Wildfire Detection

An Iterative Deep Learning Journey

Modern AI • Group 8 • Jack Meghreblian, Vernon Walker, & Brody Kasprzak

Problem

Binary classification: Fire vs. No-Fire in wildfire imagery

Goal: Adapt general-purpose AI for specialized wildfire detection.

The Problem and Its Importance:

- **Life-saving:** Early detection can prevent massive destruction of life, property, and natural resources.
- **Environmental Protection:** Minimizing the spread of wildfires preserves ecosystems and reduces air pollution.
- **Resource Allocation:** Accurate detection helps in efficient deployment of firefighting resources.
- **Economic Impact:** Reduces financial losses from property damage, forest destruction, and related economic disruptions.

Data Source and Preprocessing

Data Source: Wildfire dataset from Kaggle
https://www.kaggle.com/datasets/elmadafri/the-wildfire-dataset

- Came split into train, test and val image folders

2,681

Total Images

- Shrunk images to 224x224 pixels
- Normalized pixel values for each image
- Data size reduction from 10GB to 30MB for faster image processing
- Image size reduction augmented the input data for less overfitting

Model Overview

Dataset

- 1,878 training images
- 398 validation images
- 405 test images
- 2 classes: fire / nofire
- ~60/40 class imbalance
- 224×224 px, normalized

Evolution of Models

1	2	3	4	5
EfficientNetB0 (Frozen)	+ Data Augmentation	ResNet50V2 + Class Weights	Partial Fine-Tuning	ResNet50V2 + Hard Example Mining
62% stuck	Still 62%	81%	88–89%	

Core Challenge

Fire and smoke are amorphous — no fixed edges or shapes. Standard object-detection backbones struggle because they were trained on crisp, bounded objects. Class imbalance (60/40) further pushed early models to predict 'no fire' for everything.

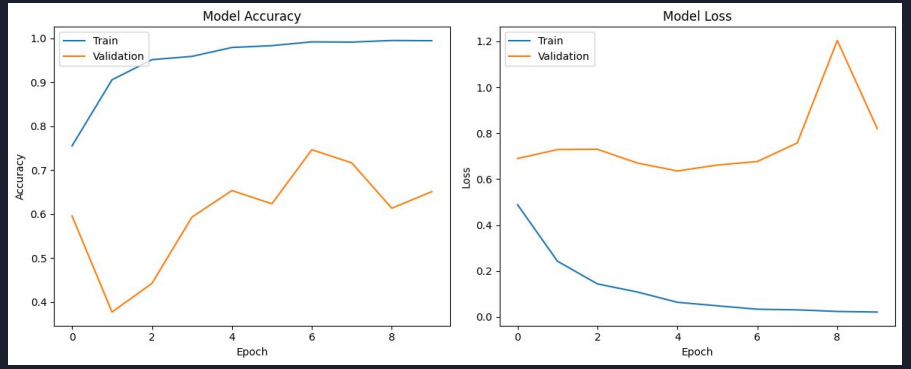
Phase 1: EfficientNetB0 — Baseline

What We Built

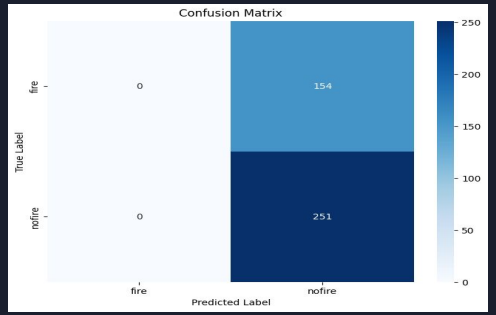
- Pre-trained EfficientNetB0 backbone (ImageNet weights)
- Base layers frozen — only custom head trained
- GlobalAveragePooling → Dropout(0.2) → Dense(2, softmax)
- 4M total params, only 2,562 trainable
- Adam optimizer, lr=0.001, 10 epochs

Result: 62% Accuracy — Model only predicted 'No Fire'

Training Curves



Confusion Matrix



62%
Test Accuracy

0.53
AUC-ROC

Data Augmentation: Teaching Invariance

Same image — 9 random augmentations applied during training

Why Data Augmentation?

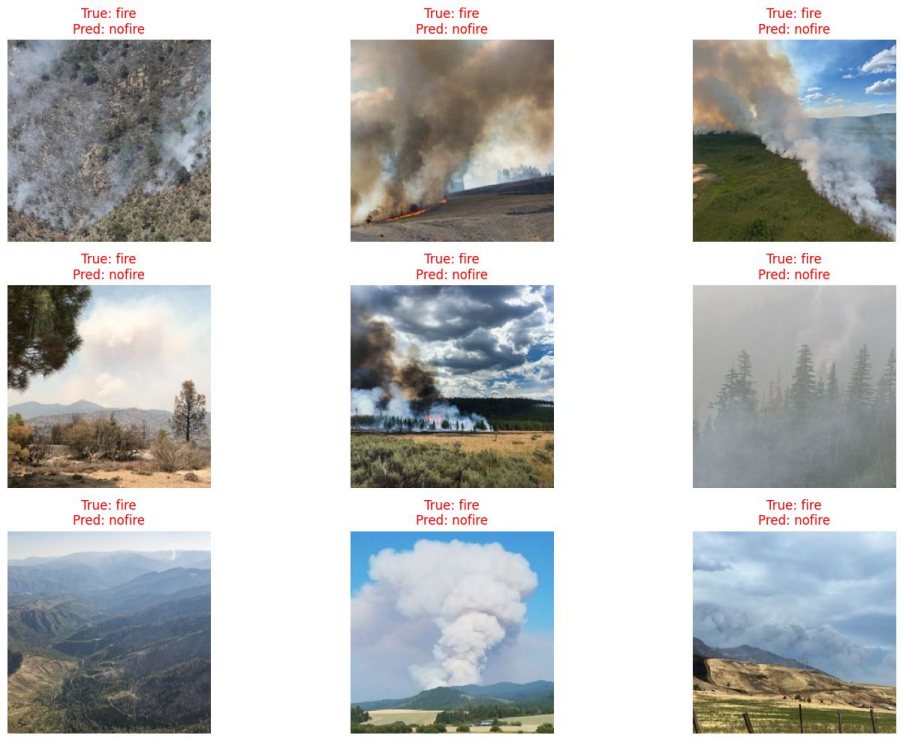
Wildfire images vary enormously: different angles, distances, smoke density, and color temperature. Without augmentation, the model memorizes training images rather than learning generalized 'fire features'.

Transforms Applied:

Random Flip (H+V) • Random Rotation ±20° • Random Zoom ±20%

Still stuck at 62% — the backbone was the real bottleneck

Why EfficientNetB0 Failed



Model predicted 'No Fire' for ALL 154 fire images

Feature Incompatibility

EfficientNetB0 was designed to detect objects with clear, fixed edges (dogs, cars). Fire and smoke have no stable shape, they're amorphous, constantly shifting.

Class Imbalance Trap

60% of images are 'no fire'. Predicting 'no fire' every time achieves 62% accuracy, a local minimum the frozen model happily stayed in.

Frozen Features

With the backbone frozen, only 2,562 parameters could update. These couldn't learn fire's texture-based features from ImageNet edge-detection filters.

→ **Next step: Switch backbone + fix the imbalance problem**

The Pivot: ResNet50V2 + Class Weights

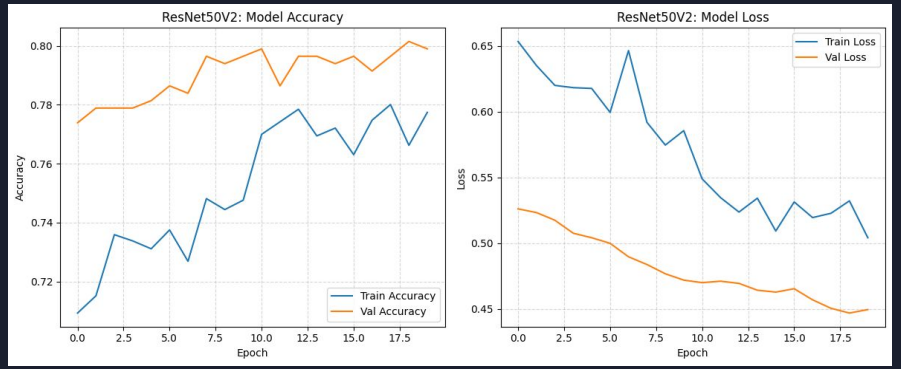
Why Switch to ResNet50V2?

- Residual connections preserve gradient flow across 50+ layers
- Pre-Activation design is better for texture/pattern features
- More robust to the 'local minima' trap from class imbalance

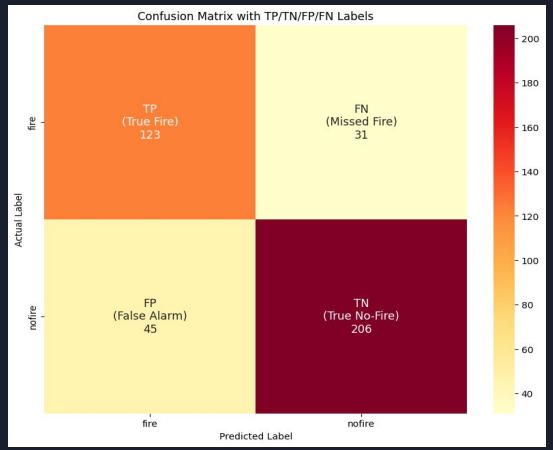
Class Weighting to Fix Imbalance

Fire class weight: 1.30 • No-Fire class weight: 0.81

Misclassifying a fire image is penalized 1.6× more than a no-fire mistake, forcing the model to learn fire features.



Training & Validation Accuracy / Loss — ResNet50V2



81%
Accuracy

73%
Fire Precision

80%
Fire Recall

Phase 4: Partial Fine-Tuning the ResNet Backbone

What Changed

Unfroze top 20 layers of ResNet backbone

Let high-level features adapt to wildfire imagery while keeping low-level edge detectors intact.

Tiny learning rate: 1e-5

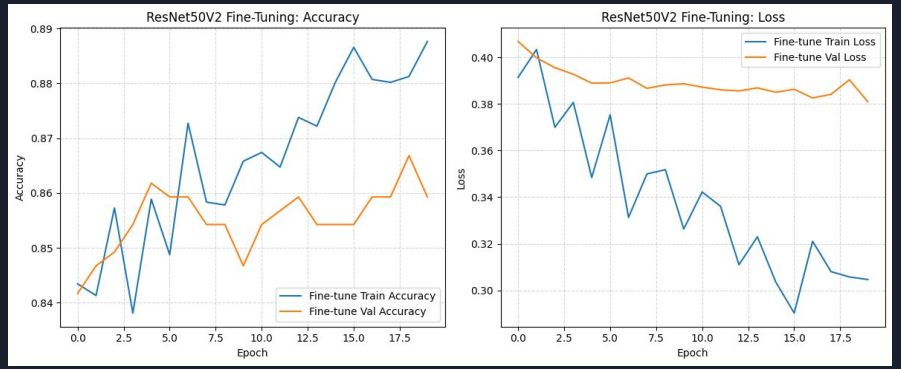
Fine-tuning requires precision — too large a LR destroys pre-trained knowledge.

Continued class weighting

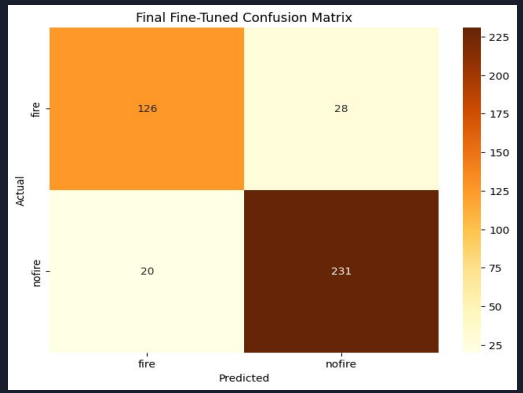
Maintained fire-class penalty to prevent regression toward majority-class bias.

Early Stopping (patience=4)

Prevents overfitting by restoring best weights when validation loss plateaus.



Fine-tuning accuracy/loss — both train and val improve together (healthy!)



Fine-tuned confusion matrix

88%
Test Accuracy

86%
Fire F1-Score

82%
Fire Recall

Threshold Optimization

Why Threshold Matters for Safety

In wildfire detection, a False Negative (missed fire) is catastrophic, delayed response can cost lives and thousands of acres.

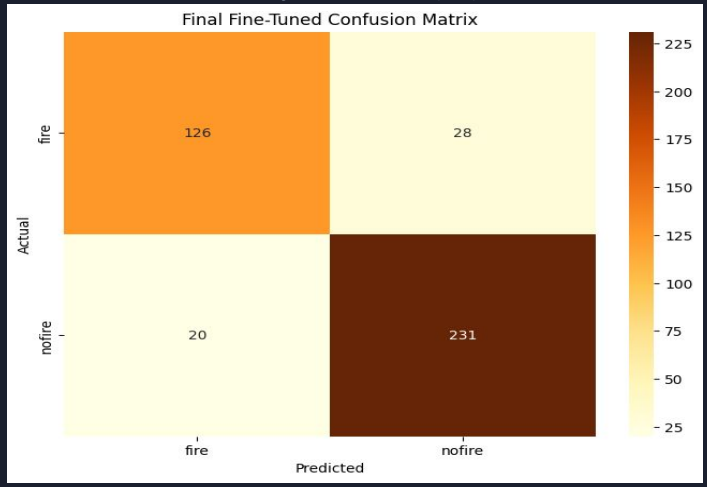
A False Positive (false alarm) is inconvenient but safe.

Lowering the decision threshold from 0.5 ⟶ 0.3 makes the model more sensitive to fire, accepting more false alarms to catch more real fires.

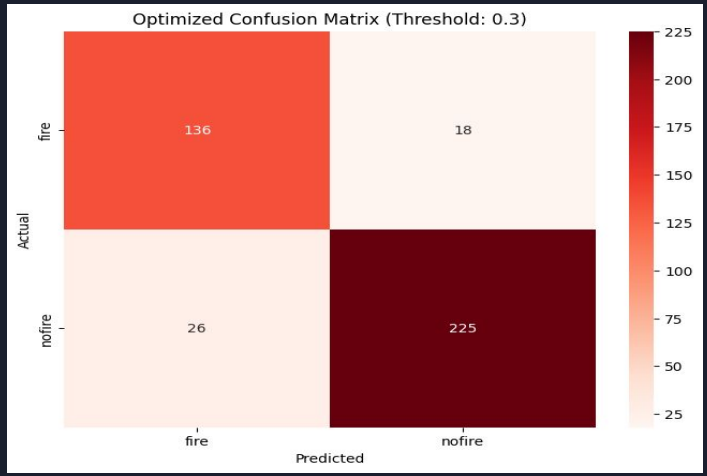
Threshold Comparison

Metric	Default (0.5)	Optimized (0.3)
Accuracy	88%	89%
Fire Recall	82%	88%
Missed Fires	28	18
False Alarms	21	30

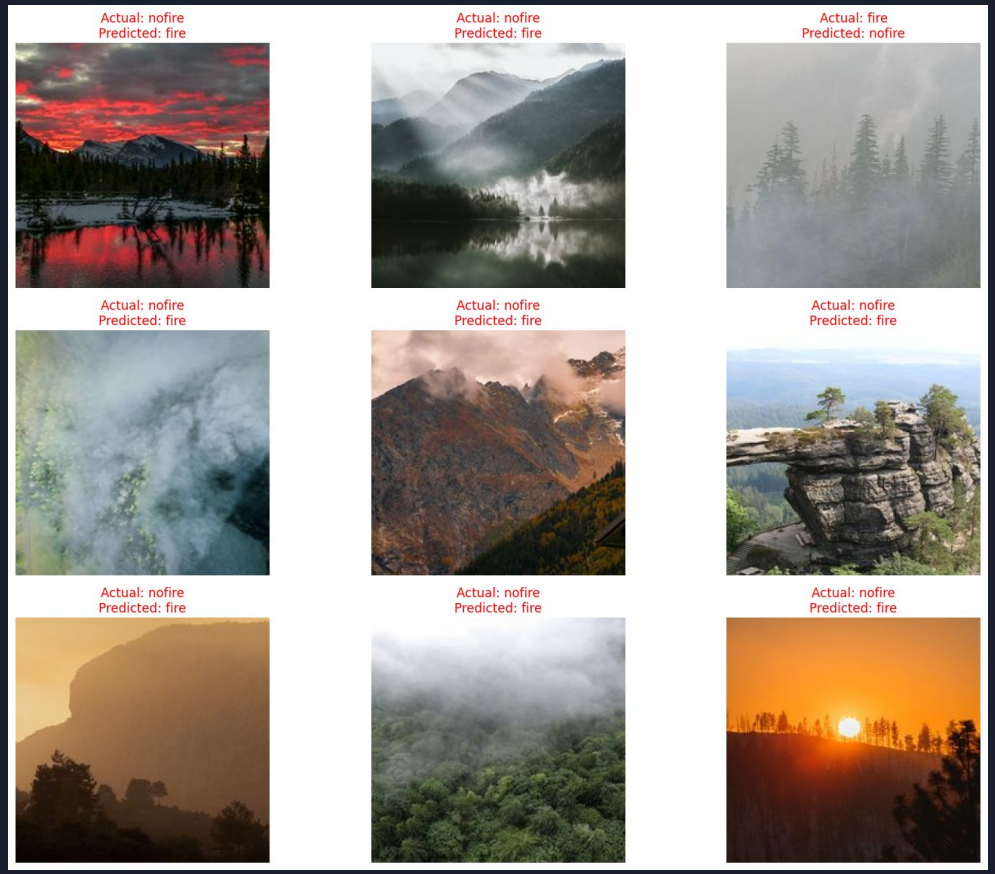
Default Threshold (0.5)



Optimized Threshold (0.3)



What the Model Still Gets Wrong



Insight: as accuracy improves, remaining errors become genuinely hard

Smoke & Cloud Confusion

Sunset / Bright Sky

Dust & Haze

Early/Faint Fire

Hard Example Mining

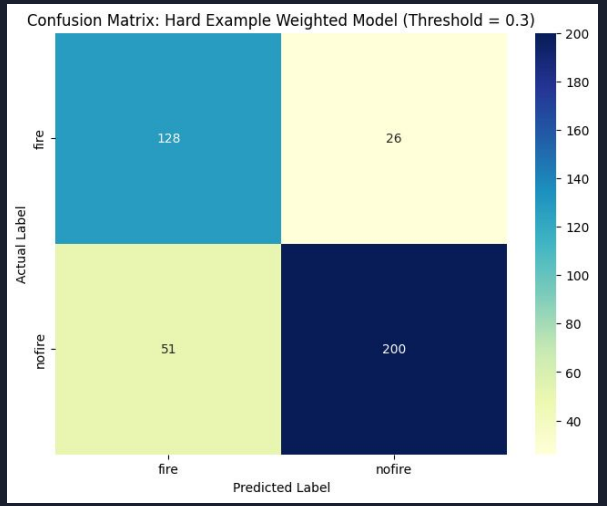
What it is:
A training strategy that helps the model learn from difficult images that look like fire or smoke, such as clouds or haze.

Why we used it:
To reduce false alarms by forcing the model to better distinguish real fire/smoke from visually confusing non-fire scenes.

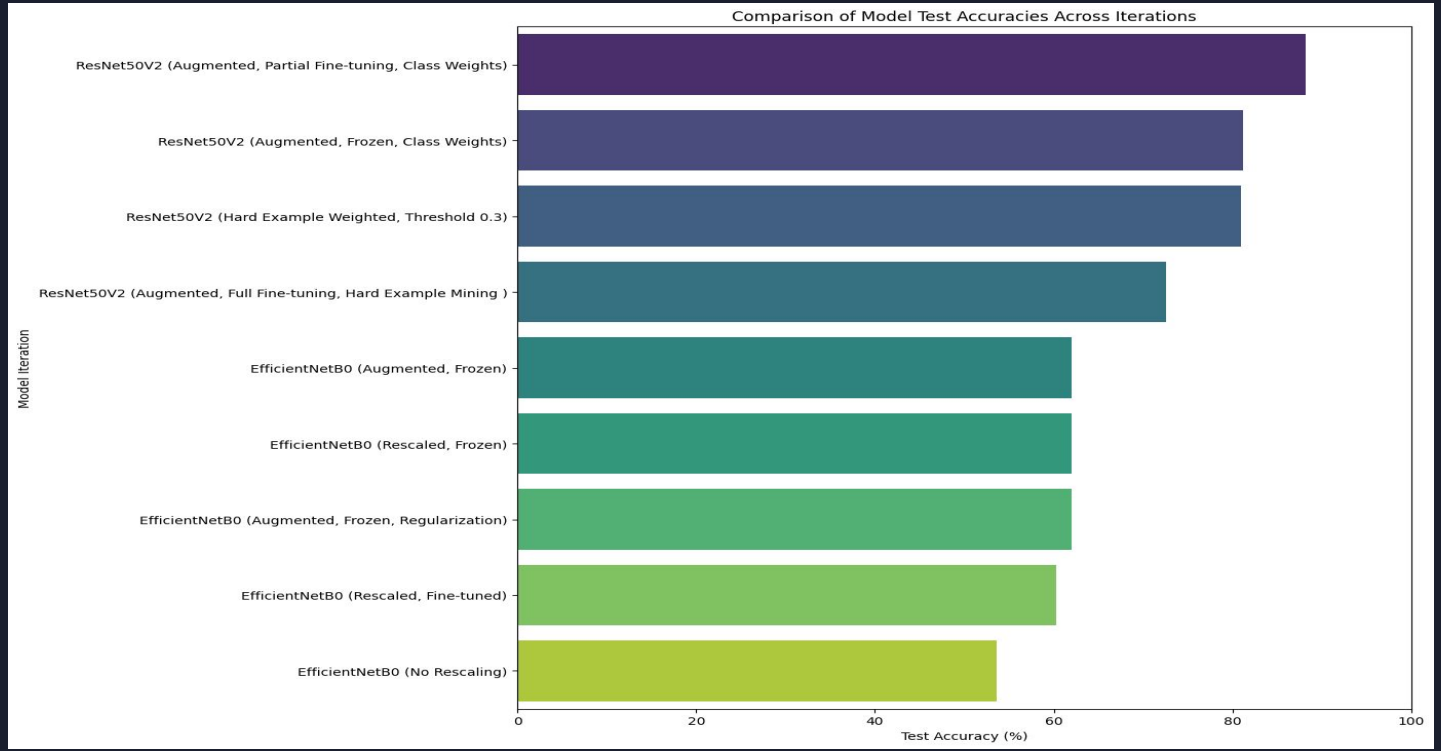
Procedure:

1. Train a baseline model.
2. Run predictions on the training set.
3. Identify No-Fire images incorrectly predicted as Fire
4. Assign these hard examples higher training weights (5x).
5. Retrain the model using the weighted samples.

Impact:
This makes the model focus more on confusing cases, improving robustness and lowering false positives.



Final Results & Key Takeaways



Key Lessons

Architecture choice > hypertuning • Class balance is important

Improvements and Limitations

More Training Data

The biggest improvement likely comes from a larger, more diverse image set, since wildfire scenes vary widely across smoke, lighting, terrain, and weather.

Hard Example Mining

Continue targeting difficult errors, especially clouds or haze mistaken for smoke, and consider adding a dedicated cloud class.

Multi-Spectral Inputs

RGB images are limited. Adding other sensor data, such as thermal or infrared, could improve detection.

Temporal and spatial Modeling

The model classifies single images in isolation, missing two major advantages of real wildfire systems: spatial context (where smoke appears relative to terrain or horizon) and temporal context (how smoke grows and moves across frames compared with clouds).

Beyond Binary Classification

Fire vs. No-Fire is too simple for real deployment. Future models should produce richer outputs such as smoke, flame, post-fire, or low-confidence smoke.

Thank you!