varied dataset of both aerial and ground-based photos, the pre-trained CNN model achieved an F1 score of 74% on the test set. The YOLOv8 model demonstrated optimal smoke detection capabilities, with a recall rate of 87% on the test set. Advanced techniques for optimizing performance were employed, such as data augmentation, hyperparameter tweaking and model ensembling. They were used to maximize the models' effectiveness and obtain a deeper understanding of their strengths and limitations. The study also emphasized the difficulties caused by limitations in computational resources and addressed approaches to reduce their impact on model performance.

The results of this study contribute to future research by employing advanced ways of facing the proposed challenge. Additionally, they will contribute to the advancement of wildfire detection techniques and shed more insight on how to facilitate early intervention and mitigate their deadly outcomes.

Keywords: Convolutional Neural Networks (CNNs), YOLOv8 architecture, wildfire classification, smoke detection, deep learning, object detection, computer vision.

Contents

Introduction

1.1 Motivation

The impacts of climate change are especially noticeable in Greece. During the dry summers of the Mediterranean climate, regular and severe wildfires have become a staple. For example, the lethal "Mati wildfire" in 2018 killed over 100 people [1]. Similarly, the most destructive wildfire in recent history on Euboia in 2021 destroyed over 125,000 hectares of agricultural land, causing significant degradation in soil properties and a reduction in organic matter and nutrient levels [2]. The urgency of addressing this man-made crisis is increasing by the year.

Data reveal that wildfires are becoming more common and severe. Greek wildfires have increased in frequency and area over the last ten years. According to data extracted from the European Forest Fire Information System (EFFIS) [3], more than 110,000 hectares burned in Greece in 2021, which is more than five times the yearly average from 2008 to 2020. New data published in June 2024 shows that this month alone, 300 fires have already been escalated, with a new fire occurring every 10 minutes [4]. In 2023, wildfires destroyed approximately 222,577 hectares of land (nearly 2% of Greece's total land area) within a single year, causing CO2 emissions equivalent to over 222,000 cars in a year [5].

To combat this, mitigation and firefighting must depend on quick and precise wildfire detection. Early discovery enables a quick response, thereby preventing more extensive damage and loss. Traditional detection techniques such as manual monitoring, sensor networks, and satellite images are often limited and inaccurate [6]. Computer vision may provide more advanced ways of detecting wildfires. Object detection systems and convolutional neural networks (CNNs) can rapidly and precisely identify smoke and fire in visual data. These approaches offer efficient methods of wildfire detection, with studies demonstrating significant improvements in early detection capabilities, reduced detection times, and improved accuracy [7, 8].

Scope

The proposed study attempts to mitigate this phenomenon by enhancing the speed and accuracy of wildfire detection through the use of customized CNN architectures. The system employs the YOLO (You Only Look Once) architecture, renowned for its rapid inference speed and high accuracy, to swiftly identify smoke. Similarly, the CNN model classifies fire characteristics and assesses the magnitude of wildfires—or the absence of them [9] based on three different architectures based on the ImageNet dataset. The objective of the proposed system is to greatly improve the speed at which early detection and response occur by combining IoT devices and advanced neural networks (SAI).

1.2 Structure of the Dissertation

The first chapter, **Introduction**, briefly covered the global wildfire situation and Greece's struggles. It emphasized early and accurate wildfire detection to mitigate a possible wildfire crisis.

Modern wildfire detection will be covered in **Literature Review**, the second part. Deep learning methods for object classification and object detection will be addressed. Relevant research will be assessed, along with techniques for wildfire detection data shortages, imbalanced classes, and processing restrictions.

In chapter 3, **"Methodology"**, the challenge of boosting wildfire detection in Greece utilizing CNN architectures will be defined, providing a high-level overview of the "Expiremental" Setup" chapter.

In the fourth chapter, **Experimental Setup** image classification and object detection datasets will be described. Data sources, image resolution, model architecture, and model evaluation alike. Both models will be evaluated using accuracy, precision, recall, F1-score, and mean average precision. Training and evaluation of hardware and software configuration will also be described. This incorporates Google Colab's and Kaggle's computational limits. Finally, the chapter will tackle advanced performance analysis methods such as detection outcomes, model interpretability, hyperparameter optimization, and model ensembling.

Results and Discussion concludes with the YOLOv8 model's and pre-trained CNN's smoke and fire detection results. This chapter will evaluate the model's fire characteristics and discuss noteworthy findings, computation complexity, prediction conflicts, and system constraints.

The final section, **Conclusion**, will summarize the research's key findings and contributions. Investigations will be done to expand and improve the proposed system. Testing model architecture, adding data sources, or overcoming study restrictions may be needed.

The document's **References** and **Appendices** sections contain citations, figures, and tables.

Literature Review

2.1 Current Technologies in Wildfire Detection

In 2023, the Greek Ministry of Research and Innovation implemented a comparable strategy. The Greek Ministry of Research and Innovation was responsible for the initial implementation of "Heat Alarm" software. This newly-created system utilizes satellite pictures and sensor networks to promptly notify people of potential wildfires. The method predicts the occurrence of fires and identifies irregularities in vegetation indicators by examining weather forecasts. [10]

On a global scale, the UN's "AINA" system employs satellite photography, ground sensor data, and historical patterns [11] to globally utilize machine learning algorithms for the purpose of identifying natural disasters, such as wildfires.

Furthermore, the National Oceanic and Atmospheric Administration (NOAA) use deep learning models in their artificial intelligence (AI)-powered wildfire early warning system to analyze satellite data and precisely identify areas of intense heat. [12].

2.2 Deep Learning Applications in Environmental Monitoring

Deep learning is aiding in the efforts of environmental monitoring. The way forests are changing before and during a wildfire has been accurately covered by much research on the utilization of Convolutional Neural Networks (CNNs). Additionally, these CNNs have proven effective in identifying satellite image patterns for the purpose of monitoring deforestation. Lastly, they are able to quickly and efficiently analyze ground cameras and satellite photos to detect the presence of smoke. By utilizing the hierarchical feature representations of CNNs, these models are able to effectively distinguish complex smoke plume patterns [13].

Deep learning is utilized to predict the probability and extent of wildfires by analyzing past data and environmental factors. Predictive models can aid in the suppression of wildfires by allocating resources and coordinating evacuations. [9].

2.3 CNNs for Image Classification(ImageNet) and Object Detection (YOLO)

CNNs, or Convolutional Neural Networks, are deep learning models specifically designed for image processing. Layered CNNs utilize arriving pictures to construct hierarchical feature representations. Convolutional approaches extract edges, textures, and patterns at each layer. The basic architecture of CNN is shown in Figure A.13 below:

The real-time performance and precision of modern object detection systems, such as the YOLO (You Only Look Once) algorithm, particularly its 8th iteration pre-trained model,

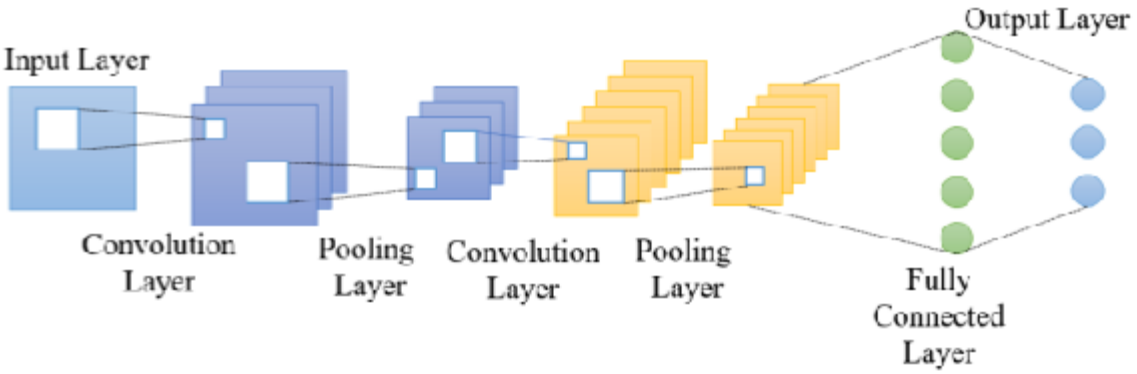

Figure 2.1: This image illustrates a simple CNN structure, which would typically include layers such as input, convolutional, pooling, and fully connected layers, culminating in an output layer. Each layer has its own specific role, such as feature extraction and non-linear transformations, which are essential for tasks like image classification. [14].

have significantly helped in combating wildfires. The proposed model predicts the coordinates of bounding boxes and the probabilities of different classes in an input image using a single neural network, without the need for region proposal techniques [15].

Each YOLO cell within the grid predicts the location and size of bounding boxes, as well as the probabilities of different object classes in the surrounding area. End-to-end training optimizes both localization and classification tasks, resulting in the creation of powerful feature representations for object detection [16]. The basic architecture of YOLO is shown in Figure A.25 below:

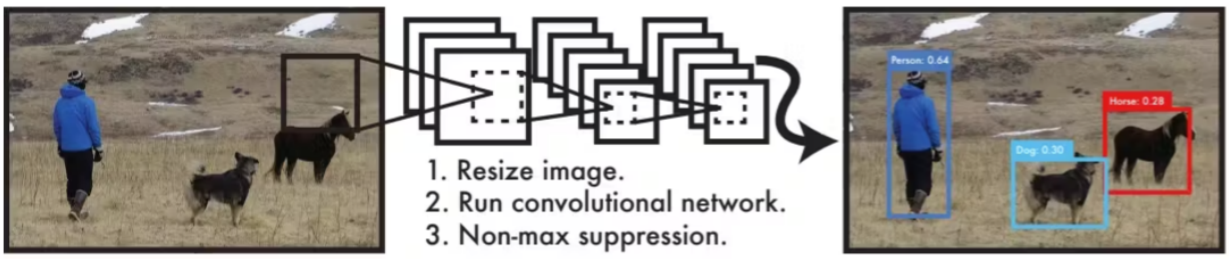

Figure 2.2: This diagram illustrates a basic YOLO architecture, which is used for real-time object detection. The network processes the whole image during inference, predicts multiple bounding boxes and class probabilities for these boxes simultaneously, making it extremely fast and suitable for real-time applications. [17]

Using large datasets of annotated photos, these models have the ability to detect and classify things associated to wildfires in real-time. [8, 18].

Methodology

3.1 Challenge Analysis

A resilient system is needed to meet wildfire detection's successful accuracy outcomes. The models must be trained with varied data, handle real-time model demands and most importantly, they have to be reliable and fully operational constantly. The system must efficiently handle data streams, deliver warnings quickly, and balance accuracy and inference speed. Lastly, firefighters and emergency responders must be able to trust and comprehend the system's forecasts.

3.2 Datasets Preparation

Two datasets, taken from Kaggle, were used to handle wildfire and smoke detection challenges: "dataset_image_classification" for classifying fires and "dataset_object_detection" for real-time smoke detection.

The dataset "dataset_image_classification" includes 2,700 aerial and ground-based photos from various websites. The dataset was already divided into training (1,887 photographs), validation (402 images), and testing (410 images). The model was built and evaluated using these subsets. [19]

The "dataset_object_detection" provides photos with annotated bounding boxes for smoke and fire objects, aiding in object detection model training and evaluation. The dataset includes 516 training photos, 147 validation photos, and 74 test photos. [20]

3.3 Model Development

3.3.1 Training and Validation

The study employed three pre-trained CNN architectures based on the ImageNet dataset. Ensembled Resnet50,Inception and Xception models were utilized for fire classification and a YOLOv8 model for real-time smoke and fire detection was used.

The pre-trained models were modified for binary fire classification with custom layers. The model was compiled using accuracy, binary cross-entropy loss, and Adam optimizer. While training, the models achieved flawless classification on the training set, with a validation accuracy of 0.9030 after 12 epochs.

Smoke detection was done in real time utilizing the "Ultralytics"-built YOLOv8 model. It was initialized with pre-trained weights and adjusted with hyperparameters for the current purpose. The identified smoke and fire items after 26 iterations of training. The validation set has an overall mean average precision (mAP) of 0.571, a mAP50-95 range, and a final mAP50 of 0.926.

Each model's validation set performance was optimized using Optuna hyperparameter tuning. Model ensembling was used to combine model strengths and improve detection.

3.3.2 Evaluation and Testing

The CNN model's accuracy, precision, and recall were computed to assess the model's overall correctness, its capacity to reduce false positives, and its usefulness in accurately identifying fire images while minimizing false negatives. These metrics offer a comprehensive comprehension of the model's performance in fire classification tasks.

The performance of the YOLOv8 model was assessed using mean average precision (mAP) metrics, specifically mAP50 and mAP50-95. mAP50 measures the mean average precision by evaluating the model's accuracy in detecting and localizing smoke images.mAP50-95 is a more rigorous evaluation that calculates the average precision over several IoU thresholds, ranging from 0.5 to 0.95. This implies that the model must maintain its high level of accuracy over various degrees of overlap.

3.4 Statistical Tests

The study employed t-test analysis to assess the models of this study against benchmarks. This technique involved comparing the internal CNN and YOLO models with other Kaggle models in the same area of research. Bar charts displayed performance comparisons and indicated statistical significance. Realistic samples—with sufficient noise added-—were created from observed metric values that accounted for model variability, resulting in a more comprehensive comparison.

Experimental Setup

4.1 Description of Datasets

The "dataset_image_classification" contains 2,700 aerial and ground-based pictures from government databases, Flickr, and Unsplash. This dataset covers a wide range of environmental circumstances, forest variations, geographical regions, and varied fire dynamics. Image resolutions vary greatly in the collection, averaging 4057×3155 pixels and ranging from 153×206 pixels to 19699×8974 pixels. Image width and height have standard deviations of 1867.47 and 1388.60 pixels, respectively.

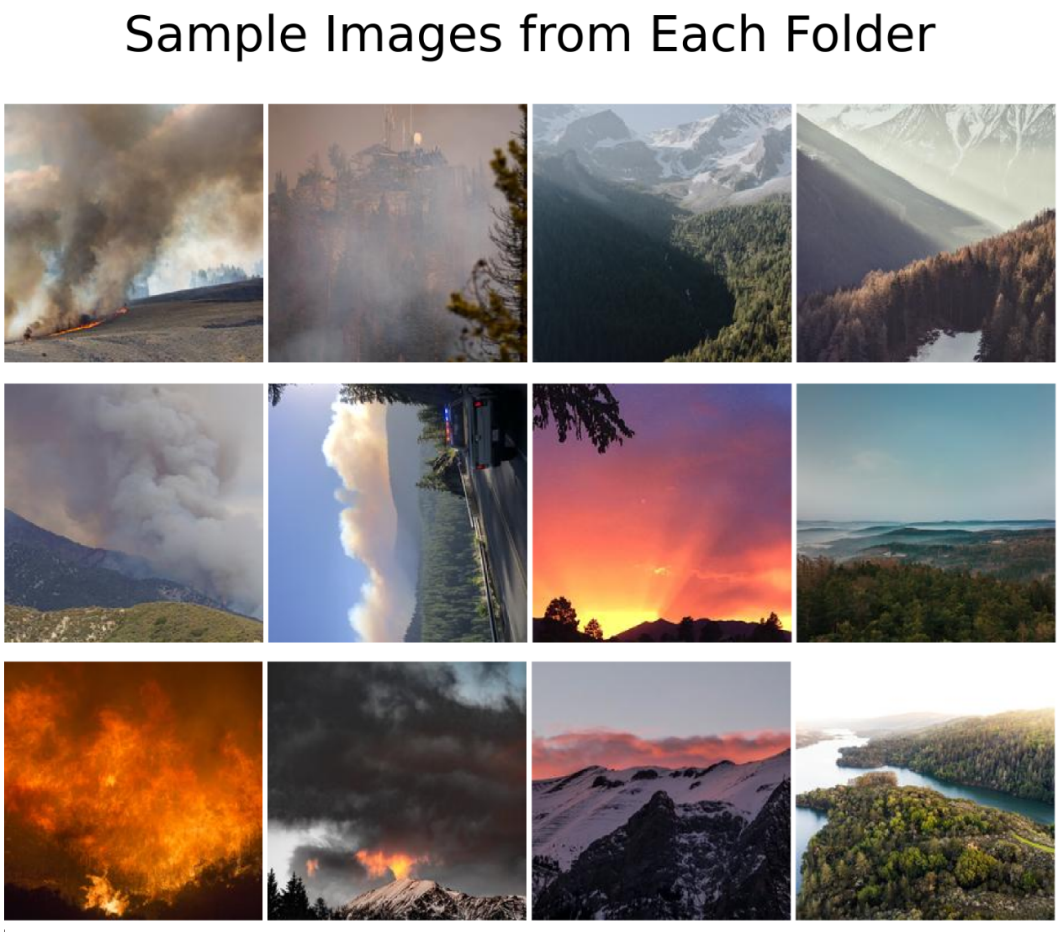

Figure 4.1: Sample Images from Each Folder

Images in the dataset_image_classification were preprocessed to meet CNN model input requirements. Images were reduced to 256×256 pixels while preserving their aspect ratio. Using center cropping, a 224×224 zone was extracted, eradicating any black bars from the resizing operation. This method was based on the observation that most fires were in the

center of the photos. The dataset has three subsets: training (1,887 images), validation (402 images), and testing (410 images).

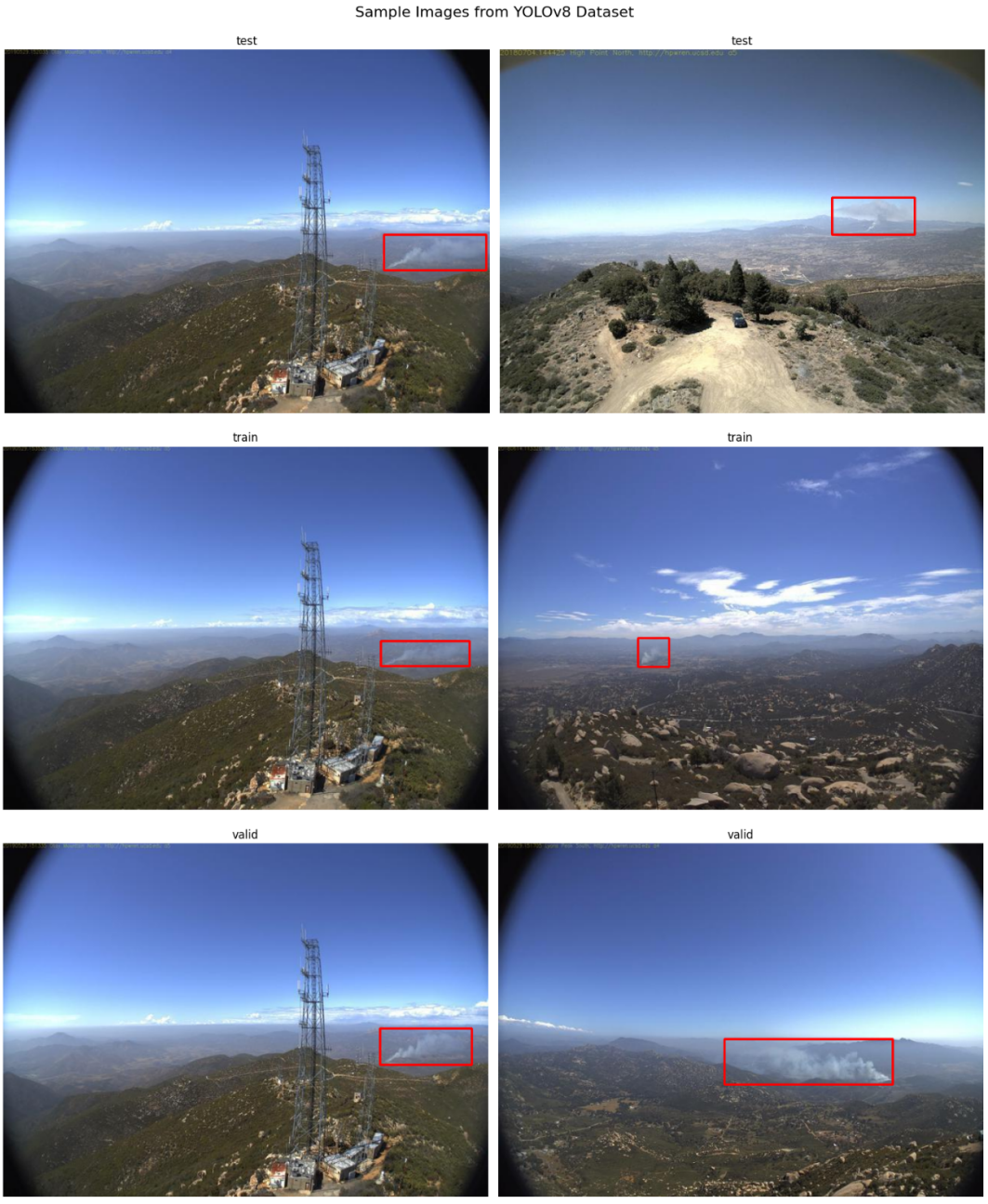

Figure 4.2: Sample Images from Each Folder

4.2 Choosing Architectures

The study employed three different architectures(ResNet50, Inception, and Xception) for fire classification and a YOLOv8 model for real-time smoke and fire detection. All models were

initialized with pre-trained weights from ImageNet, leveraging transfer learning to accelerate training and enhance generalization capabilities.

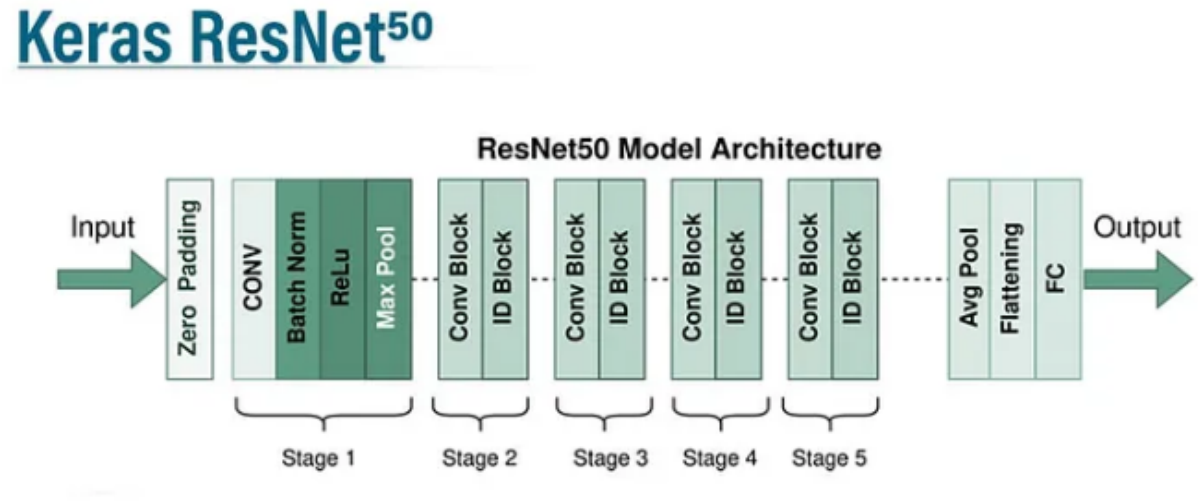

Figure 4.3: Architecture of the Keras ResNet50 model, showing various stages and layers from input through multiple convolutional and identity blocks, to the final output [21].

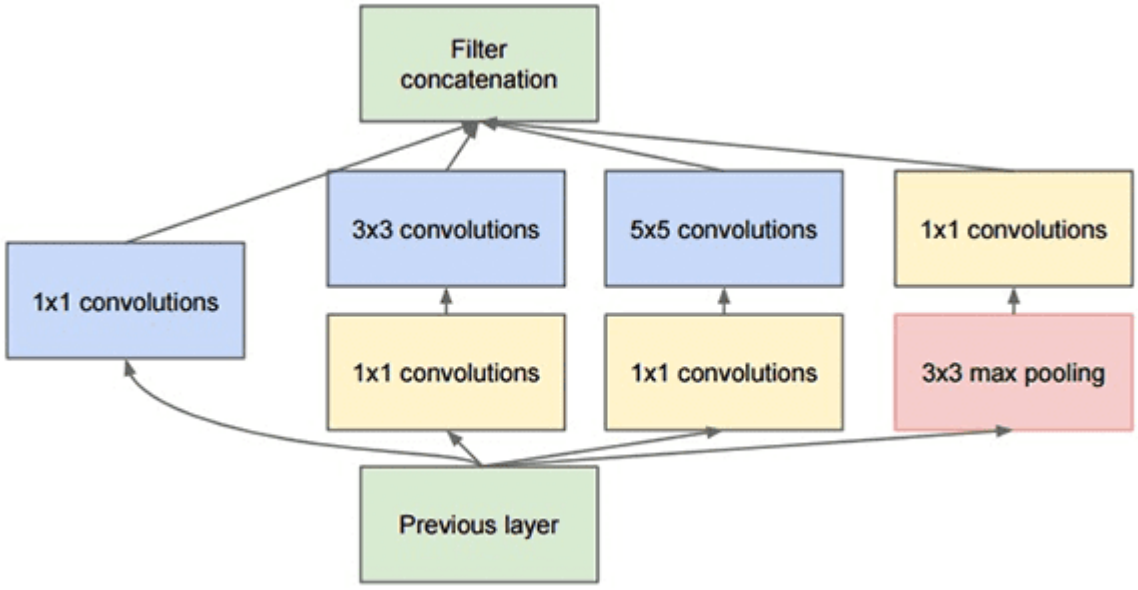

Figure 4.4: Inception modules allow a network to have multiple filter sizes (e.g., 1×1, 3×3, 5×5 convolutions) operating at the same level. This diversity in filter sizes enables the network to capture information at various scales. The concatenated outputs then ensure that the subsequent layers can learn from all these features simultaneously. [22].

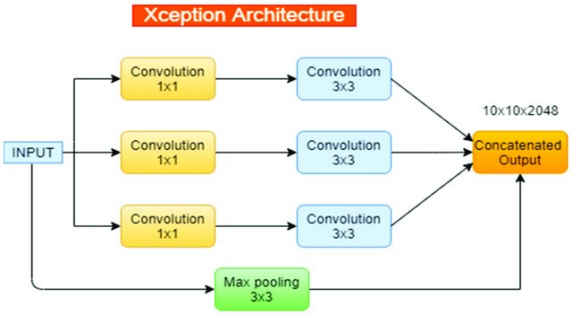

Figure 4.5: The Xception architecture introduces a modification to the standard convolutional layers by implementing depthwise separable convolutions. This model enhances the efficiency and effectiveness in learning features by separating the mapping of cross-channel correlations and spatial correlations in the input feature maps [23].

Figures A.10, A.11, and A.12 illustrate the architectures of ResNet50, Inception, and Xception models, respectively, which were evaluated in this study.

Each pre-trained model was modified to suit the binary fire classification task by adding custom layers. For evaluation purposes, all models employed the Adam optimizer and binary cross-entropy loss function, with accuracy serving as the primary metric. After extensive experimentation, the ResNet50 architecture demonstrated superior performance and was selected for the final system.

For the YOLOv8 component of the system, which focused on real-time detection of smoke and fire, the model architecture is depicted in Figure A.26.

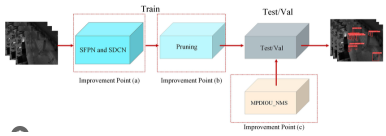

Figure 4.6: High-level overview of the YOLOv8 architecture, showing stages including training with SFPN and SDCN, pruning, and testing/validation [24].

4.3 Model Implementation

4.3.1 ResNet50

Precision reflects the model's ability to correctly identify fire images while minimizing false positives. Recall signifies the model's capacity to detect fire images without missing any, thus minimizing false negatives. The training process and performance of the chosen ResNet50 model can be observed through various plots:

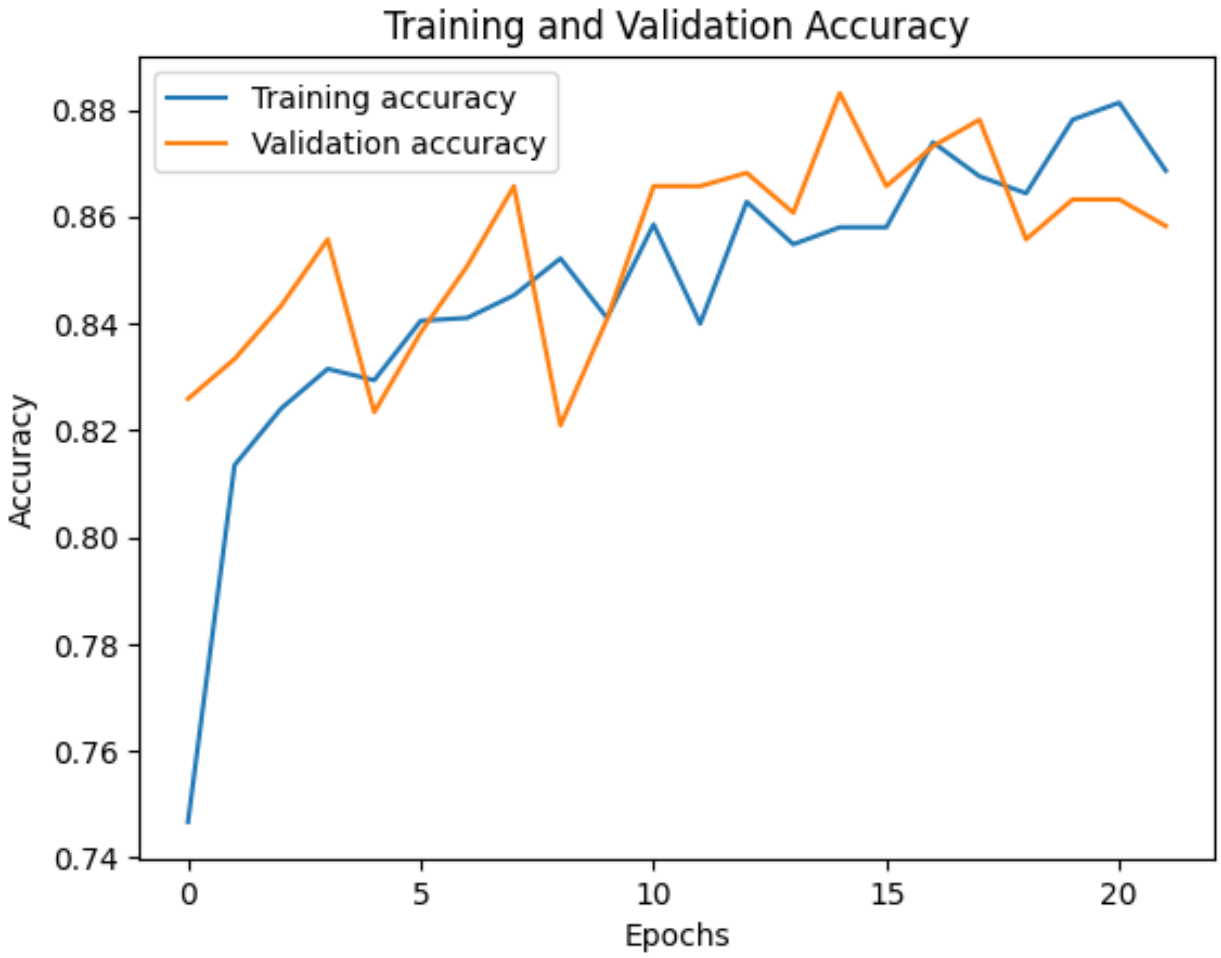

Figure 4.7: Training and Validation Accuracy over epochs, illustrating the CNN model's learning progression.

Figure A.2 illustrates the model's accuracy on both training and validation sets throughout the training process. The upward trend in both lines suggests an improvement in the model's ability to classify fire and non-fire images as training progresses.

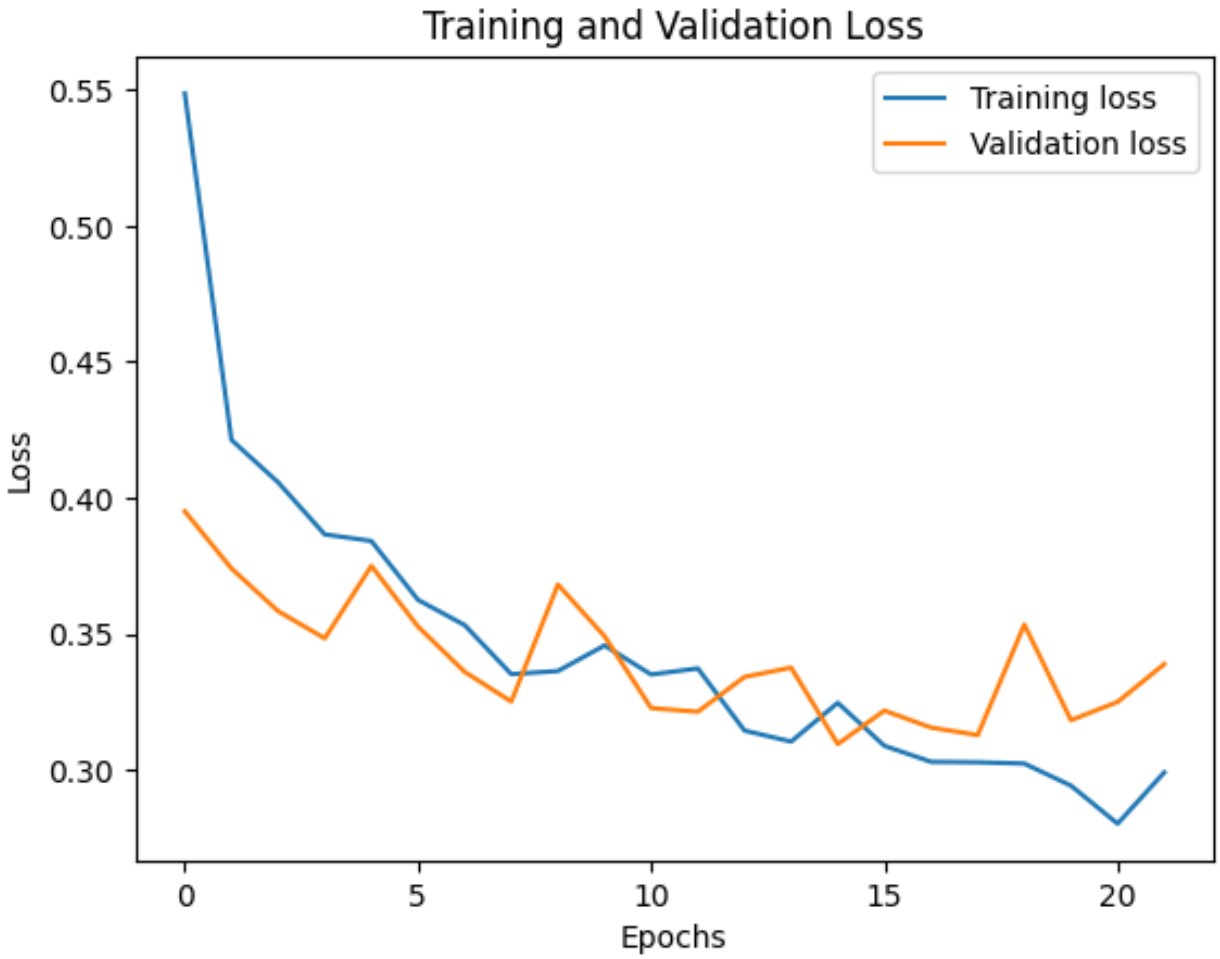

Figure 4.8: Training and Validation Loss over epochs, indicating the model's learning and generalization.

The loss values for both training and validation sets are presented in Figure A.5. The observed downward trend indicates that the model is refining its predictions and reducing errors over time.

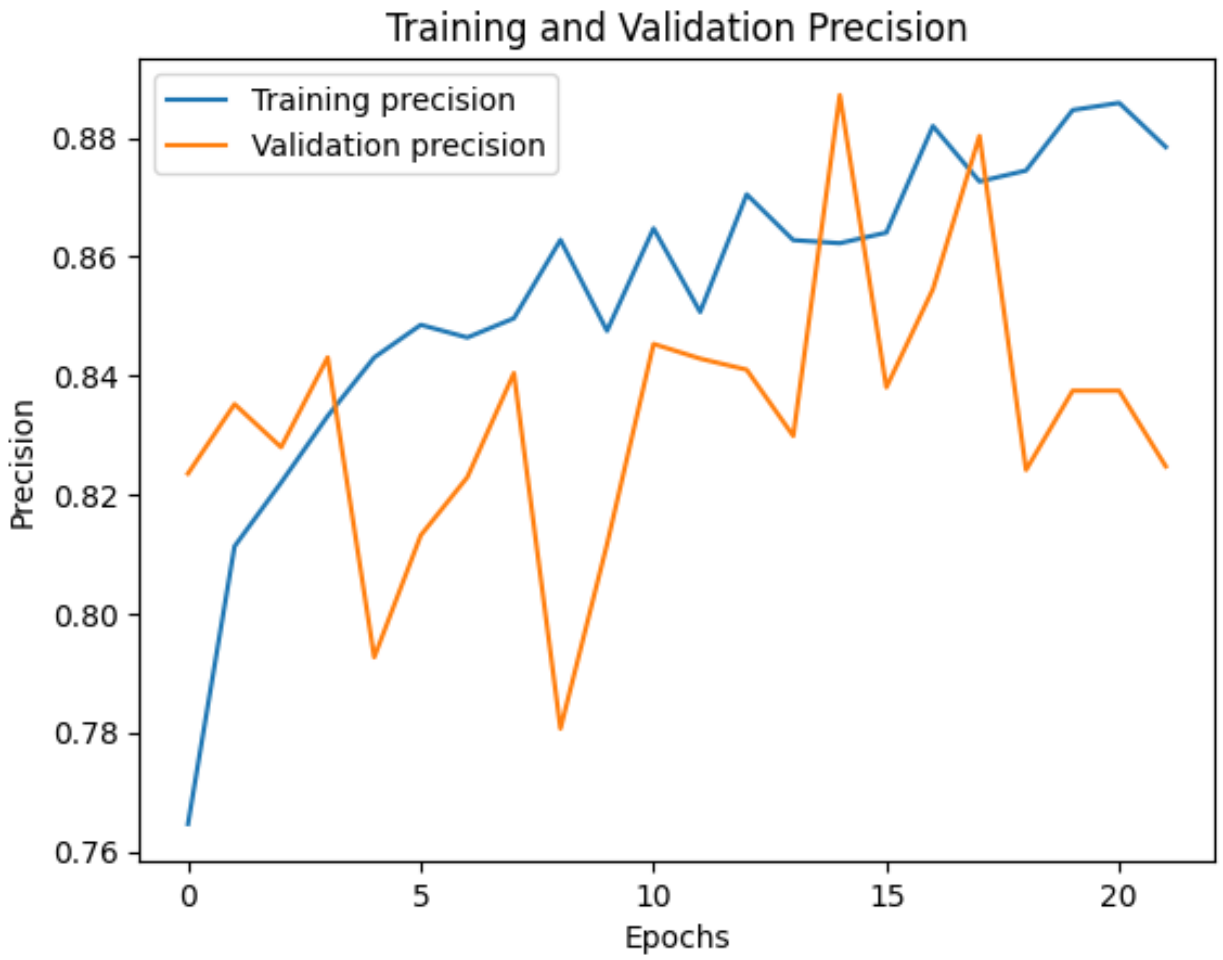

Figure 4.9: Training and Validation Precision over epochs for the CNN model.

Figure A.6 demonstrates the model's precision on training and validation sets. The increasing trend suggests that the model is becoming more precise in its fire predictions, thereby reducing false positives.

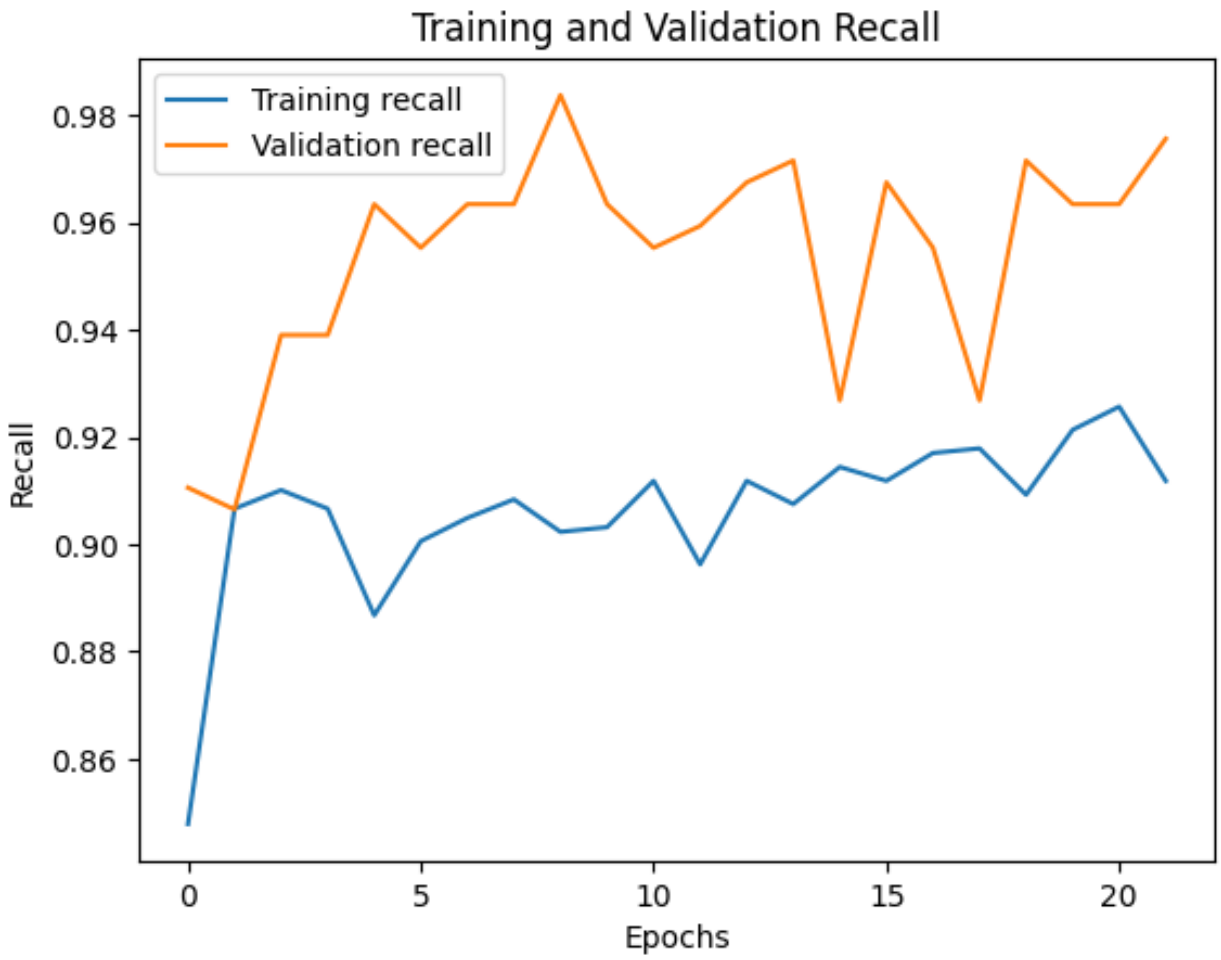

Figure 4.10: Training and Validation Recall over epochs for the CNN model.

The recall values for training and validation sets are shown in Figure A.8. The improving recall indicates that the model is enhancing its ability to identify all instances of fire, thus minimizing false negatives. The consistent improvement across all metrics on both training and validation sets suggests that the model is learning effectively without overfitting.

4.3.2 YOLO-v8 Model

The YOLOv8 model's performance is evaluated using validation predicted batches and validation label batches found below. These visual comparisons provide direct insight into the model's smoke detection capabilities across diverse environmental conditions.

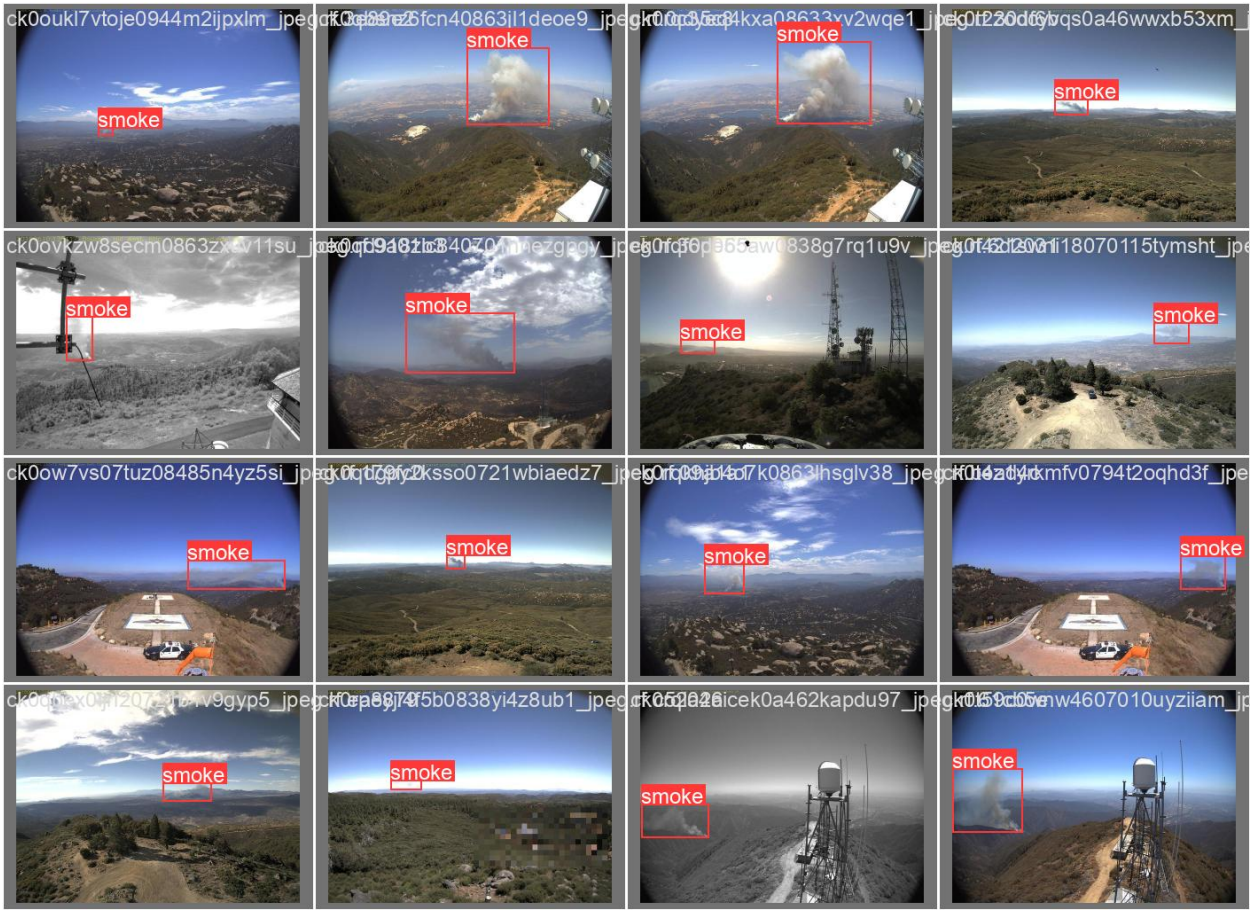

Figure 4.11: presents the ground truth labels for the images, offering a benchmark for assessing the model's accuracy. Comparing these with the predictions reveals the model's strengths in detecting obvious smoke events and its challenges with more subtle atmospheric changes.

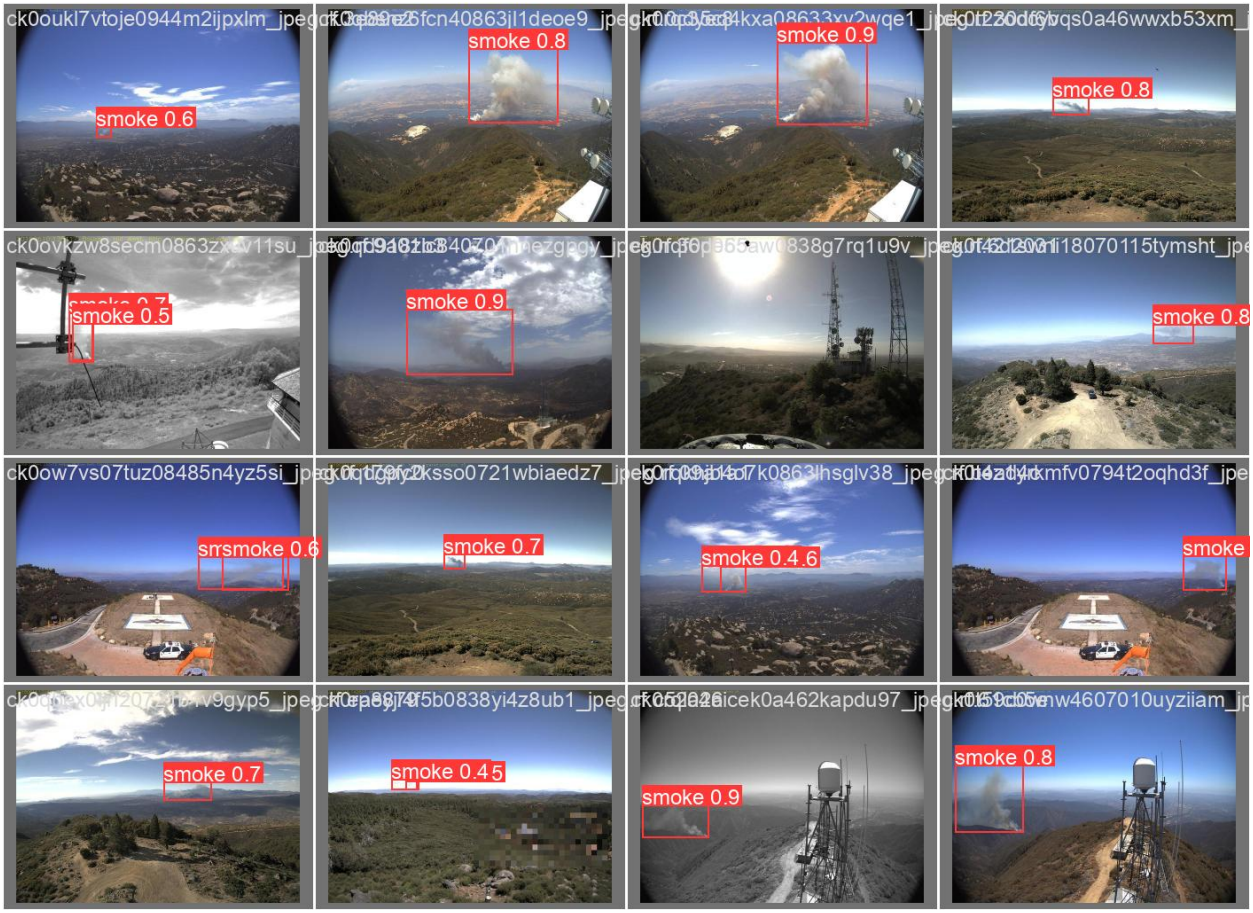

Figure 4.12: displays the model's predictions, showing bounding boxes and confidence scores for detected smoke instances.The predictions demonstrate the model's ability to identify smoke plumes of varying sizes and intensities, from distinct columns to diffuse haze. For example, in the second image of the first row, the model accurately detects a prominent smoke plume rising from the landscape, assigning it a high confidence score.

Especially for clear smoke plumes, labeled and anticipated photographs align well. Still, the model's predictions can differ from ground-truth labels. The model may detect smoke in unlabeled areas or have larger or narrower bounding boundaries. These disparities highlight areas where the model could improve to match expert comments and better distinguish smoke from other meteorological situations. Furthermore

4.4 Hardware and Software Environment

The models were trained on Kaggle's P100 GPU. The P100 GPU's 3584 CUDA cores and 16GB HBM2 memory increased computational performance and memory bandwidth, speeding training.

Models were implemented and trained using Python and common deep learning frameworks. Three fire classification ImageNet models were created and trained using TensorFlow, a Google open-source machine learning framework. The model was built using TensorFlow's Keras API. The yolov8 model utilizes Ultralytics YOLO library.

The study also used OpenCV, NumPy, and Pandas. The tools were used for data administration, preprocessing, analysis, and visualization.

Kaggle had fantastic training and assessment resources, but it also limited studies. The 40-minute GPU session limit per cell was problematic. To finish the experiments on time, training and model complexity have to be "modularized". Without GPU acceleration, training the models on the default Intel Xeon CPU would have taken 10 hours, surpassing the 9-hour session limit.

Finally, GPU memory was a problem. Kaggle's P100 GPU has 16GB VRAM, 2.5GB for dependencies. Model designs and batch sizes (16gb and 32gb) had to be examined to fit in memory. The best model used 15.6GB VRAM, exceeding the GPU's limits and crashing. This is why the study gave two Kaggle contexts. Classifier and object detector.Figure below shows GPU memory footprint.

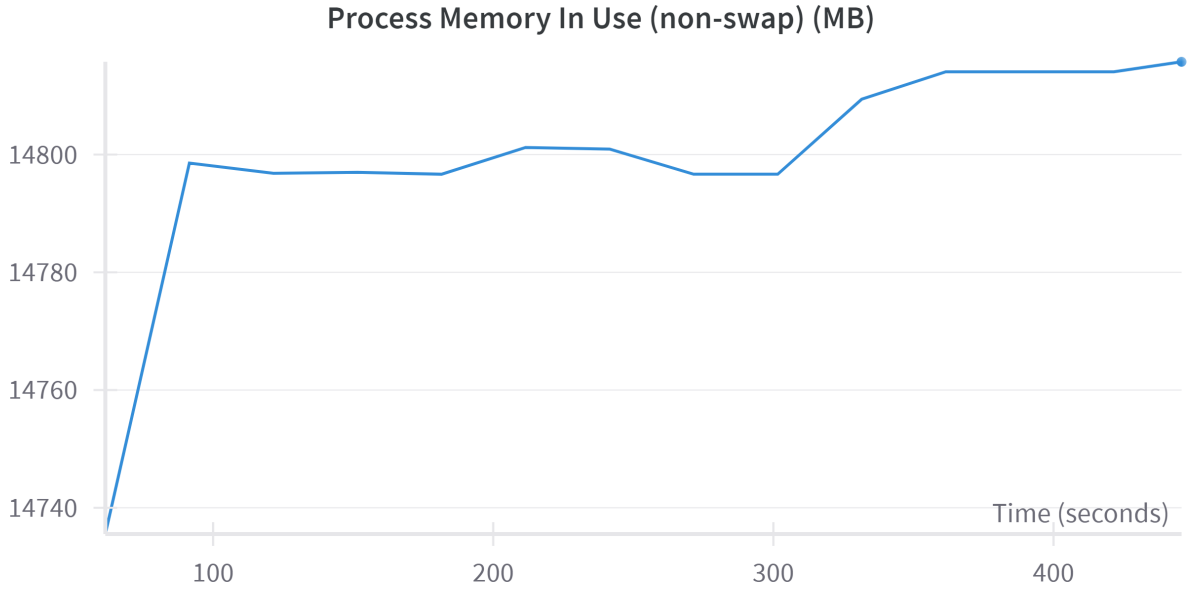

Figure 4.13: Average memory footprint of the models during training and inference.

4.5 Advanced Performance Analysis

4.5.1 ResNet50

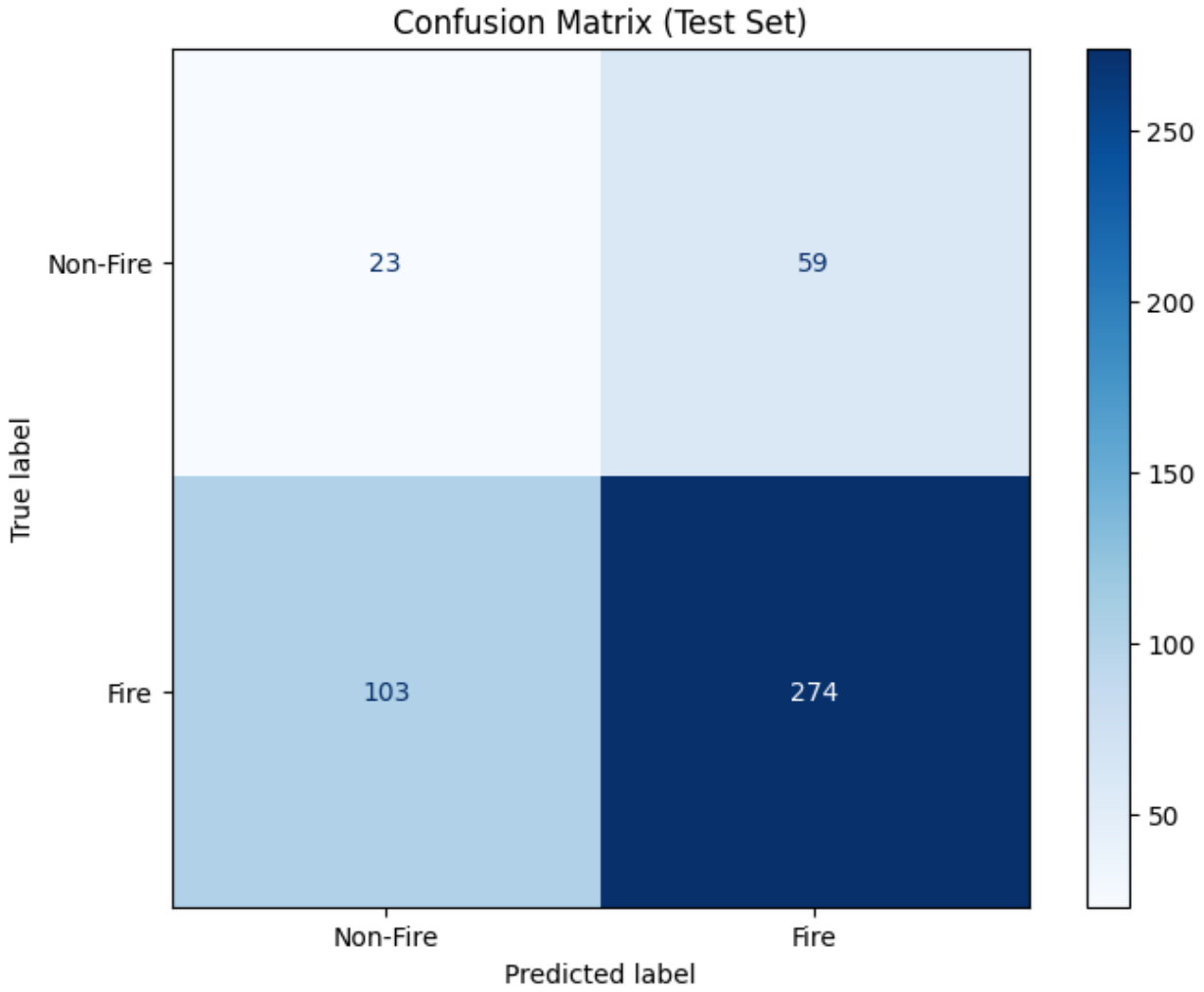

Figure 4.14: Confusion Matrix for the CNN Model. This matrix provides a tabular summary of the model's performance, showing the number of true positive, true negative, false positive, and false negative predictions.

The precision of 82.3% indicates that the model minimizes false alarms. However, the recall of 72.7% suggests that the model misses a quarter of actual fire instances. These false negatives are particularly concerning as they represent potential wildfires that can rage undetected. This is an area that further research should pay attention to.

The performance metrics of CNN model, as presented in Table A.1, offer valuable insights into its effectiveness in wildfire classification.

Additionally, model ensemble techniques were used to combine CNN architectures to improve wildfire detection system performance. The ensemble model uses ResNet50, InceptionV3, and Xception outputs, and it's being trained on the hyperameter space using Optuna.

4.5.2 YOLO-v8 Model

Yolov8 is being tested and evaluated on Precision, recall, mean Average Precision (mAP) at 0.5 and 0.5:0.95 IoU thresholds, and fitness score. The evaluation results are logged and maintained for analysis to track the model's performance across iterations and configurations.

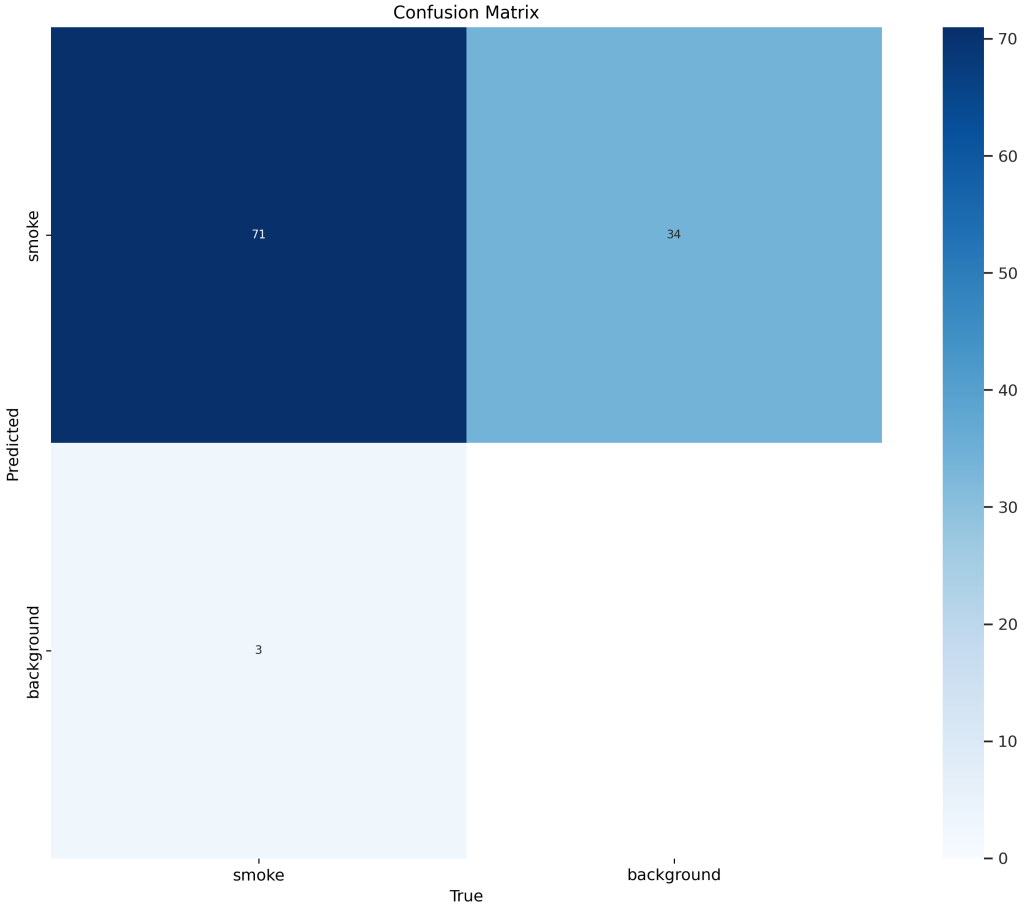

Figure 4.15: Confusion Matrix for the YOLO Model. This matrix provides a summary of the model's performance in detecting smoke instances, including true positives, true negatives, false positives, and false negatives.

The matrix reveals that the model correctly identified smoke in 71 instances (true positives). However, it also shows that the model missed smoke in 3 cases (false negatives), which, while relatively low, represents a critical area for improvement given the high stakes of wildfire detection.

On the other hand, the model incorrectly classified 34 instances as smoke when they were actually background (false positives). This suggests the model may lead to false alarms. Interestingly, the matrix does not show any true negatives.

In addition to hyperparameter modification, model ensembling improved smoke detection. The algorithm trains three YOLOv8 models using Optuna's best hyperparameters. The best model based on validation, mAP@0.5 is chosen. Table A.2 presents the performance metrics of the final YOLO model on the test dataset.

Furthermore, here are some various sample tests from the effectiveness in detecting smoke from a ground-level and aerial perspective

Figure 4.16: Wide aerial view smoke detection using the YOLO model.

Figure 4.17: Ground view smoke detection using the YOLO model.

4.6 Statistical Analysis

Figure 4.18 displays the comparison of CNN models. The internal Convolutional Neural Network (CNN) demonstrated superior performance compared to the Kaggle model in all measures, with a statistical significance of p ¡ 0.001. The most significant enhancements were observed in recall and F1 score, indicating greater performance in accurately detecting true positives while preserving a favorable trade-off with precision. The accuracy metric demonstrated a significant enhancement, suggesting superior ability in classifying data.

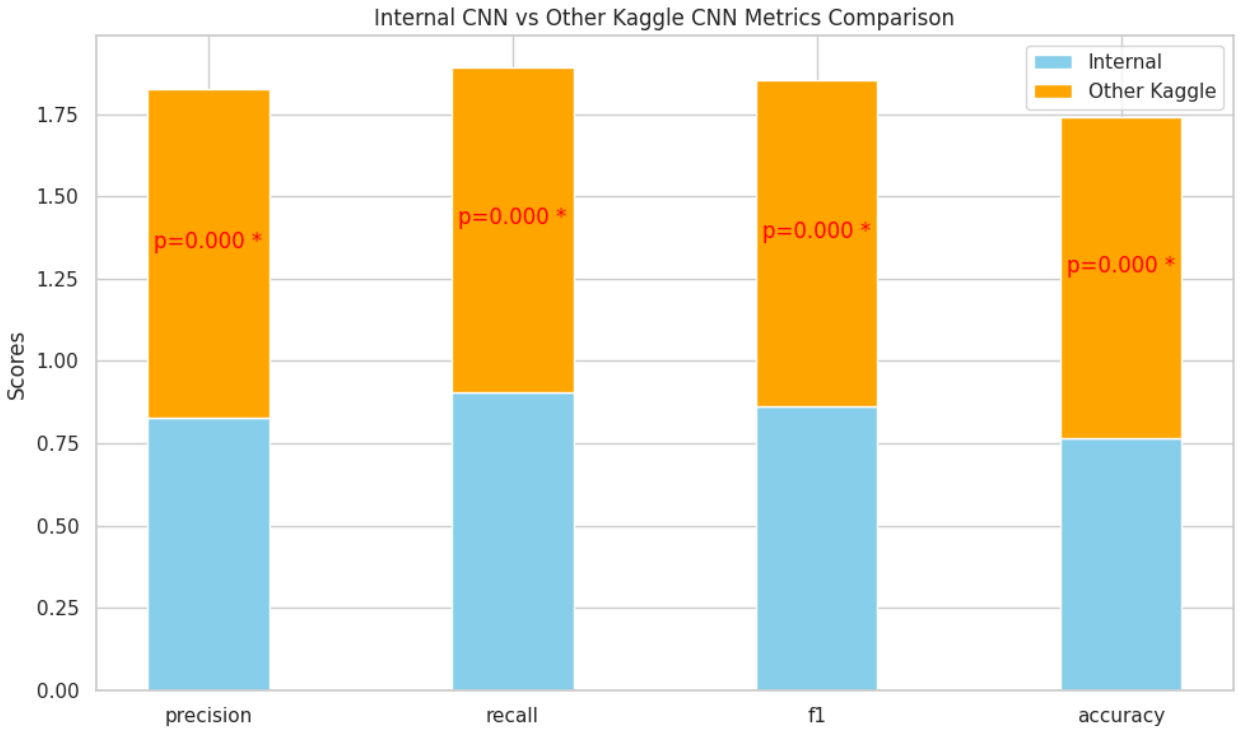

Figure 4.18: P-Test Results for CNN Model Comparison

The results for YOLO models (Figure 4.19) were more complicated. The internal model exhibited considerably higher precision (p = 0.004) and mean Average Precision at IoU 0.5:0.95 (mAP@0.5:0.95, p ¡ 0.001). These enhancements indicate improved accuracy in localizing and classifying objects at different IoU levels. Notably, there was no statistically significant distinction in recollection (p = 0.377) or mAP@0.5 (p = 0.348), suggesting similar performance in these aspects.

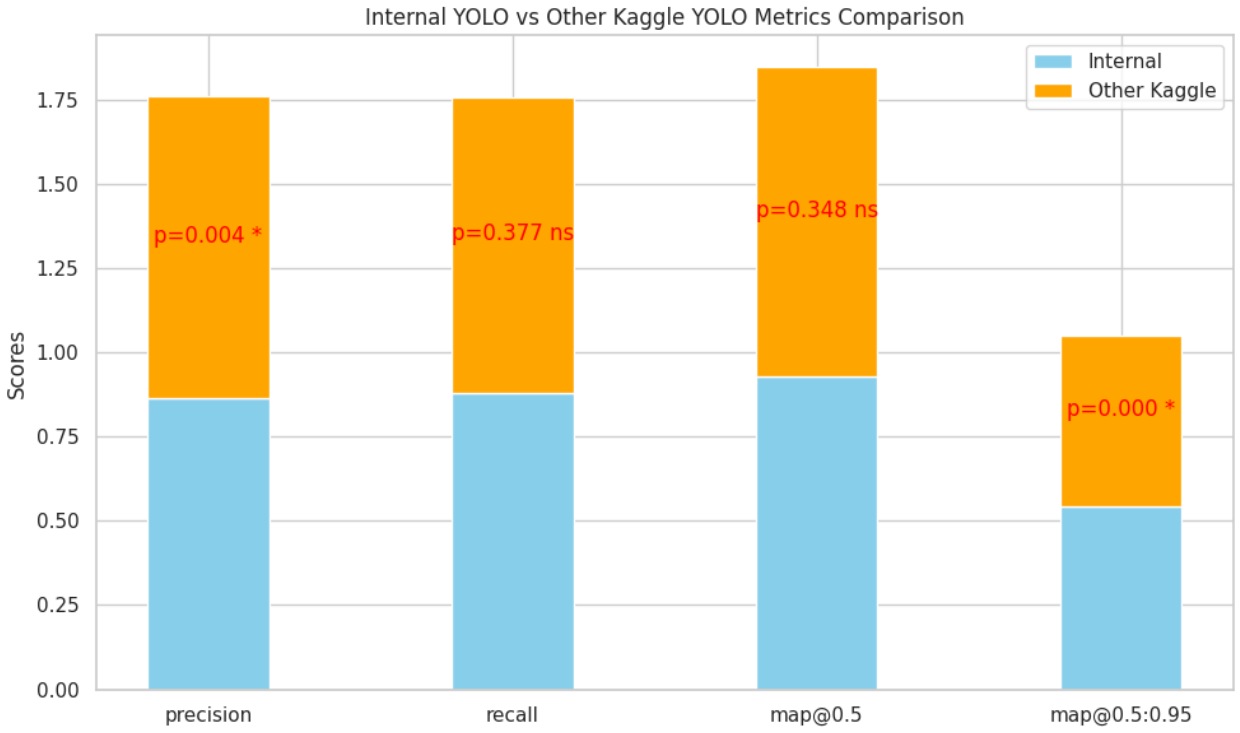

Figure 4.19: P-Test Results for YOLO Model Comparison

The significant disparity in mAP@0.5:0.95 performance, with the internal model exhibiting a notably higher score, indicates that the internal YOLO model excels at preserving accuracy when subjected to more stringent IoU criteria.

The results highlight the efficacy of the model development strategy, specifically for CNN-based classification. The enhancements made to the YOLO model, although more discerning, exhibit significant progress in crucial aspects of object detection accuracy.

Evaluation Results and Discussion

5.1 Insights on ResNet50

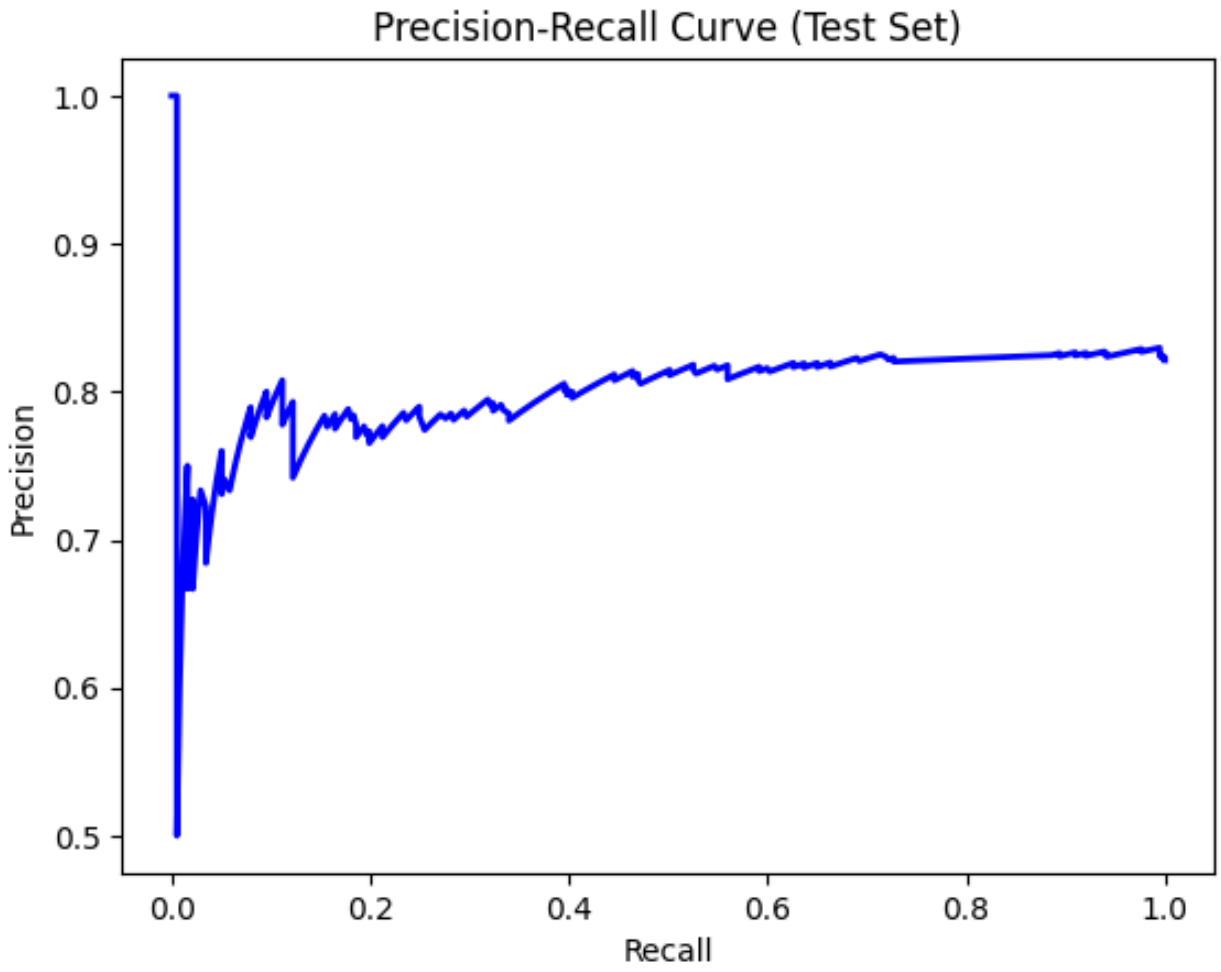

Figure 5.1: Precision-Recall Curve (Test Set - CNN Model). This figure presents the precision-recall curve for the CNN model evaluated on the test set. It shows the trade-off between precision and recall at different classification thresholds, providing insights into the model's performance across various operating points.

Figure A.7 further illustrates the model's performance, showing high precision at low recall levels and consistent performance across various thresholds. It shows the ROC curve for the test set, revealing the model's discriminative ability. The curve's proximity to the diagonal suggests there's room for improvement in the model's ability to distinguish between fire and non-fire instances.

During training and evaluation, the CNN model has good accuracy, AUC, precision, and recall. These metrics improve, and training and validation curves are similar, meaning the model generalizes well to unseen data. The test set results—ROC curve and uncertainty matrix—indicate model classification skill gaps. Future research may extend training data, validate model architecture or hyperparameters, use data augmentation or transfer learning, and incorporate domain-specific knowledge to improve model performance on new data.

5.2 Insights on Yolo-v8

The YOLOv8 model's performance can be assessed through various metrics and visualizations. In Figure A.23, the model's robustness even in distorted ground-view conditions, which is important for real-world applications where image quality may vary.

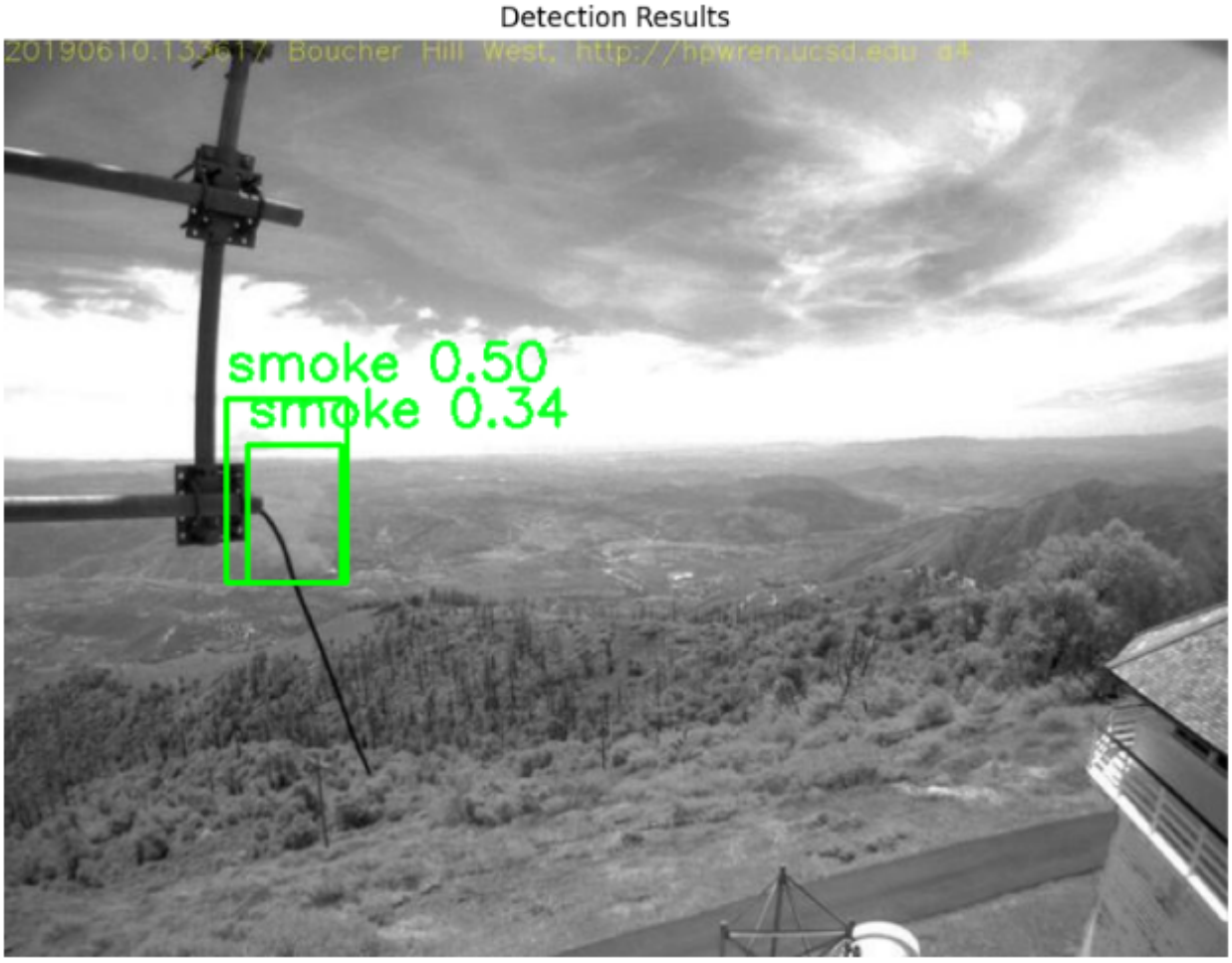

Figure 5.2: A challenging scenario where the YOLO model doesn't predict continuous smoke.

Figure A.24 presents a challenging scenario where the model struggles to predict continuous smoke, highlighting areas for potential improvement in future iterations.

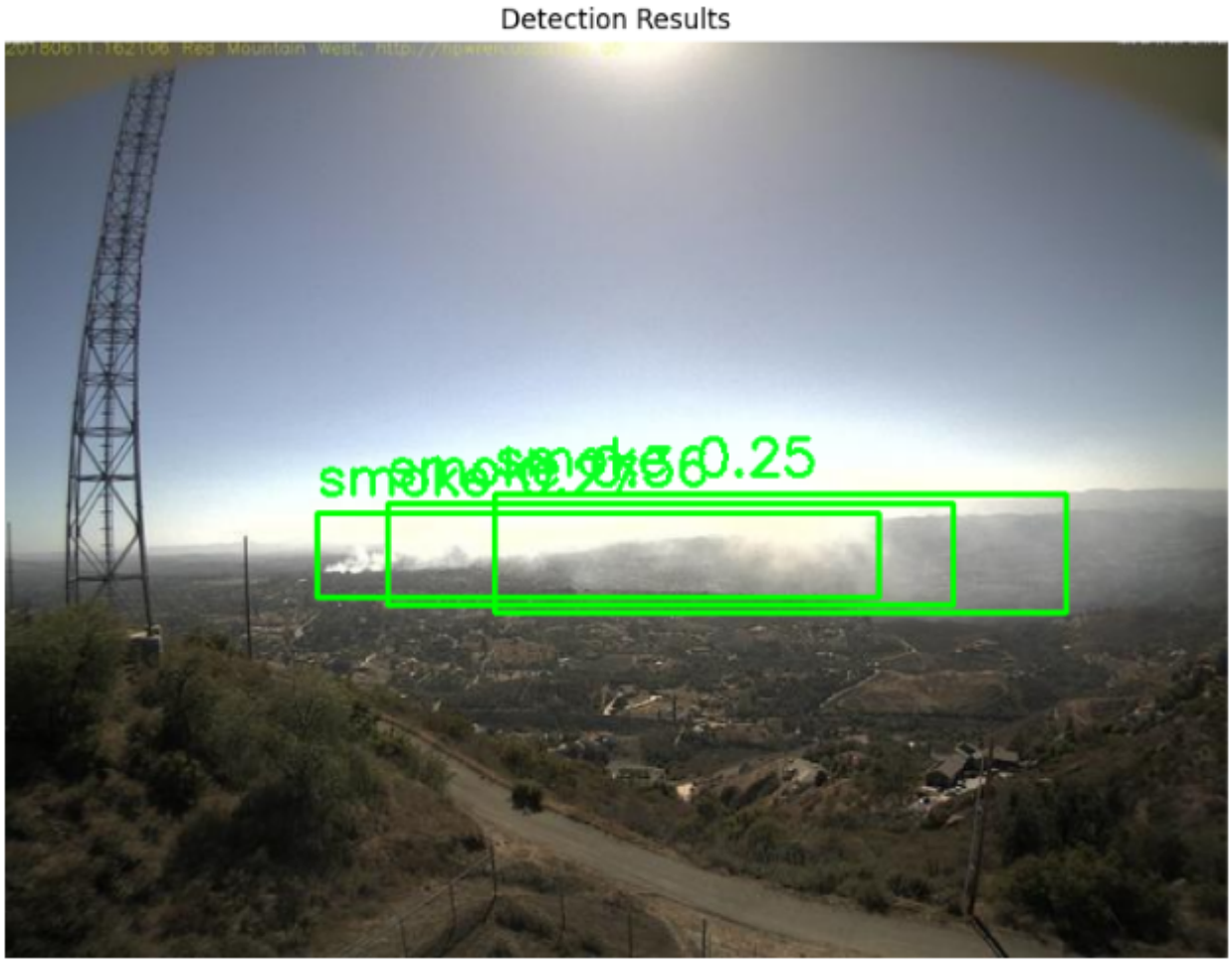

Figure 5.3: A challenging scenario where the YOLO model doesn't predict continuous smoke.

In conclusion, Precision, Recall, mean Average Precision (mAP) at 0.5 and 0.5:0.95 IoU thresholds, and fitness score evaluate the YOLOv8 model's smoke detection ability. The confusion matrix (Figure A.16) shows 71 true positives, 3 false negatives, 34 false positives, and no true negatives. This signals strong detection but oversensitivity that could cause false alerts. Multiple sample tests prove model's efficacy. The model captures aerial views for wide-area monitoring and detects ground-level objects. It is also robust under distorted viewing situations, suggesting its trustworthiness in difficult environments. On the other hand, it indicates that it struggles to forecast continuous smoke, suggesting that it has to be improved.

5.3 Error Analysis and Model Limitations

Kaggle's computing limitations affected CNN and YOLO wildfire detection model development and deployment. These constraints affected model structure, strategy, and performance. For example, advanced post-processing technologies like NMS risk kernel disruption

and time overrun. YOLO's simple post-processing may have hindered forecast refinement and false positive reduction.

Extensive experimentation revealed the CNN and YOLO models' optimal epochs and trials for performance and computer economy. Iterative testing showed that on average 25 epochs, 3 Optuna trials and 3 models yielded the greatest results, completing models in approximately 35 minutes.

Several strategies were used to overcome restrictions and improve the model. Data augmentation on-the-fly increased training data diversity, without putting more strain in the memory, allowing models to learn more variables and generalize. Extracting relevant characteristics from pre-trained models reduced computational overhead. Complexity and resource efficiency were balanced in model designs, hyperparameters, and optimization methods.

Conclusion & Future Directions

6.1 Summary of Key Findings

Ultimately, using the best-performing ResNet50 model along with YOLOv8 showed how future studies could advance the field. After training on aerial and ground photographs, ResNet50 achieved 90% recall on the test set with 76% accuracy. On the other hand, YOLOv8 achieved an average precision (AP) of 93.5% and an F1 score of 0.85 at the optimal confidence level.

Pre-trained models, preprocessing and post-processing of the data, hyperparameter tuning, model ensembling, metric evaluation, and highlighted model strengths and weaknesses.

This study also highlighted how Kaggle's computing resources limit the models' architectures and training methods. The resources were optimized due to the limited availability of GPU units, time, and memory. These constraints degrade performance compared to more competent computational environments.

6.2 Future Directions

As previously mentioned, the study used only aerial and ground-based image data. Future researchers may employ additional sources and types of data like thermal imaging, satellite imagery, and meteorological data.

Furthermore, deploying these models on the cloud and using them in a real-world scenario is another possibility. Working with the local Fire Department authorities and integrating the models into wildfire monitoring infrastructures can demonstrate the system's practicality. Field-testing and user feedback can help wildfire managers improve.

In conjunction to this, an easy-to-use interface that visualizes fire and smoke occurrences with location, intensity, and spread, may help personnel make decisions and stay informed.

Lastly, future researchers can boost wildfire detection system processing efficiency and scalability. This may happen by employing model compression, quantization, and edge computing to deploy models on resource-constrained devices and enable distant real-time processing.

Bibliography

[1] V. Papaioannou and S. Siopi, "Observational evidence of the need for gender-sensitive approaches in disaster management: The case of mati wildfire in greece," *MDPI*, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/13/3/1556

[2] Y. Panagopoulos and A. Kalogeropoulos, "Mediterranean wildfires' effect on soil quality and properties: A case from northern euboea, greece," *MDPI*, 2024. [Online]. Available: https://www.mdpi.com/2073-445X/13/3/325

[3] E. F. F. I. S. (EFFIS), "Fire consumes large swaths of greece," *NASA Earth Observatory*, 2021. [Online]. Available: https://earthobservatory.nasa.gov/images/148682/fire-consumes-large-swaths-of-greece

[4] Protothema, "Fire department representative: "every ten minutes we have a new fire; there were 300 new fires."," *Protothema*, 2024. [Online]. Available: https://en.protothema.gr/2024/06/19/fire-department-representative-every-ten-minutes-we-have-a-new-fire-there-were-300-new-fires/

[5] Euronews, "How damaging were greek wildfires? experts explain how destructive they were," *Euronews*, 2023. [Online]. Available: https://www.euronews.com/green/2023/08/02/greek-wildfires-have-unleashed-the-same-co2-emissions-in-july-as-over-222000-cars-in-a-yea

[6] A. Jazeera, "The greek wildfires: What went wrong and what can be fixed?" *Al Jazeera*, 2021. [Online]. Available: https://www.aljazeera.com/opinions/2021/8/19/the-greek-wildfires-and-the-way-forward

[7] NCBI, "Development of a deep learning-based surveillance system," *NCBI*, 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10931456/

[8] I. Xplore, "Forest fire detection using convolution neural networks," *IEEE Xplore*, n.d. [Online]. Available: https://ieeexplore.ieee.org/document/10031936/

[9] N. Communications, "Urgent call for comprehensive governmental climate action against wildfires," *Nature Communications*, 2023. [Online]. Available: https://www.nature.com/articles/s44168-023-00076-z

[10] E. G. C. Giannaros, I. Agathangelidis, "The heat-alarm project: Development of a heat–health warning system in greece," *Environmental Sciences Proceedings*, 2023. [Online]. Available: https://www.mdpi.com/2673-4931/26/1/88

[11] M. M. M. S. LD Kuratle, I Dallo, "What does my technology facilitate? a toolbox to help researchers understand the societal impact of emerging technologies in the context of disasters," *Seismica*, 2023. [Online]. Available: https://seismica.library.mcgill.ca/article/download/1144/1349/8448

[12] S. P. J. S. SA Boukabara, V Krasnopolsky, "Outlook for exploiting artificial intelligence in the earth and," *Bulletin of the American Meteorological Society*, 2023. [Online]. Available: https://journals.ametsoc.org/downloadpdf/journals/bams/aop/BAMS-D-20-0031.1/BAMS-D-20-0031.1.pdf

[13] C. Publishing, "A lightweight method for intelligent detection of four extreme wildfires," *CSIRO Publishing*, n.d. [Online]. Available: https://www.publish.csiro.au/WF/justaccepted/WF23044

[14] "Blind channel identification aided generalized automatic modulation recognition based on deep learning," *ResearchGate*, 2019. [Online]. Available: https://www.researchgate.net/publication/335086346_Blind_Channel_Identification_Aided_Generalized_Automatic_Modulation_Recognition_Based_on_Deep_Learning

[15] arXiv, "Fire detection from image and video using yolov5," *arXiv*, 2023. [Online]. Available: https://arxiv.org/pdf/2310.06351

[16] Medium, "Wildfire detection using yolo," *Medium*, 2023. [Online]. Available: https://medium.com/bokchilab/wildfire-detection-using-yolo-ecf9b93b8063

[17] A. Acharya, "Yolo object detection explained: Evolution, algorithm, and applications," *Encord.com; Encord Blog*, 2023. [Online]. Available: https://encord.com/blog/yolo-object-detection-guide/

[18] E. Proceedings, "Fire detection and segmentation using yolov5 and u-net," *EUSIPCO Proceedings*, 2021. [Online]. Available: https://eurasip.org/Proceedings/Eusipco/Eusipco2021/pdfs/0000741.pdf

[19] I. E. madafri, "The wildfire dataset," *Kaggle.com*, 2023. [Online]. Available: https://www.kaggle.com/datasets/elmadafri/the-wildfire-dataset

[20] A. F. Mankind, "Wildfire smoke object detection dataset," *Roboflow*, 2022. [Online]. Available: https://public.roboflow.com/object-detection/wildfire-smoke

[21] A. Serej, "Resnet-50," *Medium*, 2024. [Online]. Available: https://medium.com/@arashserej/resnet-50-83b3ff33be7d

[22] PyImageSearch, "Imagenet: Vggnet, resnet, inception, and xception with keras," *PyImageSearch*, 2017. [Online]. Available: https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

[23] "A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks," *ResearchGate*, 2021. [Online]. Available: https://shorturl.at/zfjrP

[24] A. Names, "Unmanned aerial vehicles general aerial person-vehicle recognition based on improved yolov8s algorithm," *ResearchGate*, 2024. [Online]. Available: https://www.researchgate.net/publication/378923575_Unmanned_Aerial_Vehicles_General_Aerial_Person-Vehicle_Recognition_Based_on_Improved_YOLOv8s_Algorithm/figures?lo=1

Appendices

A.1 Figures

Figure A.1: Within the CNN classification model, examples of images from each folder are listed.

Figure A.2: Training and Validation Accuracy. This figure illustrates the training and validation accuracy of the CNN model over the epochs during the training process. It shows the progression of accuracy as the model learns from the training data and its performance on the validation set.

Figure A.3: Training and Validation AUC. This figure presents the Area Under the Curve (AUC) for both the training and validation sets of the CNN model over the epochs. AUC is a metric that measures the model's ability to discriminate between classes, with higher values indicating better performance.

Figure A.4: Confusion Matrix (CNN Model). This figure displays the confusion matrix for the CNN model evaluated on the test set. The confusion matrix provides a tabular summary of the model's performance, showing the number of true positive, true negative, false positive, and false negative predictions.

Figure A.5: Training and Validation Loss. This figure shows the training and validation loss of the CNN model over the epochs. The loss curves indicate how well the model is learning and generalizing, with lower values suggesting better performance.

Figure A.6: Training and Validation Precision. This figure illustrates the precision of the CNN model for both the training and validation sets over the epochs. Precision measures the proportion of true positive predictions among all positive predictions.

Figure A.7: Precision-Recall Curve (Test Set - CNN Model). This figure presents the precision-recall curve for the CNN model evaluated on the test set. It shows the trade-off between precision and recall at different classification thresholds, providing insights into the model's performance across various operating points.

Figure A.8: Training and Validation Recall. This figure displays the recall of the CNN model for both the training and validation sets over the epochs. Recall measures the proportion of true positive predictions among all actual positive instances.

Figure A.9: ROC Curve (Test Set - CNN Model). This figure shows the Receiver Operating Characteristic (ROC) curve for the CNN model evaluated on the test set. The ROC curve plots the true positive rate against the false positive rate at different classification thresholds, providing an overall assessment of the model's discriminative power.

Figure A.10: This image depicts the architecture of the Keras ResNet50 model. It shows the various stages and layers of the neural network, from input through multiple convolutional and identity blocks, to the final output [21].

Figure A.11: Inception modules allow a network to have multiple filter sizes (e.g., 1×1, 3×3, 5×5 convolutions) operating at the same level. This diversity in filter sizes enables the network to capture information at various scales. The concatenated outputs then ensure that the subsequent layers can learn from all these features simultaneously. [22].

Figure A.12: The Xception architecture introduces a modification to the standard convolutional layers by implementing depthwise separable convolutions. This model enhances the efficiency and effectiveness in learning features by separating the mapping of cross-channel correlations and spatial correlations in the input feature maps [23].

Figure A.13: This image illustrates a simple CNN structure which would typically include layers such as input, convolutional, pooling, and fully connected layers, culminating in an output layer. Each layer has its specific role, such as feature extraction and non-linear transformations, which are essential for tasks like image classification. [14].

Figure A.14: Sample Images from Each Folder

Figure A.15: Example Detections (YOLO Model). This figure presents a visual representation of the YOLO model's detections on sample images from the test set. It displays bounding boxes around detected smoke instances, along with their corresponding confidence scores.

Figure A.16: Confusion Matrix (YOLO Model). This figure shows the confusion matrix for the YOLO model evaluated on the test set. It provides a summary of the model's performance in detecting smoke instances, including true positives, true negatives, false positives, and false negatives.

Figure A.17: Precision-Recall Curve (Test Set - YOLO Model). This figure illustrates the precision-recall curve for the YOLO model evaluated on the test set. It demonstrates the trade-off between precision and recall at different confidence thresholds for smoke detection.

Figure A.18: F1 Curve (YOLO Model). This figure illustrates the F1 score for the YOLO model evaluated on the test set. It demonstrates the trade-off between precision and recall at different confidence thresholds for smoke detection.

Figure A.19: Precision Curve (YOLO Model). This figure illustrates the precision score for the YOLO model evaluated on the test set.

Figure A.20: Recall Curve (YOLO Model). This figure illustrates the recall score for the YOLO model evaluated on the test set.

Figure A.21: A sample of Yolo-v8 smoke detection conducted from a wide aerial view from wandB

Figure A.22: A sample of Yolo-v8 smoke detection conducted from a wide ground view from wandB

Figure A.23: A sample of Yolo-v8 smoke detection conducted with a disrtorted view from wandB

Figure A.24: A sample of Yolo-v8 smoke detection facing difficult scenario in which yolov8 doesn't predict a continuous smoke from wandb

Figure A.25: This diagram illustrates a basic YOLO architecture, which is used for real-time object detection. The network processes the whole image during inference, predicts multiple bounding boxes and class probabilities for these boxes simultaneously, making it extremely fast and suitable for real-time applications. [17]

Figure A.26: This diagram illustrates a high level overview of the yolov8 architecture for image processing. It shows stages including training with SFPN and SDCN, pruning, and testing/validation. There are three "Improvement Points" labeled (a), (b), and (c), with (c) involving MPDIOU NMS. [24]

Figure A.27: YOLO Model Ground Truth (eg Batch 0). This figure shows the ground truth labels for a batch of validation images used in the YOLO model.

Figure A.28: YOLO Model Predictions (eg Batch 0). This figure shows the predicted labels for a batch of validation images used in the YOLO model.

Figure A.29: A sample of the average memory footprint of this study's Kaggle environment from wandb

A.2 Tables

Table A.1: CNN Model Performance Metrics(Test)

Metric	Value
Accuracy	0.76
Precision	0.82
Recall	0.90
F1 Score	0.74

Table A.2: YOLO Model Performance Metrics (Test)

Metric	Value
Precision	0.865
Recall	0.878
mAP@0.5	0.927
mAP@0.5:0.95	0.545
Fitness(Not supported by Yolov8)	N/A

Table A.3: Computational Resource Constraints

Resource	Value
GPU Units	30h/week
Time per Session	30-40 min
RAM Usage	10-13 GB
CPU Memory	24-29 GB

A.3 Experimental Setup's Coding Environment and Google Drive Files

A.3.1 Kaggle Environment

The majority of the processes and techniques in this study were implemented using the Kaggle environment. The specific notebook used for this purpose can be accessed through the following link:

YOLO Enviroment Execution:fire_smoke_wildfire_detection_final

CNN Enviroment Execution:fire_smoke_wildfire_detection_final_cnn_runtime

These Kaggle notebook contains the code and resources necessary to reproduce the experiments and analyses presented in this study. They include the implementation of the CNN and YOLO models, data preprocessing steps, model training, evaluation, and advanced performance analysis techniques. The research employed two different Kaggle environments to combat the limited GPU memory limits offed By Kaggle.

A.3.2 Google Colab Environment (CNN)

In addition to the Kaggle environment, a separate Google Colab environment was used specifically for the CNN model. This environment focuses on image preprocessing tasks and can be accessed through the following link:

Image Preprocessing with Parallel Processing and Aspect Ratio Preservation

The Google Colab notebook provides a dedicated environment for preprocessing the images used in the CNN model. It includes code for resizing the images to a resolution of 224x224 pixels by preserving their aspect ratio and original proportions to avoid distortions, which is recommended for compatibility with the ImageNet dataset.

A.3.3 Google Drive Link for files

Finally, here is the Google Drive link to access the complete project: Neural Networks Fire classification and Smoke detection project folder