

Early Detection of Wildfires using Object Recognition from National Park Surveillance Camera Footages

Author: Fabian Roh¹

Affiliation: Seoul International School¹

E-mail: fabiankmroh@gmail.com¹

ABSTRACT

Annually occurring worldwide, wildfires cause substantial damage to animals and plants in nature, as well as humans living in nearby affected areas. When a wildfire grows or spreads to a certain extent, it becomes difficult to extinguish it; therefore, early detection of wildfires is important to prevent or alleviate environmental damage. However, in most national parks or mountain ranges, it is still humans who are responsible for monitoring. This is not a feasible way to 'cover' the vast land that may become a target for fires. To efficiently and effectively detect wildfires early, this study trains a model that detects a wildfire utilizing photo data from mountain range CCTV surveillance feed. For training, EfficientDet D0 Model from Tensorflow Object Detection API was used. The results of this study were successfully validated through 92% accuracy in the validation set.

Keywords: Wildfire Detection, National Park, Artificial Intelligence, Object Detection

1. INTRODUCTION

1.1 Motivation

The World Health Organization (WHO) defines wildfire as “an unplanned fire that burns in a natural area, such as a forest, grassland, or prairie”, the causes of which can be natural or human-derived.

In the United States, nearly 85 percent of wildfires are caused by human actions, including those intentional as in the case of arson. Wildfires may rise from simple human negligence such as failure to attend to campfires or cigarettes. Humans may

also cause fires in the process of burning waste or debris or using equipment. A natural cause of wildfire is lightning.[1]

Wildfire has substantial implications on our nature. Wildfire is responsible for causing air pollution in our nature from large quantities of carbon dioxide, carbon monoxide, and fine particulate matter released into the atmosphere. Wildfires can have direct environmental impact, such as fire damage to flora and fauna, which are known to have high ecological values. Wildfire also has indirect environmental impacts that may occur some time after a wildfire. These include increased soil runoff due to loss of vegetation, dislodged soil through firefighting activity. From wildfire, there may be peat (surface organic layer of a soil that consists of partially decomposed organic matter, derived mostly from plant material [2]) and soil burned in remote areas. This may contaminate water supplies for urban areas. [3] Wildfires have an equally damaging effect on people. WHO reports that a staggering 6.2 million people suffered from wildfires and volcano-related activities during a 10 year period from 1998 including 2,400 deaths. Wildfire smoke is known to cause symptoms such as eye, nose, throat, and lung irritation, coughing and wheezing. More serious health effects range from pulmonary inflammation, bronchitis to cardiovascular problems such as asthma and heart failure. An issue of particular concern with regards to smoke is that significant amounts of mercury is released into the atmosphere, possibly leading to “impairment of speech, hearing and walking, muscle weakness and vision problems for people of all ages”.

According to the California Department of Forestry and Fire Protection’s 2021 Archive, 2,568,948

acres were burned from wildfire, 8,835 reported incidents from wildfire, 3,629 structures affected (damaged or destroyed), and 3 fatalities from wildfire.

Unfortunately, the size and frequency of wildfires are continuously growing due to climate change and global warming. This study utilizes Convolutional Neural Networks, an Artificial Intelligence model primarily used for vision, and aims to detect wildfires at an earlier stage, preventing severe consequences of wildfire.

While not all wildfires are necessarily bad, it is crucial to detect and control early on wildfires that may have a devastating impact on nature and human life. One suggestion for detection of signs of wildfire is monitoring of live-streamed images from various cameras installed in national parks for multiple purposes. Webcams are installed to record a vast overview of certain parts of the park. Yellowstone National Park, for example, offers nine webcams, which includes live feeds to the public via the Internet on its website.[4] Webcam data for other national parks can similarly be accessed upon request. Wildlife observation cams, on the other hand, record locations where the wildlife animals have a high tendency to appear. They are often camouflaged or hidden in discreet locations to not intervene in the animals' ecosystem. Compared to webcams, they are narrowly focused on one specific spot, having an angle of view similar to that of an indoor surveillance camera.

However, wildfires occur across a wide array of land, not limited to national parks. Hence, it makes more sense to observe wildfires from satellite feeds. However, due to the satellite's distance from the Earth, when a fire is detected on raw images, it means the fire has already spread. This study thus utilizes Convolutional Neural Networks, an Artificial Intelligence model primarily used for vision, to accurately detect wildfires at an earlier stage from satellite images. The study aims to offer an important solution to the increasing risks of wildfires, a consequence of climate change and global warming.

1.2 Design of the Paper

Part 2 introduces various methods used in image recognition, such as explanations of Convolutional Neural Networks. Part 3, with the software codes attached, explains the data and model used for this study and how the model was trained. Part 4

validated the model's accuracy by classifying whether the wildfire has occurred or not.

2. Background Knowledge

2.1 Convolutional Neural Network

Convolution Neural Network, abbreviated as CNN, is a deep learning network dominantly used in computer vision tasks. The network is composed of convolution, pooling, and fully-connected layers. It processes the image data in a grid pattern with each grid being an image pixel. The convolution and pooling layers perform extractions of the features of the images. The fully connected layer is responsible for producing the final output which maps the extracted features of the images from the previous layers.

Each layer of the CNN is made up of 3 dimensions that consist of width, height, and depth. Here, depth does not mean the depth of a full Neural Network but instead is a term for the 3rd dimension of an activation volume. This is related to the total number of layers in a network.

The input layer holds raw pixel values, which constitute width and height, and three color channels—Red, Green, and Blue. The convolution layer, often mentioned as the CONV layer, computes the output of neurons connected to local regions in the input. Each layer computes the dot product between the neuron or parameter weight and a small region connected to the input volume. The RELU layer applies an elementwise activation function, or RELU function, to the results calculated in each layer. The POOL layer is where a downsampling operation is performed along the spatial dimensions - width and height. Last, the class scores are computed by the fully connected layer based on the calculations made in previous layers.

The mechanisms of the convolution neural network are established from the following procedures:

- Feature Comparison
- Convolution Repetition
- Pooling
- Activation Function
- Fully Connected Layer
- Backpropagation

2.2 Feature Comparison

The model enhances its performance through comparing features, or similar parts of the image, rather than observing the whole image. The convolution filter is formed through calculating the “match to a feature”. This is done by multiplying each pixel in the feature by the value of the corresponding pixel in the image. The next step is to add up the values and divide by the total number of pixels in the feature.

2.3 Convolution Repetition

The repetition in forming convolutions can ultimately line up features, resulting in the set of filtered images or whole collection of convolutions. Each convolution creates a two dimensional array that carries the features of the image. If a value is close to 1, it is a strong match. If the value is close to -1, it is a strong match for the photographic negative of our feature. A value that is close to 0 means that there is no match of any kind.

2.4 Pooling

The step of pooling is only done on the pooling layer. Pooling is the method of reducing large images yet keeping the key information in the original images. This includes stepping a small window, or a filter, across an image and getting the maximum value from the window at each step.

2.5 Activation Function

The most prominent activation function used for CNN is the ReLU activation function. ReLU activation function swaps out the negative values into zero. This maintains the balance of the network’s learned values.

2.6 Fully Connected Layers

The outputs of this layer are all treated identically as a single list. However, some outputs have more significant votes than the others, which is expressed as weights or connection strengths of the node.

2.7 Backpropagation

Through backpropagation, it adjusts the weights and features accordingly to the loss of each training image. This is then repeated within the subsequent image in the set of the label images. Some deviations may occur within few of the entire

image dataset; however, patterns are stabilized as more images are processed.

2.8 Application of Convolutional Neural Networks (CNN)

The Convolutional Neural Network is applied in various industries, performing tasks such as classification, segmentation, and localization.

CNN is widely used in the field of radiology. Classification is performed to target lesions depicted in medical images and classify them into more than 2 classes. For example, with CNN, lung nodules on Computed Tomography images can be classified as either being benign or malignant.

The most prominent example of CNN in use is self-driving cars. A car obtains the data of the nearby environment using LIDAR or cameras. CNN is utilized to recognize and classify objects on the road, such as traffic lights, pedestrians, and road signals. It is used to model spatial information around the car.

Since CNN is proficient in extracting features from images, this allows the car to capture different patterns as the depth of the network increases. As a result, CNN has become the main type of network used in self-driving cars. There are three notable properties of the CNN architecture that make it flexible and thus a key part of self-driving cars: local receptive fields, shared weights, and spatial sampling. One benefit of such properties is that overfitting can be avoided or reduced. It is also possible to store representations and features that are necessary for classification, segmentation, and localization of images.

3. Experiment

3.1 Data Preparation

The database [5] used for this study contains a selection of wildfire smoke images.

The images are classified into 3 classes:

1. Smoke
2. Partial Smoke
3. No Smoke

Image database is divided in 4 directories (Table in final format):

- Training (51 images)

- Gt_training
 - Training images manually segmented by a human observer into three classes mentioned above
- Testing (51 images)
- Gt_testing
 - Testing images manually segmented by a human observer into three classes mentioned above



Fig 1: (Top left) Image from training folder. All RGB colors are included. (Top Right) Image from gt_training folder that corresponds to the image from the training folder. Only in grayscale.

Images are in 640×524, 704×576, 1280×960 pixels dimensions. The smoke area is manually segmented. The smoke region is highlighted as white, and the partial-smoke region is segmented as gray. No smoke is segmented as black. Image from testing folder.

TF Record

To utilize the Tensorflow Object Detection API, the data was first converted to TF Record using Labelimg and categorized into train and validation set. TF record files were stored into ‘train’ and ‘val’ directories accordingly. Also, a label_map.txt file was generated for the following data set. In the label_map.txt file, the training class information is saved in dictionary form.

```
test_record_fname = DATASET_PATH + 'val.record'  
train_record_fname = DATASET_PATH + 'train.record'  
label_map_pbtxt_fname = DATASET_PATH + 'annotations/label_map.pbtxt'  
fine_tune_checkpoint = os.path.join(DEST_DIR, "checkpoint/ckpt-0")
```

Code 1: Categorizing into train and validation set

3.2 Network Model Download & Setting

The network model that will be trained is derived for the list that has Tensorflow Model Zoo 2. This study used the EfficientDet D0 version for training. EfficientDet is a deep learning network model that is based on image-classifying Efficient Net.

The pipeline.config is edited to match the model with the learning environment. Batch_size, learning_rate, and num_classes (denote number of classes being learned) were mostly edited. Existing, pre-trained model had high batch sizes since it was trained in a high-spec environment using TPUs. In this study, the batch_size was reduced to prevent OOM (Out of Memory) errors. In accordance with this reduction, the num_step was increased to a point where overfitting does not occur.

```
num_steps = 5000  
num_eval_steps = 100  
  
MODELS_CONFIG = {  
    'efficientdet_d0': {  
        'model_name': 'efficientdet_d0_coco17_tpu-32',  
        'pipeline_file': 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config',  
        'batch_size': 8  
    },  
}  
  
selected_model = 'efficientdet_d0'  
  
MODEL = MODELS_CONFIG[selected_model]['model_name']  
pipeline_file = MODELS_CONFIG[selected_model]['pipeline_file']  
batch_size = MODELS_CONFIG[selected_model]['batch_size']
```

Code 2: Hyperparameter settings for model and training

3.3 Training

Utilizing Tensorflow Object API, the network model is trained by activating model_main_tf2.py. In this case, information on pipeline.config file mentioned above, such as its route and model checkpoint, were transmitted using parameters. All learning models were trained using a CentOS Linux machine with an Intel Xeon E5-2680 V3 (@ 2.50GHz) Dual-core CPU, paired with 256GB of memory.

```
!python /content/models/research/object_detection/model_main_tf2.py \  
--pipeline_config_path={pipeline_fname} \  
--model_dir={model_dir} \  
--alsologtostderr \  
--num_train_steps={num_steps} \  
--sample_1_of_n_eval_examples=1 \  
--num_eval_steps=100
```

Code 3: Shell commands to run an object detection API

4. Results & Analysis

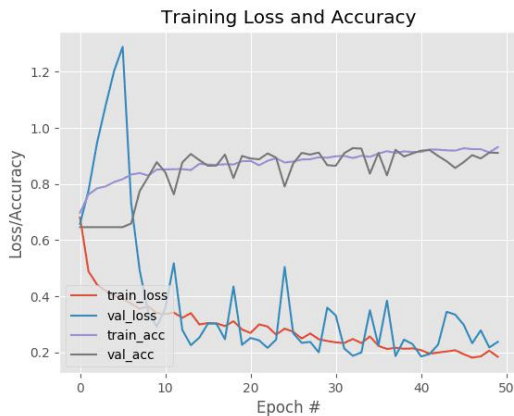


Fig 2: Training and validation loss/accuracy graph

To find the optimal hyperparameter value, the study was conducted with various learning rates. When adjusting learning rates from 1/1000 to 1/10 via log scale, 1/100 and 1/10 showed the optimal results. However, for the learning rate of 1/10, it was inferred that overfitting occurred due to the surge of loss after certain epochs. Ultimately, 1/100 was decided as the final learning rate. Training was conducted in the following manner. As a result, an accuracy of 92% could be obtained as shown in the graph above.

To verify if the tests were properly conducted, the trained model was called to the disk, and random images were inserted to the data set.

```
for image_path in images:
    print(image_path)
    image_np = load_image_into_numpy_array(image_path)

    input_tensor = tf.convert_to_tensor(
        np.expand_dims(image_np, 0), dtype=tf.float32)
    detections, predictions_dict, shapes = detect_fn(input_tensor)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'][0].numpy(),
        (detections['detection_classes'][0].numpy() \
         + label_id_offset).astype(int),
        detections['detection_scores'][0].numpy(),
        category_index,
        use_normalized_coordinates=True,
        max_boxes_to_draw=200,
        min_score_thresh=.6,
        agnostic_mode=False,
    )

    cv2.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
```

Code 4: Verifying the trained model with validation data sets

As a result, it could be confirmed there were cases when the smoke is not properly separated from the background. This is due to the lack of quantity and consistency in the training data. The data used for this study are not extracted from videos but from still image data; therefore, the recognition accuracy was relatively lower due to the lack of data that had a slightly different outlook in the same wildfire situation.



Figure 3: Correct prediction sample images from validation sets

5. Conclusion

To detect wildfire earlier, this study trained a model detecting wildfire using photo data from mountain range surveillance CCTVs. Tensorflow Object Detection API's EfficientDet D0 model was used to train the model. The model had a successful accuracy of 92% in the validation set.

In order to develop an enhanced model in the future, fine quality training data is first needed. To enhance the model accuracy, the training image should be derived from individual frames of video data, not from photos. Also, the limit of CCTV surveillance is the difficulty to detect wildfire at night time. This study will continue to seek for other methods, such as infrared cameras, that can detect wildfires without the presence of light, being able to protect residents and animals without any time gap.

6. REFERENCES

- [1]. US National Park Service, "Wildfire Causes and Evaluations", March 8, 2022, <https://www.nps.gov/articles/wildfire-causes-and-evaluation.htm>
- [2]. International Peatland Society, "What is peat?", <https://peatlands.org/peat/peat>
- [3]. National Fire Chiefs Council, "Environmental impact of wildfires", <https://www.ukfrs.com/guidance/search/environmental-impact-wildfires>.
- [4]. US National Park Service, "Webcams - Yellowstone National Park", June 17, 2022,

<https://www.nps.gov/yell/learn/photosmultimedia/webcams.htm>

- [5]. Damir Krstinic, Toni Jakovcevic, "Image database", February 10, 2010, http://wildfire.fesb.hr/index.php?option=com_content&view=article&id=49&Itemid=54.