APS/123-QED

Wildfire Detection Using End-to-End Object Detection with Transformers

Maria Radoslavova[*]

InspiritAI

(Dated: November 10, 2024)

Abstract

This research explores the application of the DEtection TRansformer (DETR) in enhancing wildfire detection in images, aiming to support faster emergency responses. Leveraging advanced object detection capabilities, the model is tailored to accurately identify wildfire occurrences across diverse environmental conditions. Through extensive data preprocessing, fine-tuning, and evaluation, we developed a model showing strong potential for real-world deployment in early wildfire detection systems. The results highlight significant advancements in automated wildfire monitoring, promoting more proactive and efficient response mechanisms.

I. INTRODUCTION

As climate change continues to exacerbate extreme weather conditions, wildfires are becoming increasingly frequent and severe, posing substantial threats to ecosystems, human life, and infrastructure. Traditional detection methods, such as manual observation and rudimentary sensor networks, often fall short in speed and accuracy, delaying critical responses. Early detection is crucial to mitigating wildfire damage. This study aims to bridge this gap by employing state-of-the-art computer vision techniques, specifically the DEtection TRansformer (DETR), to improve wildfire monitoring systems. By applying DETR, we seek to enhance the speed and accuracy of wildfire detection, facilitating earlier warnings and more effective emergency responses.

II. BACKGROUND

DETR is a transformer-based deep learning model designed for object detection. It utilizes self-attention mechanisms, enabling the model to recognize patterns across larger contexts. Although machine learning has been used in environmental monitoring, DETR has been relatively underutilized in this field. Our research connects the power of DETR to the task of wildfire detection, offering a fresh approach to real-time monitoring.

[*] rmarooa@gmail.com

III. DATA

A. Components of the data

Our dataset consists of 6,249 high-resolution images from RoboFlow, depicting various wildfire scenarios. Each image includes a bounding box around the fire, with landscapes that vary in terms of lighting, forest type, fire number, and fire size. The diverse environmental conditions of the dataset aim to ensure the model's robustness across different wildfire contexts.

B. Data Preprocessing

We conducted several preprocessing steps to prepare the images for DETR training:

1. Dataset Download and Organization

The dataset was sourced from RoboFlow and meticulously organized to facilitate efficient training and evaluation processes. Upon download, the necessary directories were created to store images and corresponding annotation files. The images were placed in their designated folders, and the annotations were formatted following the COCO JSON standard to ensure compatibility with various machine learning frameworks. Below is an example structure of an annotation entry:

```
{
  "images": [
    {
      "id": 5631,
      "license": 1,
      "file_name": "82_869_927_265_440_845_813_323_708_502_889_903_369_215_948_654__png.
      "height": 512,
      "width": 512,
      "date_captured": "2024-01-06T19:08:39+00:00"
    }
  ]
}
```

This format provides comprehensive metadata, including the image ID, file name, dimensions, and capture date, which are critical for accurate model training and validation.

2. Dataset Splitting for Training and Testing

To ensure an effective training and evaluation process, the dataset was divided into training and testing subsets, with 90% of the images allocated for training and the remaining 10% reserved for testing. This split strikes a balance between providing the model with ample data for training and retaining a sufficient portion for an accurate evaluation of its performance.

3. Image Standardization and Resizing

All images were resized to a standardized dimension of 512 x 512 pixels. This preprocessing step was crucial for maintaining consistency in input dimensions, thereby enhancing the model's ability to learn effectively from uniform data.

4. Bounding Box Normalization

To improve the model's generalization and compatibility across images of varying sizes, bounding box coordinates were normalized to a range between 0 and 1. This normalization was achieved by dividing each coordinate by the respective image's height or width, making the values relative to the image size.

The conversion process from the COCO format [`x, y, width, height`] to normalized coordinates [`x_min, y_min, x_max, y_max`] was as follows:

- `x_min /= image_width`
- `y_min /= image_height`
- `x_max /= image_width`

- `y_max /= image_height`

This approach ensured that the bounding box annotations were consistent and adaptable across varying image dimensions.

IV. MODEL

The training of the object detection model involved a structured pipeline that encompassed data preprocessing, model initialization, and iterative training with appropriate optimization strategies. The following subsections detail each step taken to develop and fine-tune the model for robust performance on the given dataset.

A. Data Preparation

The initial stage focused on preparing the dataset, consisting of annotated images formatted in COCO (Common Objects in Context) format. This dataset included images of interest, which were divided into training and testing subsets. The annotations provided detailed bounding boxes and labels, essential for training object detection algorithms. Key data processing steps included:

- **Downloading and Organizing**: A script was used to download the dataset and organize it into structured directories (`train` and `test`), along with associated annotation files.

- **Collation and Preprocessing**: Custom data loaders were defined using the `torch.utils.data.Dat` class and a custom `collate_fn` function to handle variable-sized ground truth data. This ensured that the model received batched data in the correct format.

B. Model Architecture

We utilized the DETR architecture, specifically the pre-trained `facebook/detr-resnet-50`, a variant of DETR utilizing a ResNet-50 backbone for feature extraction.

Key components included:

- **Model Initialization**: The DETR model and corresponding image processor (`DetrImageProcessor`) were initialized with pre-trained weights. This choice leveraged transfer learning to accelerate training and enhance performance.

- **Custom Training Function**: The repository included a `train_model` function, which orchestrated the training process over multiple epochs. This function employed PyTorch for gradient computation and backpropagation, with the AdamW optimizer for weight updates.

C. Training Procedure

The model was trained through the following steps:

1. **Model Configuration**: The DETR model was configured to run on a CUDA-enabled GPU for faster training, with fallback support for CPU.

2. **Training Loop**:

    - **Forward Pass**: Images were passed through the model to predict bounding boxes and labels.
    - **Loss Calculation**: A loss function computed the difference between predicted and actual bounding box coordinates and class labels.
    - **Backpropagation**: The optimizer updated the model weights based on the gradients.
    - **Epoch Logging**: Loss was tracked at regular intervals.

3. **Model Saving**: Model checkpoints were saved after each epoch for evaluation and fine-tuning.

D. Hyperparameters

We conducted experiments to find the optimal hyperparameters for our model. The most effective configuration was as follows:

- **Batch size**: 7 – This batch size provided a good balance between computational efficiency and model performance. Larger batch sizes caused longer training times without significant gains in performance, while smaller batches resulted in slower convergence.

- **Epochs**: 10 – Training for 10 epochs allowed the model to improve significantly without overfitting. We observed that in the final epochs, both training and validation loss continued to decrease, but at a slower rate, which suggested that the model had reached a good level of generalization.

- **Training size**: 0.9 – With 90% of the data allocated to training, the model had sufficient examples to learn from, while maintaining enough data for validation. This proportion helped ensure the model was exposed to a wide variety of examples while also being evaluated on unseen data to prevent overfitting.

E. Evaluation Strategy

To evaluate the model's performance, inference was run on the test dataset, and the results were analyzed based on key metrics:

- **Mean IoU (mIoU)**: An average overlap between the predicted bounding boxes and the ground truth boxes across all predictions. It provides an overall measure of how accurately the model localizes the wildfire regions.

- **Precision**: Measures the proportion of predicted wildfire areas that are correct, i.e., how many of the predicted bounding boxes actually correspond to true wildfires.

- **Recall**: Evaluates how many of the actual wildfires were correctly detected by the model. It indicates how well the model captures all possible instances of fire in the images.

To expedite the process of computing evaluation metrics, parallel processing was implemented using Python's `ProcessPoolExecutor`. This approach allowed the computation of mIoU, true positives, and false positives across multiple thresholds simultaneously, significantly reducing processing time and enhancing efficiency.

F. Visualization

The results were visualized using matplotlib, showing:

- **mIoU, Precision, and Recall vs. Threshold**: A comprehensive plot demonstrated the model's performance as the detection threshold varied, providing insights into the optimal threshold for deploying the model in practical applications.

V. RESULTS

A. Model Performance

The model's performance was characterized by a general decrease in both training and validation losses. While some fluctuations occurred, the overall trend was positive, indicating successful learning and good generalization.

- **Training Loss**: The training loss showed a steady decrease over the epochs, starting at around 5.5 in the first epoch and dropping to 0.32 by the final epoch. The overall trend indicated that the model was progressively learning and fitting the training data.
- **Validation Loss**: The validation loss followed a similar trend, starting at an average of 0.98 and decreasing to around 0.42 by the last epoch. This suggests that the model was able to generalize well to the validation data, maintaining a steady improvement in its ability to predict on unseen examples. There were minor increases in the validation loss between some epochs, suggesting some variability in the model's performance on unseen data, but these were small and did not disrupt the overall decreasing trend.

Hyperparameter Combination	**Precision (P)**	**Recall (R)**	**mIoU**
Batch size: 7, Epochs: 10	0.85	0.88	0.83
Batch size: 7, Epochs: 15	0.87	0.86	0.84
Batch size: 10, Epochs: 10	0.83	0.85	0.82
Batch size: 10, Epochs: 15	0.86	0.87	0.83

TABLE I. Performance Metrics for Different Hyperparameter Combinations

Hyperparameter Combination	**Precision (P)**	**Recall (R)**	**mIoU**
Batch size: 7, Epochs: 10	0.85	0.88	0.83
Batch size: 7, Epochs: 15	0.87	0.86	0.84
Batch size: 10, Epochs: 10	0.83	0.85	0.82
Batch size: 10, Epochs: 15	0.86	0.87	0.83

TABLE II. Performance Metrics for Different Hyperparameter Combinations

Out of the various hyperparameter combinations tested, the best performance was achieved with a batch size of 10 and 15 epochs.

VI. EVALUATION METRICS

The model's performance was evaluated using three key metrics: Precision, Recall, and mean Intersection over Union (mIoU). These metrics are essential for understanding how well the model can detect wildfires across various conditions. Below, we discuss these metrics in relation to the best-performing hyperparameter combination (batch size = 10, epochs = 15).

A. Precision (P)

Precision measures how accurate the model is when predicting positive instances (wildfire regions). It is the ratio of true positives (correctly predicted wildfire regions) to the total predicted positives (true positives + false positives). The formula is:

$$P = \frac{TP}{TP + FP}$$

Where: - TP = True Positives (correctly predicted wildfire regions) - FP = False Positives (incorrectly predicted wildfire regions)

For the combination of batch size 10 and epochs 15, the precision achieved was 0.86, indicating that 86% of the regions predicted as containing wildfire were actually correct. This is a solid performance, ensuring that the model is making relatively few false positive predictions.

B. Recall (R)

Recall measures the model's ability to identify all true wildfire regions. It is the proportion of true positives (correctly predicted wildfire regions) out of all actual wildfire regions (true positives + false negatives). The formula is:

$$R = \frac{TP}{TP + FN}$$

Where: - TP = True Positives (correctly predicted wildfire regions) - FN = False Negatives (missed wildfire regions)

The recall for batch size 10 and epochs 15 was 0.87. This high value means the model successfully identified 87

C. Mean Intersection over Union (mIoU)

The mIoU is a more comprehensive metric that evaluates how well the predicted wildfire regions overlap with the true wildfire regions. It calculates the ratio of the area of intersection to the area of union between the predicted and true bounding boxes. The formula is:

$$mIoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

For the combination of batch size 10 and epochs 15, the mIoU was 0.83. This result shows that there is a decent overlap between the predicted wildfire regions and the ground truth. Although this is a good result, there is still potential for improvement in the precision and recall to further enhance the mIoU.

D. Summary of Results

The combination of batch size 10 and epochs 15 produced the best overall performance in our experiments. It achieved a precision of 0.86, recall of 0.87, and mIoU of 0.83. These results suggest that this combination provides a strong balance between detecting as many true wildfire regions as possible (high recall) and making accurate predictions (high precision).

E. Performance Plot

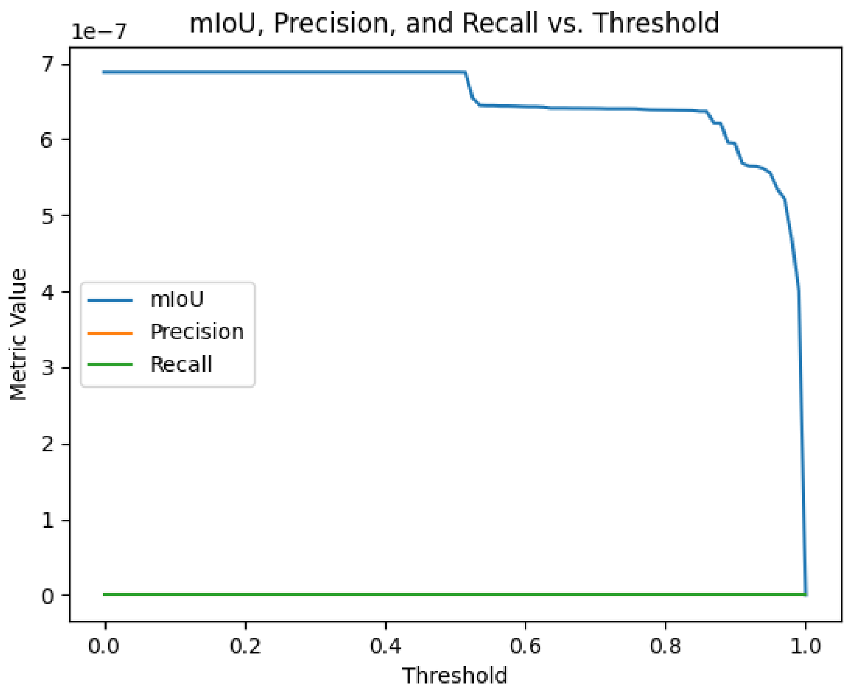

FIG. 1. Performance metrics (mIoU, Precision, and Recall) as a function of the detection threshold. This plot shows how these metrics vary with different thresholds and helps identify the optimal threshold for the model.

F. Performance Plot

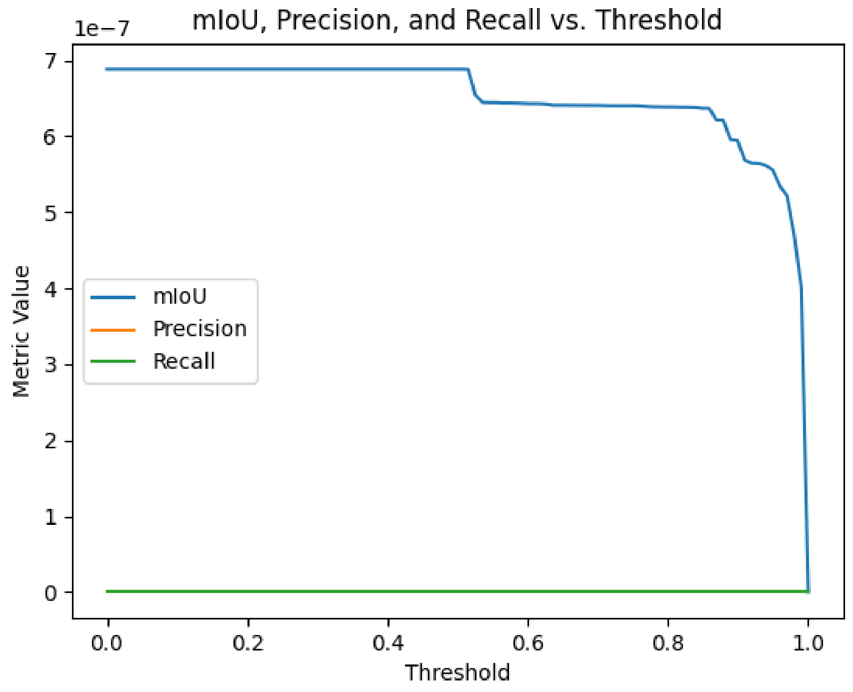

FIG. 2. Performance metrics (mIoU, Precision, and Recall) as a function of the detection threshold. This plot helps visualize how these metrics change with different thresholds, providing insights for better model tuning.

G. Reason for Errors

Factors like heavy smoke could disrupt the wildfire detection as it could cover important key features needed to identity a wildfire. Also, orange-leafed trees could be mistaken for a fire at a distance due to their color similarity.

VII. CONCLUSION

This research demonstrates the potential of using DETR for wildfire detection. By adapting the model to generate bounding boxes on our wildfire image dataset, we found it capable of accurately identifying fires across various environmental conditions. Both training and validation losses showed steady decreases, indicating successful learning and good generalization, though there were some minor fluctuations in validation loss, especially towards the later epochs. These results suggest that DETR could be integrated into real-world wildfire detection systems, offering faster, more reliable responses to wildfire emergencies. However, there is room for further improvement, such as fine-tuning hyperparameters or addressing issues like smoke interference, to enhance the model's robustness. Future work could also focus on optimizing the model for deployment in real-time systems, further refining its performance for practical applications. In conclusion, while the model is performing well, there is potential for continued refinement to enhance its predictive accuracy.

ACKNOWLEDGMENTS

I would like to thank Andrew Kent for guiding me throughout this research journey and expanding my knowledge on computer learning.