Group 73 Progress Report: Wildfire Detection Classification Plan

Andy Huynh, Berk Yilmaz, Tanisha Tasnin
huynha3@mcmaster.ca, yilmag1@mcmaster.ca, tasnint@mcmaster.ca

1 Introduction

Wildfires are becoming increasingly prevalent and devastating due to climate change [1]. This poses a growing threat to both human safety and the environment, causing billions of dollars in damages each year and severely affecting air quality and biodiversity [5], [10], [9]. Early detection of wildfires is crucial for effective response and mitigation efforts. However, detecting wildfires at an early stage can be challenging. Subtle indicators such as light smoke or small flames can be obscured by dense vegetation, clouds, or varying terrain, often making traditional monitoring methods unreliable [14]. In an attempt to address this issue, our project explores the use of Convolutional Neural Networks (CNNs) for automatic wildfire detection from images [13], [11]. Our project formulates the task as a binary classification problem: given an RGB image of an area, the model identifies it as a "fire" or "no fire" scenario using 'The wildfire dataset' by El Madafri on kaggle [6].

2 Related Work

Advancements in wildfire research have led to the use of deep learning to enhance prediction, monitoring, and detection capabilities [11]. There are several existing solutions that use a combination of various CNN architectures and other neural networks for wildfire prediction. Popular models include **FirePred** and **WFNet**. *FirePred* is a hybrid multi-temporal CNN model for wildfire spread prediction and *WFNet* is a hierarchicalCNN for wildfire spread prediction [8], [7]. However these models, like most renowned ones, focus on modeling wildfire spread and spatiotemporal dynamics rather than direct detection from visual data which our project addresses. For direct wildfire detection from images, the paper **Advanced Wildfire Detection Using Deep Learning Algorithms: A Comparative Study of CNN Variants** is a notable example [3]. This study evaluates models such as InceptionV3, Xception, and NASNetMobile on over 25,000 images, achieving accuracies above 98%. While this study focuses on benchmarking CNN architectures for accuracy, our project differs by optimizing image preprocessing and augmentation pipelines to balance accuracy with computational efficiency for real-time detection. There are however other CNN projects such as Malaria Detection using TensorFlow's malaria dataset that resemble our project's workflow more closely [12], [4]. Similar to our approach, they emphasize image preprocessing, model training, and performance optimization for efficient binary classification.

3 Dataset

We are training our model with the splits gathered from Kaggle. We have 1887 data points for training, 402 for validation, and 410 for test. This roughly corresponds to a 70-15-15 split of our dataset.

The dataset is composed of images mainly depicting forests, fields, and rural areas. Images are sorted into two categories; "fire" and "nofire". These categories constitute the labels we use during training.

The images are of different resolutions and quality. Some images are considerably higher resolution than others, and it is apparent that the images are from different time periods and were taken by different devices. This presents a minor challenge in training our model, where we need to ensure our model can use each and every one of these data points. We take this by normalizing our data as outlined in the Features section.

4 Features

The feature set for our model comprises of the pixel values from the images themselves. The images are mainly RGB type images meaning they compose of three colour channels red, green, and blue. The resolution also dictates the number of pixels within a given image. For example, a 128x128 resolution image means there are 128 * 128 * 3 = 49152 feature values that corresponds to the RGB colour values. The higher the resolution more features there are and vice versa. During preprocesssing, the images are compressed to a fixed resolution (128x128, 224x224, etc...), so that the input array remains the same. The pixel values are placed in 3D-dimensional matrix. This form of feature engineering alows for spacial recognition and patten extraction on a 3D space. These values are also normalized for efficiency and robustness. Different resolutions sizes were experimented on to see if the increase in pixel sizes (and feature length) would increase the accuracy or validity of the model; ranging from 128x128, 224x224, 299x299, and 1000x1000. The following augmentations were used to vary the pixel locations so that the model becomes invariant to orientation, position and scale; rotation range rotates the pixels from a random angle, width shift range shift the images left and right, height shift range shifts the images up and down, zoom range magnifies the image, horizontal flip will mirror the image.

5 Implementation

Our implementation is a feedforward classification model. It is composed of TODO Conv2D layers of size TODO with ReLu activation, followed by a Dense layer with TODO units, and . The model has TODO parameters in total, taking up about TODO MB in total. Before training, we optimized various parameters empirically. We set out certain augmentations and resolutions to be tested, then trained a model on every dimension of this space via nested for loops that can be found in our code. This process is further outlined in the Features section. The size of layers and the number of layers were determined by manual empirical testing.

To find the most optimized configuration for the classification model, the program checks for differnet resolution and augmentations combinations for the best performance. The program performs short epoch intervals session on the training data for each combination to identify which combination outputs the best validation accuracy.

The program utilizes a binary cross entropy as a loss function. This is good for binary classification ,because it integrates nicely with the sigmoid activation function that outputs 0-1 probabililty values and penalizes confident predictions. It also provides a smooth gradient for backprogration and learning [2].

Preliminary issues occured when implementing the code firsthand such as extremely long preprocessing times, and code the running indefinitely. The long prepocessing was fixed by increase batch sizes and simplifying the model. The infinite long run times was caused by a bug from earlier Tensflow version, so the code was updated accordingly.

6 Results and Evaluation

Our model evaluation process focuses on comparing different image preprocessing and augmentation configurations to identify the combination that offers the best trade-off between computational efficiency and classification accuracy. The array of pixel sizes we are testing includes pixelsizes TODO Each configuration was trained for five epochs, and both validation accuracy and average computation time per epoch were recorded to measure performance. The system automatically stops iterating over new resolutions when accuracy improvements fall below 3%, reducing unnecessary computation.

Initial experiments at lower resolutions (128x128) with standard augmentation (rotation, brightness, and zoom variations) achieved validation accuracies around 57 - 58%.

Larger image resolutions are currently being evaluated to determine whether higher spatial detail leads to significant accuracy gains. The best configuration identified so far will be used to train the final CNN model, which is then validated and tested on separate data subsets.

The final architecture employs a four-block convolutional neural network (Conv2D-MaxPooling layers) with dropout for regularization, compiled with binary cross-entropy loss, a metric of accuracy, and the Adam optimizer. Model performance is monitored across epochs using accuracy and loss curves saved as visual outputs. Test performance will be reported once all preprocessing configurations finish executing. For baselines, the model without augmentation and at lower resolutions serves as the control, while subsequent tests compare the impact of increasing image size and data augmentation strength.

Our current best model uses no augmentation and normalizes images to 224x224. This model achieves a training accuracy of 83.73% and a validation accuracy of 79.85%. However, this model is an early experiment and many areas of improvement were raised during the experimentation process. as outlined in the Feedback and Plans section.

To evaluate model performance, the following classification metrics are computed on the test set:

Metric	Fire	No Fire	Weighted Avg.	Support
Precision	xxx	xxx	xxx	xxx
Recall	xxx	xxx	xxx	xxx
F1-Score	xxx	xxx	xxx	xxx
Accuracy			xxx	

Table 1: Model performance metrics on the test set (placeholders to be updated after full training).

7 Feedback and Plans

The TA's feedback centered on four key areas: ensuring reproducibility across experiments, clearly defining baselines for comparison, tracking computational efficiency and overfitting control.

To ensure reproducibility, all preprocessing and augmentation experiments are handled by a single function. The function evaluates each configuration using the same dataset structure and parameters. This ensures consistent conditions across runs. Future improvements will include fixing random seeds for TensorFlow, NumPy, and data generators to make results fully repeatable.

For baseline definition, the non-augmented, low-resolution model (128×128 with no transformations) is used as the baseline mostly. Subsequent augmented models are quantitatively compared against this using validation accuracy and loss.

To address computational efficiency, the experiment function tracks the average training time per epoch for each configuration, comparing between accuracy gains and processing cost. Further refinement will involve logging this data to CSV and plotting runtime versus resolution to visualize performance scalability.

The current implementation already includes an outer-loop early stopping criterion, halting new preprocessing experiments when accuracy improvements fall below 3%. However, an in-training early stopping callback will be added in future iterations to prevent overfitting during prolonged training runs.

As an optional addition, the TA provided suggested additional models other than CNN to compare with in order to explore benefits and trade-offs between each learning model.

Team Contributions

Andy Huynh contributed by writing the Features and Implementations sections (4 & 5) of the report. He provide some early implementation for the code, as well look over made additions to sections 1,2,3,6, and 7.

References

[1] National Aeronautics and Space Administration. 2025. Wildfires and climate change. Web page. Available at https://science.nasa.gov/earth/explore/wildfires-and-climate-change/. Accessed: 2025-11-09.

[2] Arize AI. 2023. Binary cross entropy: Where to use log loss in model monitoring. https://arize.com/blog-course/binary-cross-entropy-log-loss/. Published January 30 2023.

[3] Abhila O. Anju, M. Jayasree, S. Yashica, K. S. Vishali, M. Yuvaraj, and K. Suresh Babu. 2025. Advanced wildfire detection using deep learning algorithms: A comparative study of cnn variants. *International Research Journal on Advanced Science Hub (IRJASH)*, 7(02):79–86.

[4] Bnsreenu. 2023. Malaria binary classification using tensorflow lite. https://github.com/bnsreenu/python_for_microscopists/tree/master/237_tflite_using_malaria_binary_classification. GitHub repository, accessed on 2025-11-10.

[5] Health Canada. 2024. Human health effects of wildfire smoke — report summary. Web document. Available at https://www.canada.ca/en/services/health/healthy-living/environment/air-quality/wildfire-smoke/human-health-effects-report-summary.html. Accessed: 2025-11-09.

[6] I. El-Madafri, M. Peña, and N. Olmedo-Torre. 2023. The wildfire dataset – enhancing deep learning-based forest fire detection with a diverse evolving open-source dataset. https://www.kaggle.com/datasets/elmadafri/the-wildfire-dataset?resource=download. Kaggle dataset, accessed on 2025-11-10.

[7] Wenyu Jiang, Yuming Qiao, Guofeng Su, Xin Li, Qingxiang Meng, Hongying Wu, Wei Quan, Jing Wang, and Fei Wang. 2023. Wfnet: A hierarchical convolutional neural network for wildfire spread prediction. *Environmental Modelling & Software*, 170:105841.

[8] Mohammad Marjani, Seyed Ali Ahmadi, and Masoud Mahdianpari. 2023. Firepred: A hybrid multi-temporal convolutional neural network model for wildfire spread prediction. *Ecological Informatics*, 78:102282.

[9] NBC News. 2025. Billion-dollar disasters: The economic toll of wildfires, severe storms and earthquakes is soaring. Web document. Available at https://www.nbcnews.com/science/climate-change/billion-dollar-disasters-economic-toll-wildfires-severe-storms-earthqu-rcna221473.

[10] Laxita Soontha and Mohammad Younus Bhat. 2026. Global firestorm: Igniting insights on environmental and socio-economic impacts for future research. *Environmental Development*, 57. Accessed: 2025-11-10.

[11] Dario Spiller, Andrea Carbone, Stefania Amici, Kathiravan Thangavel, Roberto Sabatini, and Giovanni Laneve. 2023. Wildfire detection using convolutional neural networks and prisma hyperspectral imagery: A spatial-spectral analysis. *Remote Sensing*, 15(19):4855.

[12] TensorFlow Datasets. 2023. Malaria dataset. https://www.tensorflow.org/datasets/catalog/malaria. Contains 27,558 thin blood-smear cell images with parasitized/uninfected labels.

[13] Eleni Tsalera, Andreas Papadakis, Ioannis Voyiatzis, and Maria Samarakou. 2023. Cnn-based, contextualized, real-time fire detection in computational resource-constrained environments. *Energy Reports*, 9:247–257.

[14] Berk Özel, Muhammad S. Alam, and Muhammad U. Khan. 2024. Review of modern forest fire detection techniques: Innovations in image processing and deep learning. *Information*, 15(9):538.