DATASCI 207 Applied Machine Learning Final Project, Spring 2024

Wildfire Risk Prediction with Remote Sensing and Transfer Learning

Derek Yao[1], Nora Povejsil[1], and Marlon Fu[1]

[1]School of Information, University of California—Berkeley

April 16, 2024

Abstract

Wildfires pose significant threats to ecosystems and communities worldwide, necessitating proactive measures for risk assessment and management. In this study, we explore the efficacy of machine learning models in predicting wildfire risk using satellite imagery. Leveraging a curated dataset comprising labeled satellite images of wildfires and non-wildfire areas, we employ various feature extraction techniques, including raw pixel flattening, RGB histograms, and deep learning-based feature extraction techniques using MobileNetV2 and ResNet-50 architectures. Subsequently, we train and evaluate 10 distinct models, encompassing K-Nearest Neighbors, Logistic Regression, Support Vector Machines, Naive Bayes, Logistic Regression, Convolutional Neural Networks, a pre-trained Vision Transformer, and ensemble methods like AdaBoost and Gradient Boosting. Our results indicate promising outcomes, with Gradient Boosting Classifier, CNN with ResNet50, and AdaBoost Classifier emerging as the top-performing models based on F2 beta score. Despite these achievements, we call for further work to be done in this area with a particular focus on the data collection methodology as the inclusion of other sensory data could significantly improve results. This study provides valuable insights into leveraging machine learning for wildfire risk prediction and identifies avenues for future research to enhance the robustness and applicability of predictive models for wildfire prevention.

1 Introduction

1.1 Problem Statement

Since the 1980s, wildfires have become increasingly destructive and widespread due to a variety of factors including climate change [2]. These fires have far-reaching impacts on the health of natural ecosystems and the economies of the affected regions. As such, it is crucial to attempt to use technologies at our disposal to assess areas for wildfire risk early. This information could allow environmental protection agencies and other government authorities to prioritize high-risk areas and take preventive measures like managing vegetation or conducting controlled burns. [1].

More and more research harnesses the power of machine learning to analyze satellite imagery to assess a host of environmental risks, including wildfire risk prediction [6]. Taking advantage of the wealth of satellite imagery at our fingertips, and the advanced pattern recognition techniques of machine learning algorithms, we may be able to accurately assess whether a given area is at risk of a wildfire occurring just from a photo.

1.2 Objective

This exploratory investigation seeks to assess the applicability of various machine learning models in discerning patterns within satellite imagery to forecast wildfire risk. Rather than formulating a production-ready methodology for real-world wildfire risk prediction, our objective is to explore the efficacy of diverse model architectures in processing satellite image data. In addition, we will not

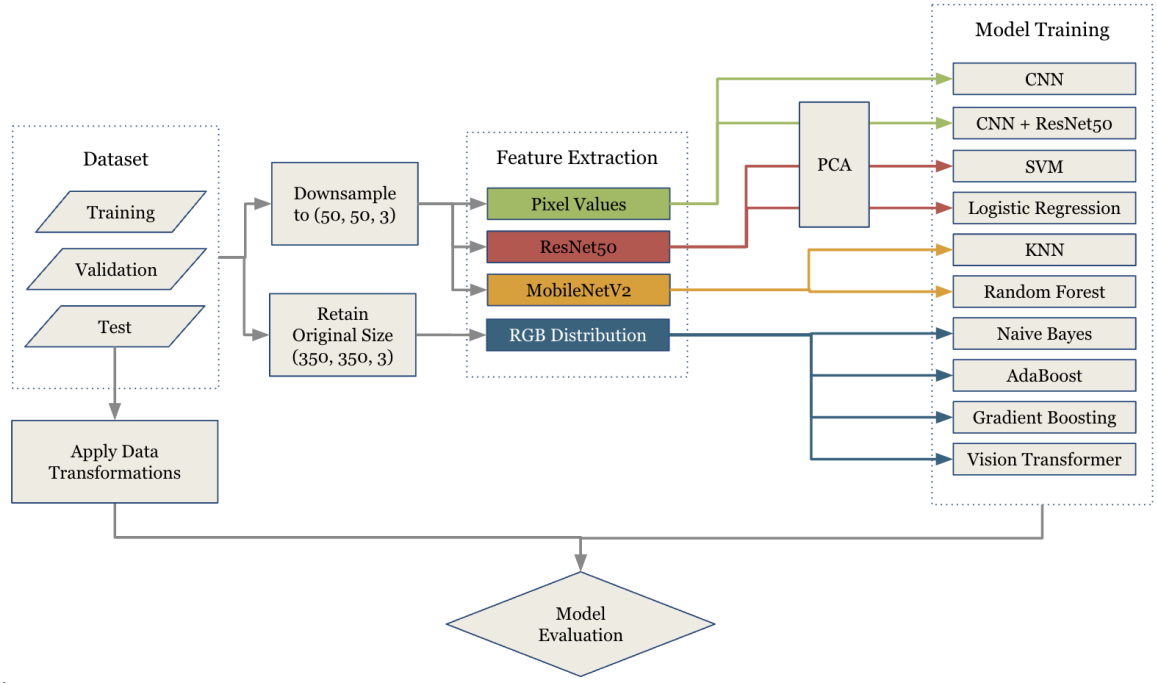

Figure 1: Methodology Block Diagram

encompass data collection methodology, as will mainly be operating on an existing repository of pre-processed satellite images. It is essential to recognize that this simplified approach to wildfire prediction also overlooks other critical factors such as meteorological data. Our focus is specifically on examining satellite imagery as a subset within this broader domain.

2 Data Overview

The dataset for this study originates from the Wildfire Prediction Dataset (Satellite Images) available on Kaggle.com, curated by Abdelghani Aaba [3]. It consists of 22,710 satellite images labeled "wildfire" and 20,140 labeled "nowildfire," already divided into training, validation, and test sets with a 75/15/15 split by the dataset publisher. Each image has dimensions of 350 pixels in height and width.

Aaba's dataset, sourced from Canada's Open Government Portal, was collected and labeled using Longitude and Latitude coordinates of known wildfire locations ($>$ 0.01 acres burned) as query parameters for the MapBox API to extract satellite images.

3 Methods

3.1 Overview

Figure 1 provides an overview of this study's methodology. Since the data provided for us has already been split into training, validation, and test sets, we will begin by applying image augmentations. From these augmentations, we will employ four feature extraction techniques, before conducting principal component analysis (PCA) for dimensionality reduction. We will then use these reduced datasets to fit ten different models and apply the same transformations to validation and test sets to evaluate each model.

3.2 Evaluation Parameters and Success Metrics

Success in the context of the wildfire detection project would entail building a model that accurately identifies wildfire regions in satellite images, achieving precision and recall rates of at least 85%, indicating minimal false alarms and missed detections, respectively. On the other hand, failure

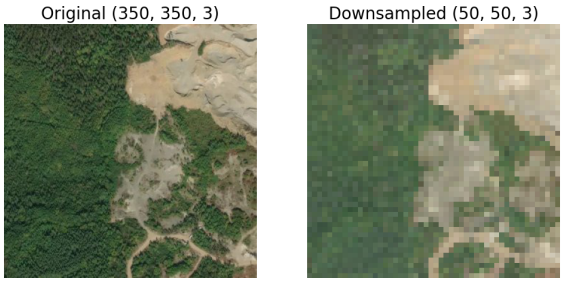

Figure 2: Original image (left) and downsampled image (right)

would involve a model that demonstrates poor performance, with precision and recall rates below 70%, either by inaccurately classifying wildfire and non-wildfire regions or by failing to generalize to unseen data, thereby impeding its practical utility in wildfire management efforts. In addition to precision and recall, our study will also take into consideration accuracy, F1 beta, and F2 beta scores.

3.3 Image Augmentation

When loading images for this study, an RGB (Red, Green, Blue) colorspace is used, giving three distinct color channels. This gives all 42,850 image shapes of (350, 350, 3) each in a matrix representation. Due to the large size of this dataset, we begin the study by first downsampling each image to (50, 50, 3), as shown in Figure 2, which aims to accelerate the time to load and process all images. This downsampled resolution was chosen through visual examination for its computational feasibility while preserving significant information. Additionally, we excluded 8 corrupted image files from our datasets that could not be read.

3.4 Feature Selection

3.4.1 Flatten Raw Pixel Values

Image flattening is a technique commonly used in image classification tasks to convert multi-dimensional image data into a one-dimensional vector. By flattening the image, each pixel value is sequentially arranged in a single vector, allowing it to be processed by traditional machine learning algorithms or fully connected layers in neural networks. This transformation simplifies the input representation while preserving important spatial information, enabling efficient feature extraction and classification. However, flattening may lead to loss of spatial structure and context, particularly in tasks where spatial relationships are crucial, such as object detection or segmentation. Therefore, our study employs image flattening as an elementary baseline feature.

3.4.2 RGB Histograms

RGB color histograms serve as a foundational technique in image processing, providing a method to capture the distribution of colors within an image. This technique involves partitioning the RGB color space into a predefined number of bins along each channel: Red (R), Green (G), and Blue (B). Each channel's intensity values are discretized into these bins, resulting in the creation of individual histograms for each channel. To ensure comparability across images of varying sizes and resolutions, normalization is applied to scale the histogram values to a common range. The normalized histograms from all three color channels are then concatenated to construct a single feature vector. Each element of this feature vector represents the frequency of color intensities within a specific bin across all three channels. This feature vector is subsequently paired with a label corresponding to its class (e.g., 1 for wildfire images, 0 for non-wildfire images).

In our context, we experiment with utilizing these RGB color histograms as input features for three distinct models: Naive Bayes, AdaBoost, and Gradient Boosting Classifiers. This feature extraction technique lays the groundwork for robust classification algorithms, leveraging the inherent characteristics of color distributions within images to aid in our task of binary image classification.

We choose this approach to feature extraction for several reasons:

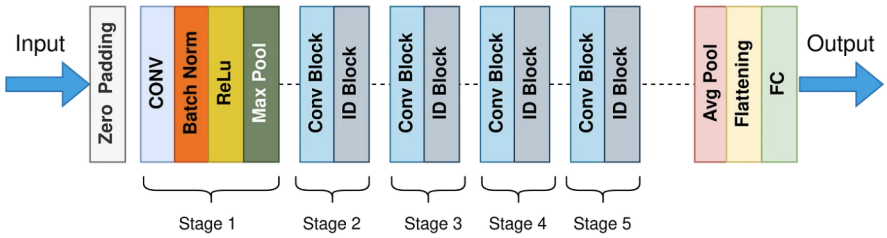

Figure 3: ResNet50 Architecture [5]

- **Simple and Intuitive Representation**: RGB color histograms offer a straightforward representation of color distributions in images, facilitating easy interpretation and understanding.
- **Robustness**: Histograms are computed based on pixel intensities, rendering them invariant to variations in image size, scale, and orientation. This robustness ensures consistent feature extraction across diverse datasets.
- **Low Computational Cost**: The process of calculating RGB histograms is computationally efficient, making them well-suited for large-scale image datasets. This efficiency enables expedited feature extraction without compromising accuracy.

3.4.3 MobileNetV2 Extracted Features

MobileNetV2 is a specialized neural network architecture designed for efficient image analysis. It incorporates expansion and depth-wise convolutional layers as well as a linear activation function (to prevent the model from becoming too complex) to create lightweight yet accurate models. In feature extraction, MobileNetV2 employs its convolutional layers to progressively gather meaningful information from input images, capturing both fine-grained details and high-level patterns relevant to classification tasks.

3.4.4 ResNet-50 Extracted Features

ResNet-50 is a deep convolutional neural network architecture renowned for its effectiveness in image recognition tasks. Introduced by Microsoft Research in 2015, ResNet-50 employs residual connections to address the vanishing gradient problem, allowing for the training of much deeper neural networks with improved performance [4].

To extract features from satellite images using the ResNet50 model, the fully connected layer is removed, leaving only the convolutional base. The satellite images are then passed through the ResNet50 convolutional layers to extract relevant features. Global average pooling is applied to reduce the spatial dimensionality of the extracted features, resulting in a feature vector representing each input image. Finally, the extracted features are passed into other classifiers for wildfire prediction, which allows for ResNet50's powerful feature extraction capabilities to be used in conjunction with other modeling techniques.

For this study, ResNet50 extracts a total of 8192 features, which we reduce to 2000 using PCA to explain a total variance of 0.847. By plotting the first two principal components seen in Figure 4, we observe a noticeable amount of separation between the "wildfire" and "nowildfire" classes, which suggests that ResNet50 extracted features could be suitable for our modeling procedure.

3.5 Models

3.5.1 K-Nearest Neighbors

The k-nearest neighbors (KNN) algorithm is a non-parametric method that operates by finding the 'k' nearest data points to a given input, and classifies the input based on the most common class among its neighbors (when used for classification). We used a K-Nearest Neighbors algorithm on a transformed training set that we feature-extracted and flattened to make wildfire risk predictions. We used pre-trained weights for feature extraction from MobileNetV2 in the Keras package. We

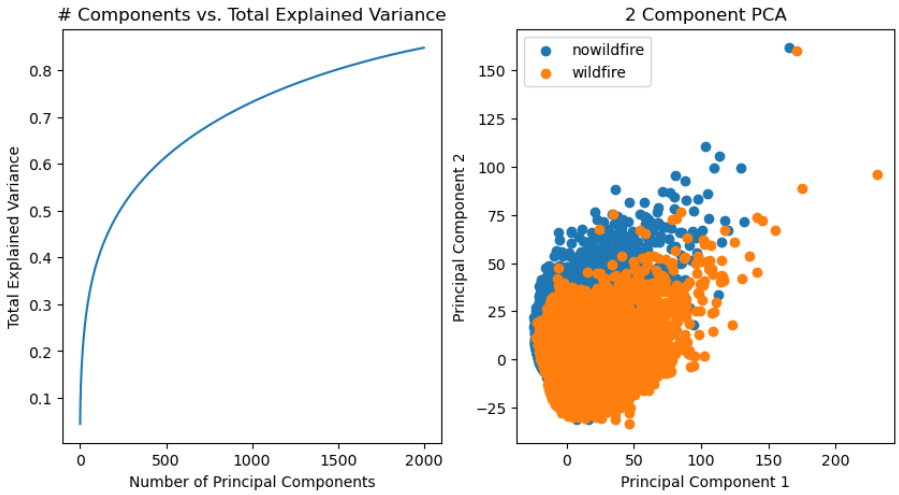

Figure 4: Principal Component Analysis of ResNet50 Extracted Features

used the Scikit-Learn automatic KNN classifier with k=3. We achieved a test accuracy score of 0.895 and an F2 score of 0.901 for this model.

3.5.2 Logistic Regression

Logistic regression is a statistical method used for binary classification tasks, where the outcome variable is categorical and has only two possible values. It models the probability of the outcome using a logistic function, making it suitable for predicting binary outcomes based on the available feature sets. In addition, the algorithm simple and interpretable, which provides transparency to practitioners in understanding why the model makes its predictions. Using the Scikit-learn Python package, we ran logistic regression with the ResNet50 feature set, achieving a train accuracy of 0.946 and a validation accuracy of 0.935. Since the validation performance does not deviate from the training performance by a large margin, we can say that this model generalizes well.

3.5.3 Support Vector Machine

Support Vector Machines (SVMs) are a class of supervised learning models used for classification and regression tasks. They work by finding the optimal hyperplane that best separates data points into different classes, maximizing the margin between classes to enhance generalization performance. SVMs are effective for high-dimensional data, which makes them particularly useful when working with image data. Similar to logistic regression, we fit an SVM using flattened raw pixel values, MobileNetV2, and ResNet50 feature sets, of which ResNet50 had the best performance. The training and validation accuracies of this model are 0.908 and 0.904 respectively. Since the validation performance does not deviate from the training performance by a large margin, we can say that this model generalizes well.

3.5.4 Convolutional Neural Network (CNN)

A CNN employs convolutional layers to extract spatial hierarchies of features from input images, followed by pooling layers to reduce dimensionality, and fully connected layers for classification. We developed a CNN with a baseline single-layered model using the Sequential modeling tool in the Keras package in Python using a sigmoid activation function to simulate a logistic regression methodology since we are working with a binary classification task. We also used the "Adam" optimizer and a binary cross-entropy loss function. Because this baseline model performance was not sufficient for complex classification tasks, we made a more robust CNN with two convolution layers, two pooling layers, two dense layers, a flattening layer, and a dropout layer, which significantly increased performance metrics like test accuracy. We achieved a training accuracy score of 0.907, a validation accuracy of 0.909, a test accuracy of 0.918, and an F2 score of 0.925. Using these metrics we can determine that we were not overfitting the data and still achieved high accuracy.

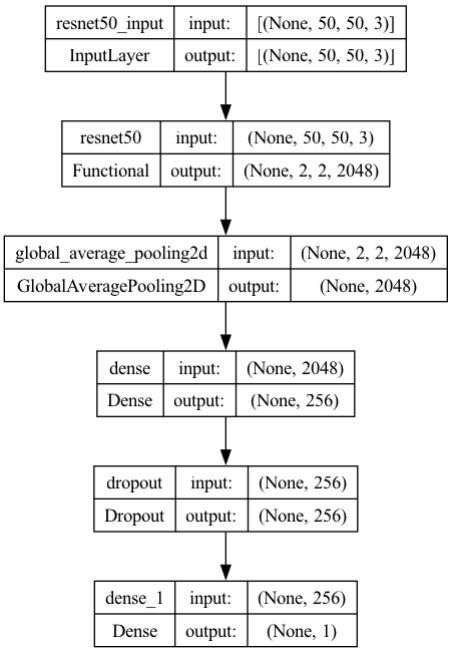

Figure 5: Model Architecture of CNN wrapped around the ResNet50 Pre-Trained Architecture

3.5.5 Random Forest

The Random Forest algorithm is an ensemble learning method used for classification and regression tasks that constructs multiple decision trees during training and outputs the mode (for classification) of the individual trees. The randomness in both feature selection and data sampling helps to reduce overfitting and improve generalization performance. Again using pre-trained weights for feature extraction from MobileNetV2 in the Keras package, we constructed and trained a random forest classifier from Scikit-Learn on feature-extracted and flattened training images. Using the pre-trained weights for feature extraction, we were able to achieve similar accuracy results with our K-Nearest Neighbors and Random forest algorithms to the initial CNN model. We achieved a test accuracy of 0.889 and an F2 score of 0.896 with this model.

3.5.6 CNN with ResNet-50 Pre-trained Block

Given ResNet50's potency for feature extraction, we wrapped the base architecture around a CNN developed with Tensorflow and Keras. First, we loaded a pre-trained ResNet50 model with weights trained on the ImageNet dataset. Then, we utilized global average pooling to extract features from the intermediate layers of the ResNet50 model. Subsequently, we added dense layers for classification, incorporating dropout regularization to prevent overfitting. This comprehensive approach not only enables the model to learn complex representations but also ensures scalability and efficiency in processing satellite imagery for wildfire risk prediction.

Overall, the model comprises a total of 24,112,513 parameters, with 524,801 of them being trainable. The full model architecture is outlined in Figure 5. After 10 epochs of training, shown in Figure 6, the model can significantly reduce loss and increase accuracy. Additionally, the relatively small difference between training and validation accuracies—0.973 and 0.953—respectively suggests that the model is able to generalize well without overfitting.

3.5.7 Naive Bayes

Naive Bayes is a simple classifier that assumes that the predictors contribute equally and independently to selecting the output class. Although this assumption is oftentimes unrealistic in image classification due to pixel dependence and the high-dimensional feature space of images, Naive Bayes still generally produces a satisfactory outcome in the majority of instances due to its simplicity, computational efficiency, and insensitivity to irrelevant features.

Using RGB histograms as input features, we train a Naive Bayes Classifier using sklearn's

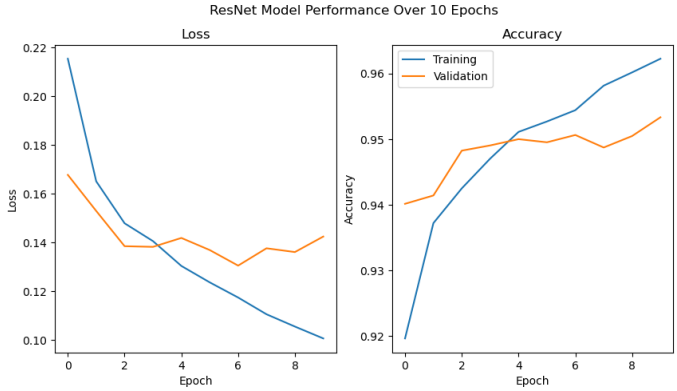

Figure 6: ResNet50 Training and Validation Loss and Accuracies

GuassianNB package. We opt to use a Gaussian Naive Bayes since we can reasonably assume the predictor values are continuous and follow a Gaussian distribution.

The Gaussian Naive Bayes Classifier with RGB feature extraction achieved a test accuracy score of 0.8244. Compared to the training accuracy of 0.80406 we notice that the model performs reasonably well on unseen data and does not show signs of overfitting. The F2 beta score is approximately 0.93, indicating strong performance in identifying true positive samples of wildfires. Overall, the model appears to have a higher number of false positives than false negatives, suggesting potential areas for improvement, especially in minimizing false positives.

3.5.8 AdaBoost Classifier

Adaptive Boosting (AdaBoost) is an ensemble learning method that constructs a robust classifier by iteratively combining the predictions of multiple weak learners – often decision trees with a depth of 1, making decisions based on a single feature threshold. These weak learners are sequentially trained, with each subsequnt model assigning higher weights to the misclassified samples from the previous iteration.

Using RGB histograms as input features, we train an AdaBoost Classifier using sklearn's ensemble AdaBoostClassifier package. We opt to use this ensemble method due to its interpretability and robustness to overfitting compared to more complex models like deep neural networks.

The AdaBoost Classifier with RGB feature extraction performed remarkably well across the datasets. The model achieved a high accuracy of approximately 0.949 on the training data. The test accuracy is even higher at about 0.9571, suggesting that the model generalizes well with no signs of overfitting. The F2 beta score is approximately 0.957, indicating that the model has a high recall rate.

3.5.9 GradientBoost Classifier

Gradient Boosting is another ensemble learning technique that, like AdaBoost, combines the predictions of multiple weak learners to create a strong classifier. In GradientBoosting, the ensemble is built sequentially, with each new weak learner trained to correct the errors made by the previous ones. Instead of adjusting the sample weights like AdaBoost, GradientBoosting fits the new weak learner to the residuals of the current ensemble. This iterative process continues until convergence to an optimal solution.

Using RGB histograms as input features, we train a GradientBoosting Classifier using sklearn's ensemble GradientBoostingClasifier package. GradientBoosting typically achieves higher accuracy than AdaBoost especially when the dataset contains noise. GradientBoosting is also more flexible in the choice of weak learners and loss functions. While AdaBoost primarily relies on decision stumps as weak learners, GradientBoosting can accommodate various types of weak learners, including decision trees with different depths or nodes.

The Gradient Boosting Classifier with RGB feature extraction had a training accuracy of 0.965 and a test accuracy of 0.969. It also has a high F2 beta score of 0.967. Because of this, we see no indication of overfitting as the model seems to generalize well to unseen data.

Model	Test Accuracy	Precision	Recall	F1 Score	F2 Score
Gradient Boosting Classifier	0.969	0.980	0.964	0.972	0.967
CNN + ResNet50	0.961	0.968	0.961	0.964	0.962
Vision Transformer (ImageNet-21k)	0.916	0.957	0.941	0.927	0.956
AdaBoost Classifier	0.957	0.968	0.954	0.960	0.956
Logistic Regression + ResNet50	0.945	0.961	0.939	0.949	0.943
Naive Bayes	0.824	0.767	0.979	0.860	0.927
CNN	0.918	0.938	0.911	0.917	0.925
SVM + ResNet50	0.918	0.937	0.913	0.925	0.917
K-Nearest Neighbors + MobileNetV2	0.895	0.912	0.897	0.905	0.901
Random Forest Classifier + MobileNetV2	0.889	0.903	0.895	0.899	0.896

Table 1: Comparison of Model Test Metrics

4 Results

From the compiled results in Table 1, we can see that the Gradient Boosting Classifier (F2 Score = 0.967), CNN with ResNet50 pre-processing (F2 = 0.962), and the Vision Transformer with ImageNet-21k (F2 = 0.956) were the top-performing models. We chose to prioritize models based on their F2 Scores, which prioritizes minimizing false negatives, due to its double weighting on recall score. We decided on this metric because we thought the consequences of a false negative (failing to predict a wildfire) outweighed the consequences of a false positive (falsely identifying a high-risk area).

5 Discussion

5.1 Comparison

The models that used pre-trained weights generally outperformed our other models and tended to be more computationally efficient (crudely measured on runtime). However, all of our models performed within a 10-point percentage range of each other across all metrics.

Our results also show the potential for ensemble methods for image classification as a computationally efficient alternative to more complicated models with longer runtimes like CNN-based and Transformer-based architectures. This is something that we also observe in our feature extraction techniques as we find that RGB color histograms perform just as well, if not better, as a feature extraction technique compared to using pre-trained CNN architecture for feature extraction. These findings may be useful for researchers who are more computationally constrained in their access to resources.

5.2 Bias, Constraints, and Limitations

It is widely recognized that predictive AI algorithms often exhibit biases when applied to human populations. Despite this, there is a current lack of research addressing geospatial biases and unintended biases in predictive algorithms for disaster prediction. One limitation is the potential bias present in the images of the dataset. We have very little information about the geographic and time period spread of these images. Additionally, we cannot say with certainty that these images were split into the wildfire versus no wildfire labels without bias. We noticed that many images in the no wildfire category had images of lakes and roadways, so our algorithms may have picked up on urbanity more than wildfire risk.

Sampling bias is a significant concern in this context, particularly in the delineation of wildfire boundaries, which are typically documented by forest services. To mitigate this bias, "nowildfire" labels must be sampled from areas outside these boundaries, ensuring a comprehensive representation of various topographies and land uses. Furthermore, the limitations of satellite images in the visual spectrum, such as their inability to capture vertical information and their birds-eye-view perspective, contribute to inherent biases in the data.

This study does not encompass any data collection methods, as we rely completely on an existing repository of satellite images. A significant limitation of the Kaggle dataset is the unspecified time gap between the photo taken and the occurrence of the fire, resulting in an uncontrolled dataset.

Additionally, this methodology overlooks other aspects of wildfire prediction beyond the visual spectrum. These factors include infrared bands and meteorological variables such as temperature, wind, and humidity. Despite these complexities, we acknowledge the simplified nature of our approach to wildfire prediction and suggest researching the aforementioned topics in future work.

5.3 Future Work

In future work, it will be imperative to extend beyond evaluating the performance of models solely based on existing datasets. Transitioning towards practical wildfire prediction in production environments necessitates refining data collection and sampling methodologies. By ensuring comprehensive coverage and representative sampling, we can enhance the robustness and reliability of predictive models for real-world application in wildfire risk assessment and management. Since wildfire prediction is largely dependent on additional factors beyond the visual spectrum future work can leverage other sensory data such as humidity, infrared bands, and wind.

Conclusions

Our models demonstrated a potential for predicting wildfires from satellite imagery using machine learning using a binary classification method. However, due to potential sampling biases, we cannot be sure that these results generalize or are particularly applicable to real-world wildfire predictions. We recognize that our approach is a simplified approach of a larger and more complicated problem but despite this, we still believe that our results here indicate both promise and directionality in terms of showing which models might hold the most promise for future work in this area.

Standards

- huggingface-hub==0.22.2
- keras==2.15.0
- matplotlib==3.8.0
- matplotlib-inline==0.1.6
- numpy==1.26.3
- opencv-python==4.9.0.80
- pandas==2.1.4
- scikit-image==0.22.0
- scikit-learn==1.2.2
- scipy==1.11.4
- seaborn==0.12.2
- tensorflow==2.15.0
- timm==0.9.16
- torch==2.2.2
- torchvision==0.17.2

References

[1] Wildfire prevention. *CAL FIRE*.

[2] Climate change indicators: Wildfires. *US EPA*, 2024.

[3] A. Aaba. Wildfire prediction dataset (satellite images), 2022.

[4] G. Boesch. Deep residual networks (resnet, resnet50) – 2024 guide. 2024.

[5] S. Mukherjee. The annotated resnet-50. 2022.

[6] A. Shmuel. A machine-learning approach to predicting daily wildfire expansion rate. *Fire 6(8)*, 2023.