Pontificia Universidad Católica del Perú

Faculty of Science and Engineering

Deep Learning (1INF52)

Project Report

Cántaro Márquez, Patricia Natividad	20210907
Nicho Manrique, Saymon Estefano	20211866
Zegarra Barrenechea, Carlos Eduardo	20216177

March 18, 2025

Contents

1 Introduction

Wildfires pose a serious threat to the environment and public health. Their frequency has increased due to climate change and growing human activity in forested areas [1]. Wildfires can be triggered naturally or by human negligence, causing severe damage to infrastructure, biodiversity, and the economy.

The early detection of wildfires is crucial to enable a faster response from firefighting crews. Traditional sensor-based or satellite-based methods often face limitations in coverage or timeliness.

Deep learning has proven highly effective in image analysis, especially for fire or smoke detection, surpassing many traditional approaches. In this work, we propose a **deep learning model** capable of **real-time wildfire detection** from drone-captured images, aiming for faster and more efficient identification of active fires.

2 Problem Statement

Although deep learning has led to remarkable progress in wildfire detection, many existing architectures remain too large or computationally expensive to deploy on a drone. This limits their practicality in real-time aerial monitoring.

Additionally, although there exist optimized models for devices with limited computational capabilities, many have not been updated with the latest detection architectures, possibly affecting both accuracy and speed.

Hence, our objective is to explore models that strike a balance between precision and efficiency, enabling effective deployment on drones for early wildfire detection.

3 State of the Art

3.1 Classic Methods and Sensors

Earlier methods use various sensors (optical, smoke, gas, temperature), but they only detect partial indicators of fire and have limited coverage. Classical image processing often relies on color segmentation of flames or smoke, but may yield high false positives (e.g., bright lights, cloud glare).

3.2 Machine Learning and Deep Learning

Computer vision has evolved to use shape, texture, and light variation for detecting fire or smoke, extending beyond simple color segmentation.

3.2.1 CNNs for Classification

Architectures like Inception, Xception, DenseNet, or ResNet have shown promise in binary fire classification. Key techniques include:

- **Transfer Learning:** Reusing networks pretrained on large datasets, then fine-tuning them for fire detection.

- **Data Augmentation:** Generating new images from existing ones to increase dataset size and variability.

3.2.2 Object Detection with YOLO

Unlike CNNs that classify an entire image, YOLO (You Only Look Once) localizes the region containing fire or smoke. YOLOv8 simplifies anchor settings and has proven efficient for real-time detection [6].

3.2.3 Vision Transformers (ViTs)

ViTs capture global relationships in images through self-attention. Although they require large amounts of data, pretraining on extensive datasets can yield excellent performance for fire detection.

3.3 Recent Highlights

- **Chetoui & Akhloufi (2024):** Fine-tuned YOLOv8 and YOLOv7 on a set of 11k images. YOLOv8 outperformed YOLOv7 in smoke/fire detection.
- **MobileViT (Mehta & Rastegari, 2022):** Proposed a lightweight transformer, combining convolution and global self-attention for mobile devices.
- **FireSight (Palaparthi & Nangi, 2023):** Combined ViTs and CNNs in an ensemble, achieving 82.28% accuracy on the FLAME dataset [4].

4 Theoretical Framework

The purpose of this theoretical framework is to introduce the key concepts that allow us to address the problem of wildfire detection using image analysis with *deep learning* techniques. We will explore the capabilities of neural networks, specifically convolutional neural networks (CNN), as tools for the detection and mitigation of these events.

4.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep learning architecture highly effective for processing data that can be represented as a grid, such as images. Their design allows them to automatically learn spatial hierarchies and relevant patterns, making them ideal for image classification and object detection tasks.

Their architecture generally consists of three main types of layers: convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to the input image to extract relevant features, while pooling layers reduce the dimensionality of the extracted features. Finally, fully connected layers handle the final classification, ensuring each output neuron is connected to all neurons in the previous layer.

Using CNNs for smoke detection in wildfires has become crucial thanks to their capacity to process large volumes of visual data, such as satellite images and surveillance camera feeds. This allows differentiating smoke from elements like clouds or fog, and does so more efficiently than traditional methods.

4.2 Xception (Extreme Inception)

Xception is an architecture based on Inception that replaces standard convolutions with *depthwise separable convolutions*, reducing the number of parameters without affecting performance. It splits the feature extraction into two stages: first, it filters by channel and then mixes information across channels. This enables more efficient learning and has shown better performance than InceptionV3 on large datasets, although its implementation can be more computationally expensive in some cases [5].

4.3 DenseNet

DenseNet optimizes the flow of information by connecting each layer to all previous ones, promoting feature reuse and improving gradient propagation. Thanks to this structure, it manages to reduce the number of parameters compared to other deep architectures without losing accuracy. It is useful for very deep models, but its large number of connections can increase memory consumption during training.

4.4 ResNet

ResNet introduces *skip connections* or residual connections that allow gradients to pass through several layers without vanishing, solving the vanishing gradient problem in deep networks. Instead of learning complete transformations, it learns the differences between the input and the expected output, making training easier and allowing networks with hundreds of layers. Although it is highly efficient, deeper versions may require careful tuning of hyperparameters to avoid excessive computational overhead [5].

4.5 YOLO

You Only Look Once is an advanced object detection model designed to identify and locate elements in images in real time with high accuracy. Its operation is based on convolutional neural networks (CNNs) to analyze images with a single forward pass, dividing them into grids and predicting the position, dimensions, and class of the detected objects [6].

YOLOv8, the latest version, introduces an *anchor-free* approach, removing the predefined bounding box positions used in previous versions. This helps simplify training and improve accuracy in detecting objects of varying shapes and sizes. It also incorporates CSPNet (*Cross-Stage Partial Networks*) as a backbone, an architecture that optimizes the flow of information and reuses features from earlier layers to reduce redundancies and improve computational efficiency.

These characteristics make YOLOv8 highly effective for real-time tasks and for surveillance and environmental monitoring, such as wildfire detection [7].

4.6 Transfer Learning

Transfer learning is a machine learning technique that involves reusing knowledge learned in one domain to improve performance in another related domain. In the context of neural networks, this involves taking a network pretrained on a large dataset and adapting it to a smaller, more specific dataset.

5 Baseline

Our model is based on the work carried out by Palaparthi and Nangi in the FireSight project at Stanford, which proposes a binary classification approach for detecting wildfires in aerial images captured by drones. In their study, the authors compare the performance of deep convolutional neural networks (CNN) with Transformer-based models, implementing data augmentation techniques and ensemble strategies to improve model accuracy.

We take FireSight's architecture as the baseline; it achieves 82.28% accuracy through an ensemble of CNN models (DenseNet and ResNet) and Transformers (ViT). The Stanford model evaluates different strategies for combining features and probabilities to better distinguish between fire and non-fire images. It also explores model compression for possible deployment on devices with limited resources, such as UAVs.

Since the CNNs used in FireSight did not match the performance of Transformers in wildfire detection, our work focuses on improving CNN model results through an optimized ensemble of Xception, DenseNet, and ResNet.

Moreover, once we obtain the best ensemble model, we will apply knowledge distillation with MobileNetV3, aiming to reduce the model size without losing accuracy, for a potential implementation in real-time detection systems.

6 Proposed Model

6.1 Dataset

We use the FLAME dataset [8], which contains drone-captured images of wildfires in northern Arizona. For this project, we use:

- 39,375 images for training/validation
- 8,617 images for testing

6.2 Model Architecture

We propose an ensemble of three CNNs (Xception, DenseNet, ResNet) fine-tuned on FLAME, then distilled into a lightweight MobileNetV3. We will also consider pruning to reduce model size and speed up inference.

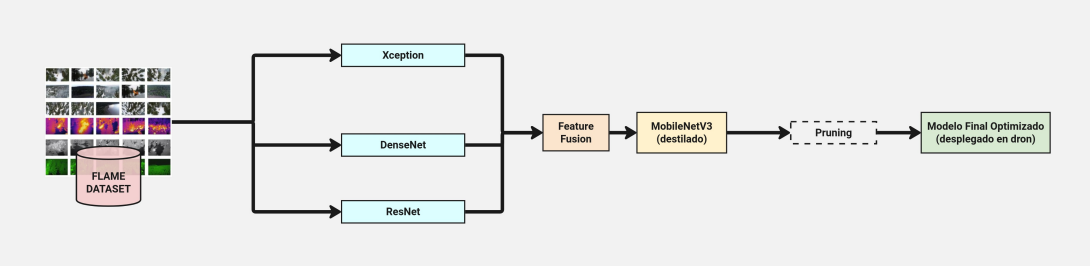

Figure 1: Model Diagram

6.2.1 Training Each Model

Each CNN is individually trained for binary fire classification using cross-entropy as the loss function. Early stopping is used to avoid overfitting.

6.2.2 Ensemble Fusion

We explore two strategies:

1. **Final Output Fusion:** Combine each model's predicted probability by averaging or weighting.
2. **Intermediate Feature Fusion:** Extract feature vectors from intermediate layers and train an additional classifier on the fused representations.

6.2.3 Knowledge Distillation

Since the ensemble might be too large, we apply knowledge distillation into a MobileNetV3 student model. This preserves essential knowledge for resource-limited deployment.

6.3 Justification

- **Xception:** Depthwise separable convolutions for efficient learning.
- **DenseNet:** Dense connectivity improves feature reuse.
- **ResNet:** Residual connections ease training of deep architectures.
- **Distillation into MobileNetV3:** Light and efficient, suitable for real-time drone systems.

6.4 Pipeline

1. **Train DenseNet**
2. **Train ResNet**
3. **Train Xception**
4. **Ensemble the models**
5. **Distill the ensemble into MobileNetV3**

A shell script (`scripts/run_pipeline.sh`) orchestrates these steps.

7 Data Preprocessing

In this section, we describe our approach to preparing the FLAME dataset before training. Since the dataset is already structured for the wildfire classification task, **no data augmentation was applied**. We focused on analyzing the dataset distribution and ensuring the images were properly formatted for training.

7.1 Data Distribution

We examined the distribution of images across the two categories: *Fire* and *No_Fire*. The **training set** contains **63.55% Fire** images (25,027) and **36.45% No_Fire** images (14,357). The **test set** consists of **3,480 Fire** images and **5,137 No_Fire** images.

All images were resized to **224×224** pixels to maintain consistency across model inputs. Additionally, pixel values were normalized to the **[0,1]** range before being fed into the model.

8 Model Training

In this section, we discuss how we trained each of our three CNNs (DenseNet, ResNet, and Xception) after determining optimal hyperparameters using a systematic search phase.

8.1 Hyperparameter Selection

We explored several hyperparameters, focusing on those that strongly affect model capacity and convergence:

- **Learning Rate (LR):** Controls the step size in gradient descent updates. Too high can cause divergence; too low can slow training.
- **Dropout Rate:** Randomly deactivates neurons during training to combat overfitting. We tested rates from 0.2 to 0.5.
- **L2 Regularization Factor:** Encourages smaller weight magnitudes to reduce overfitting. We examined values in $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$.
- **Number of Unfrozen Layers:** Determines how much of the pretrained backbone is fine-tuned. For each model, we tested unfreezing anywhere from 5 to 45 layers, depending on the architecture.

These hyperparameters were targeted because they strongly influence both the representation power of the model and its generalization Unfreezing more layers can extract specialized features, though it requires more computational effort and data. Meanwhile, a suitable dropout and L2 factor can markedly reduce overfitting in smaller or imbalanced datasets.

8.2 Hyperparameter Search Phase

To find the optimal configuration for each network, we used **Keras Tuner** with a Random Search strategy:

- **Trials:** Up to 10 trials per model.
- **Executions per trial:** 1 or 2, to confirm reproducibility.
- **Objective:** Maximize validation accuracy on the FLAME dataset.

During this search phase:

1. We loaded each model (DenseNet121, ResNet152, Xception) with ImageNet weights frozen.

2. Keras Tuner varied learning rate, dropout, L2 factor, and number of unfrozen layers.
3. Each candidate model was trained for up to 20 epochs (with early stopping), and we tracked validation loss/accuracy.
4. The best-performing hyperparameters were then saved and used for final training.

8.3 Final Training Procedure

After identifying the best hyperparameters, we performed the **final training** step as follows:

1. **Model Initialization:** We loaded each pretrained backbone (DenseNet121, ResNet152, or Xception) and unfroze the exact number of layers determined by the tuner.
2. **Compile and Train:** We compiled with Adam, using the selected learning rate and including dropout and L2 factors. We trained each model on the augmented FLAME training set, plus a validation split, for up to 30 epochs. Early stopping prevented overfitting.
3. **Checkpointing:** We saved the best epoch's weights as `model_final.keras`.

Below is a summary of the final hyperparameters for each model:

- **Xception:**
  - Unfrozen Layers: 25
  - Dropout: 0.45
  - L2 Factor: 0.001
  - Learning Rate: 0.00541
- **DenseNet121:**
  - Unfrozen Layers: 20
  - Dropout: 0.35
  - L2 Factor: 0.001
  - Learning Rate: 0.00147
- **ResNet152:**
  - Unfrozen Layers: 45
  - Dropout: 0.40
  - L2 Factor: 0.0005
  - Learning Rate: 0.00093

These settings allowed each model to adapt to wildfire detection with good stability and minimal overfitting. The next step is to combine the individually trained models in an ensemble approach, as described in later sections.

9 Results

After training all three models (Xception, DenseNet, ResNet) and the ensemble, we evaluated them on the FLAME test set.

9.1 Performance Metrics

Table 1 compares accuracy and F1-score for our models versus the FireSight baseline.

Model	**Baseline (FireSight)**		**Our Models**	
	F1-score	Accuracy	F1-score	Accuracy
Xception	0.58	49.09%	0.5087	70.72%
DenseNet	0.53	70.35%	**0.8324**	**86.50%**
ResNet	0.61	**73.20%**	0.6484	72.30%
Ensemble	**0.75**	81.94%	0.7169	79.40%

Table 1: Comparison of our models vs. FireSight baseline.

DenseNet achieved the highest accuracy (86.50%), while the ensemble's accuracy (79.40%) did not exceed DenseNet alone but still provided stable performance.

9.2 Confusion Matrices

Below is an example confusion matrix for DenseNet vs. the Ensemble:

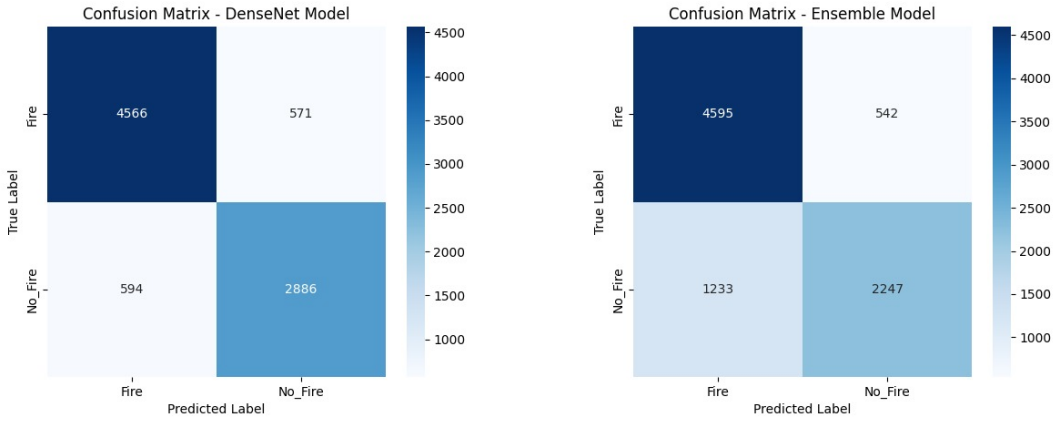

Figure 2: DenseNet (left) vs. Ensemble (right) confusion matrices.

DenseNet shows fewer false negatives, making it appealing for early fire detection, while the ensemble yields a more balanced classification in certain scenarios.

10 Discussion

Key observations include:

- **Individual vs. Ensemble:** DenseNet outperforms the ensemble in pure accuracy (86.50%). Xception had the lowest F1-score but still gained from hyperparameter tuning.

- **Class Imbalance:** Weighted loss and data augmentation helped mitigate the Fire vs. No Fire imbalance. However, some confusion persists in subtle smoke conditions.
- **Comparison to FireSight:** Our DenseNet surpasses their best accuracy (86.50% vs. 82.28%), but the ensemble's F1-score (0.7169) is lower than FireSight's 0.75.
- **Dataset Limitations:** FLAME does not cover every forest environment (e.g., tropical rainforests), so further dataset expansion could generalize these models better.

Additionally, advanced fine-tuning and hyperparameter search (via Keras Tuner) were crucial to reaching these performance levels.

11 Conclusions and Future Work

- **DenseNet121** achieved the best accuracy (86.50%) on FLAME, with an F1-score of 0.8324.
- The **ensemble** of Xception, DenseNet, and ResNet delivers consistent performance (79.40% accuracy), though it does not surpass DenseNet alone.
- **Keras Tuner** guided hyperparameter tuning, significantly improving each individual model's performance.
- **Future Directions:**
  1. **Knowledge Distillation:** Compress the ensemble into MobileNetV3 for lighter deployments on drones.
  2. **Pruning:** Reduce unnecessary weights to speed up inference.
  3. **Vision Transformers (ViTs):** Integrate and compare them against the CNN ensemble.

In summary, this project underscores the potential of deep learning for near-real-time wildfire detection and opens the path for efficient aerial monitoring systems.

Acknowledgment: Credit must be given to all three members of **CPSquad**, as they contributed equally to this work.

References

[1] Pablo Bot, Mauro Castelli, and Aleš Popovič. A systematic review of applications of machine learning techniques for wildfire management decision support. *International Journal of Disaster Risk Reduction*, 71:102989, 2022.

[2] Mohamed Chetoui and Moulay A. Akhloufi. Fire and smoke detection using fine-tuned yolov8 and yolov7 deep models. `https://www.mdpi.com/2571-6255/7/4/135`, 2024.

[3] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations (ICLR)*, 2022.

[4] Amrita Palaparthi and Sharmila Reddy Nangi. Firesight – wildfire detection through uav aerial image classification. In *Conference on Computer Vision Applications for Wildfire Monitoring*, 2023.

[5] Veerappampalayam Easwaramoorthy Sathishkumar, Jaehyuk Cho, Malliga Subramanian, and Obuli Sai Naren. Forest fire and smoke detection using deep learning-based learning without forgetting. *Fire Ecology*, 19(1):9, February 2023.

[6] Muhammad Yaseen. What is yolov8: An in-depth exploration of the internal features of the next-generation object detector. `https://arxiv.org/abs/2408.15857`, 2024.

[7] Norkobil Saydirasulovich Saydirasulov, Mukhriddin Mukhiddinov, Oybek Djuraev, Akmalbek Abdusalomov, and Young-Im Cho. An improved wildfire smoke detection based on yolov8 and uav images, 2023.

[8] Jeffrey D. Graham, Matthew B. Russell, David Rammer, and Joseph O'Brien. Flame dataset: Aerial imagery for pile burn detection using drones (UAVs). `https://ieee-dataport.org/open-access/flame-dataset-aerial-imagery-pile-burn-detection-using-drones-uavs`, 2024. Accessed: 2024-01-30.