Wildfire Detection with a CNN Ensemble

1INF52 Deep Learning

CPSquad

PUCP

March 4, 2025

Contents

Introduction and Problem Statement

- The increasing frequency of wildfires has led to a demand for **automated monitoring systems**.
- Traditional methods (satellites, thermal sensors) suffer from **delayed data retrieval**.
- **Deep learning** techniques can be used to **improve detection speed and accuracy**.

Figure 1: Wildfire in Peru. Source: *Daily Sabah*

Objective of this Study

- Develop an **ensemble of CNNs** for **wildfire detection** in aerial images.
- Evaluate the performance of **Xception, DenseNet121, and ResNet152**.
- Improve classification accuracy while keeping computational efficiency.

Traditional Approaches to Wildfire Detection

- **Sensor-based methods:** Use **temperature, smoke, and gas sensors**, but have **limited coverage**.
- **Classic Computer Vision methods:** Use **color-based segmentation**, but suffer from **high false positives**.
- **Machine Learning and Deep Learning approaches:**
  - CNN-based classification (e.g., Xception, DenseNet, ResNet).
  - Object detection with **YOLOv8**.
  - Vision Transformers (**ViTs**) for feature extraction.

Baseline Model: FireSight (Stanford)

- FireSight combines **CNNs and ViTs** for aerial wildfire detection.
- Achieves **82.28% accuracy** and an **F1-score of 0.75** using a **DenseNet + ResNet + ViT** ensemble.
- Our approach:
  - Perform **Transfer Learning** on **Xception, DenseNet, and ResNet** using pre-trained weights.
  - Improve each model's performance by **unfreezing and training the last layers** to refine feature extraction.
  - Ensemble the fine-tuned models using a **voting-based strategy** to compare against the SOTA baseline.

FLAME Dataset Overview

- The dataset contains **drone-captured** images of wildfires.
- Designed specifically for **binary wildfire classification** (*Fire* vs *No Fire*).
- **Training Set:** 39,375 images
- **Test Set:** 8,617 images.
- **Validation Split:** 10% of training set.
- **Class Distribution:**
  - Fire: **25,027 images (63.55%)**.
  - No-Fire: **14,357 images (36.45%)**.

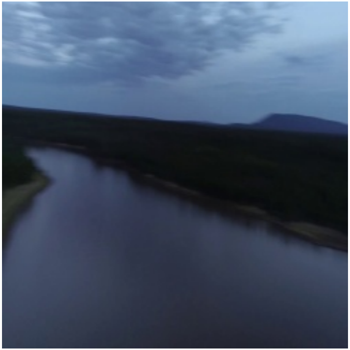


Figure 2: Sample FLAME dataset image.

Pipeline for Model Training

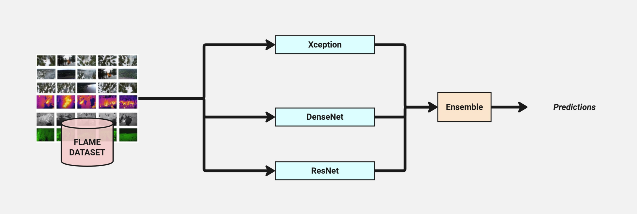


Figure 3: Model Architecture

- **Preprocessing:** Images resized to **224x224** and augmented.
- **Training Strategy:** Individual CNNs trained separately.
- **Evaluation:** Models compared based on **accuracy**, precision, recall, and **F1-score**.

Keras Tuner for Hyperparameter Search

- **Keras Tuner** used for optimizing **unfrozen layers, dropout, L2 regularization, and learning rate**.
- Best hyperparameters found:

Model	U. layers	Dropout	L2 Factor	L. Rate
Xception	25	0.45	0.001	0.00541
DenseNet121	20	0.35	0.001	0.00147
ResNet152	45	0.4	0.0005	0.00093

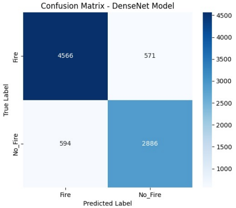
Why Use an Ensemble?

- **Goal:** Improve model **stability and accuracy**.
- We experimented with:
  - Majority Voting (Final Selection).
  - Weighted Averaging.
  - Stacking.

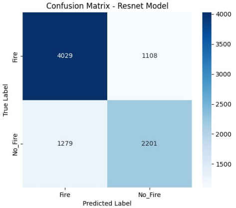
Final Approach: Voting

- Each model votes on the predicted class.
- The most frequent class is the **final prediction**.

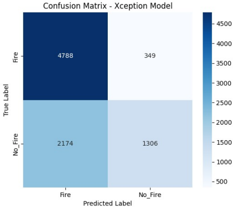
Confusion Matrices



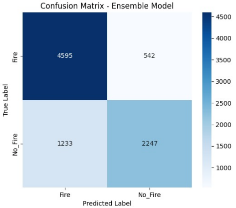
DenseNet



ResNet



Xception



Ensemble

Final Model Performance

Model	**Baseline**		**Our Models**	
	F1-score	Accuracy	F1-score	Accuracy
Xception	**0.58**	49.09%	0.5087	**70.72%**
DenseNet	0.53	70.35%	**0.8324**	**86.5%**
ResNet	0.61	**73.2%**	**0.6484**	72.3%
Ensemble (Voting)	**0.75**	**81.94%**	0.7169	79.4%

Key Takeaways

- **DenseNet121 and Xception** models outperformed the ensemble in some metrics.
- The ensemble's advantage might be due to the use of **Vision Transformers (ViTs)**.
- **Close-to-SOTA results** may be due to this dataset not being used at scale with recent models.
- **Keras Tuner** was crucial for finding the best hyperparameters, improving individual models.
- **Future Work:**
  - **Pruning** to reduce model size and improve inference speed.
  - **Distillation** to compress into a lighter architecture like MobileNetV3.

Thank You!

Our code is available at:

`github.com/Litusuwu/DeepLearning_Project`

Trained models can be found at:

`huggingface.co/superflash41/fire-chad-detector-v1.0`