BBM416 - COMPUTER VISION
TERM PROJECT FINAL REPORT

Group Members
Deniz Gönenç 21946146
M. Davut Kulaksız 21946419
Hikmet Güner 21946179

INTRODUCTION

In this project, we aim to develop a computer vision system for detecting fires. In recent years, wildfires have been happening more often around the world, leading to devastating environmental and economic consequences. The detection of fires can help prevent or mitigate damage to property and save lives. The main challenges in this problem include the variability in fire sizes, shapes, and environments where fires can occur. Our motivation for this project is to contribute to the development of a real-world solution for detecting fires in diverse environments.

1 METHOD

We utilized transfer learning with the **ResNet (Residual Network)** architecture implemented using **PyTorch**. Transfer learning involves leveraging the knowledge gained from training on a large dataset (pretrained model) and adapting it to a specific task (fire detection) with a smaller dataset. The following steps are involved:

- **Dataset Preparation:** Gather a dataset of labeled images containing both fire and non-fire instances. We used the D-Fire dataset for this. Setup a dataset class.
- **Preprocessing:** Perform data augmentation techniques such as random flipping and rotation, color jittering to increase the dataset variance and improve generalization. Resize and normalize the images to ensure consistent training.
- **Transfer Learning:** Load a pretrained ResNet model (ResNet-101) that has been trained on a large-scale dataset (ImageNet). Freeze all the layers except the last two layers to train them.
- **Training:** Train the model using the labeled dataset. Employ the Adam optimizer with a suitable learning rate(0.001). Monitor the loss and accuracy metrics during training.
- **Evaluation:** Split the dataset into training and validation sets. Measure the model's performance using evaluation metrics such as accuracy, precision, recall, and F1 score. Monitor the learning curves (loss vs. epochs, accuracy vs. epochs) to assess the training progress.
- **Further Evaluation:** Test the trained model on different datasets and measure the model's performance. Plot the results.

2 EXPERIMENTAL SETTINGS

For this project, we utilized the **D-Fire Dataset**, which is available at https://github.com/gaiasd/DFireDataset. The dataset consists of images that are annotated according to the **YOLO** format, providing bounding box coordinates for fire and smoke instances.

The dataset statistics are as follows:

Regarding the training environment, we primarily trained the model on our local environment using CUDA, taking advantage of the GPU's parallel processing capabilities to expedite the training process. We also utilized Google Colab for training and collaboration purposes.

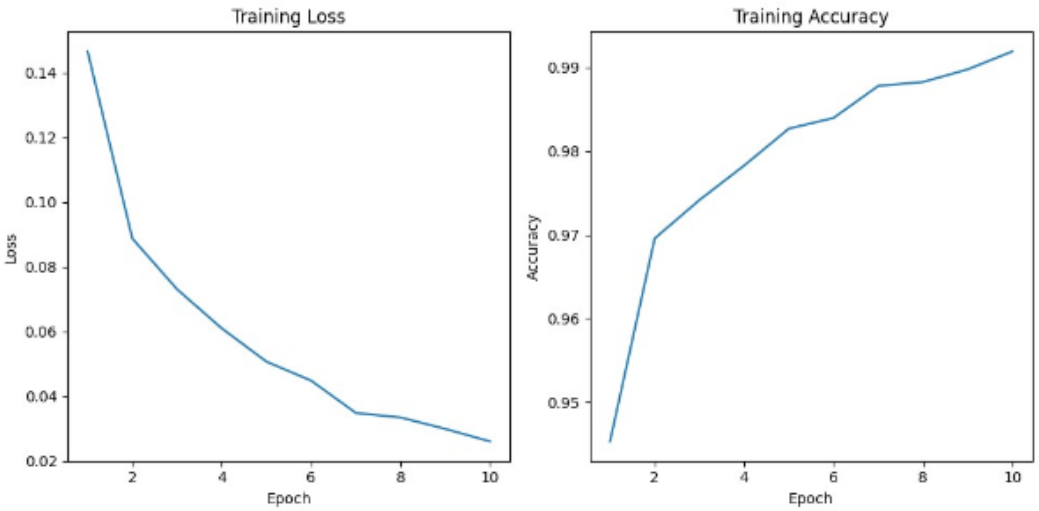
Table 1: Dataset Statistics

Number of Images	
Only fire	1,164 images
Only smoke	5,867 images
Fire and smoke	4,658 images
None (no fire or smoke)	9,838 images
Number of Bounding Boxes	
Fire	14,692 bounding boxes
Smoke	11,865 bounding boxes

3 Experimental Results

3.1 Quantitative Evaluation:

3.1.1 D-Fire



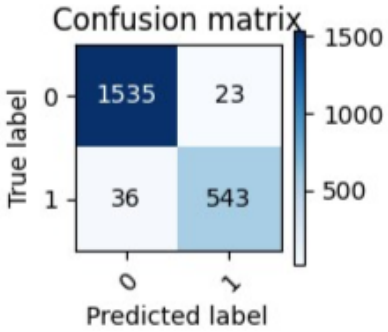


Table 2: Model Performance

Evaluation Metrics	Value
Validation accuracy	97.378%
Test accuracy	97.239%
F1 Score	0.948
Precision	0.959
Recall	0.938

As we can see our model is performing quite well on the dataset it is trained on. Compared with FireNet-v2 [1] which has an accuracy of 98.43% our model's accuracy is not too far behind.

3.1.2 FIRE

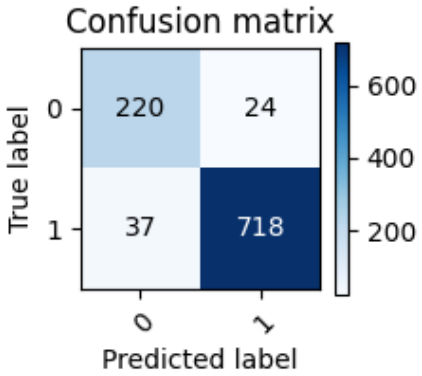
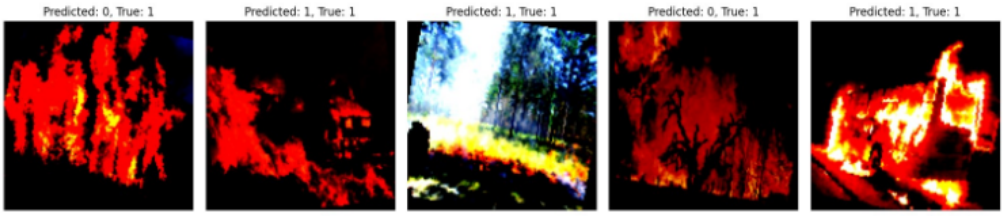


Table 3: Model Performance

Evaluation Metrics	Value
Test accuracy	93.894%
F1 Score	0.959
Precision	0.968
Recall	0.951

Our model is performing slightly worse on the FIRE dataset compared to the D-Fire dataset. But the F1 score, precision and recall values are similar.

3.2 Qualitative Evaluation:



Our model struggled to predict fires that are in the midst of darkness.

3.3 Runtimes:

It took roughly half a day to train the model.

Table 4: Testing Time Metrics

Metric	**Value**
Total inference time	77.199 seconds
Inference time per image	0.077276 seconds

4 Discussions/Conclusions

In the course of this project, we have encountered various challenges. The most significant one was the resource-intensive nature of our initial CNN model, which caused memory exhaustion issues during training on both Google Colaboratory and our local machines. The problem was particularly severe on local machines, necessitating training with a limited number of epochs. Despite our efforts to identify the root cause, the exact reason for this memory issue remains unclear; it may be attributable to the size of our dataset or other unknown factors.

To overcome this obstacle, we trained an alternative model using the **ResNet** architecture, which proved to be both more memory-efficient and more accurate. This model achieved a promising accuracy of 96%.

We have extensively experimented with different parameters for the **ResNet** model to achieve optimal results in fire detection. This has resulted in improved accuracy and reliability in identifying the presence of fires in images.

Moreover, we have extended our fire detection system by incorporating an object detection model. This additional model allows us not only to detect fires but also to locate them within the image by identifying specific bounding boxes around the fire instances. This enrichment provides valuable spatial information, enabling more precise and effective fire detection.

By implementing these improvements, we have created a more robust and accurate fire detection system that can effectively operate in diverse environments and varying circumstances. Our efforts have already yielded promising outcomes, contributing to the advancement of the field of fire detection.

5 BONUS

5.1 Object Detection

In addition to our fire detection system based on transfer learning with **ResNet**, we have also developed an object detection model using **YOLOv8** to further enhance our capabilities. This object detection model specifically focuses on detecting and localizing fires and smokes within images. We recognized the importance of accurately identifying these specific instances as they provide critical information for fire detection and emergency response systems.

To train our **YOLOv8** model, we faced several challenges:

- Firstly, the limited availability of bounding box labels in the **D-Fire Dataset** prompted us to select a subset of 500 images and annotate them using the **Computer Vision Annotation Tool (CVAT)**. This process ensured accurate and consistent labeling of fire and smoke instances, enabling effective training of the object detection model.
- The training process of the **YOLOv8 model** involved training for **50 epochs**, which spanned approximately **10 hours** due to the complexity of the model. Despite the lengthy training time, we deemed it necessary to ensure the model achieved optimal performance.

After training, we evaluated the performance of the YOLOv8 object detection model using several metrics, as shown in Table 5.

Table 5: Performance Metrics of the YOLOv8 Object Detection Model

Class	Precision (Box P)	Recall (Box R)	mAP50	mAP50-95
All Classes	0.648	0.553	0.606	0.324
Smoke	0.687	0.573	0.646	0.401
Fire	0.608	0.534	0.566	0.247

These results indicate that the YOLOv8 object detection model demonstrates moderate to good performance in detecting and localizing fire and smoke instances. While there is room for improvement, particularly in achieving higher precision and recall, the model provides a valuable contribution to our fire detection system by enabling accurate identification of fire and smoke regions within images.

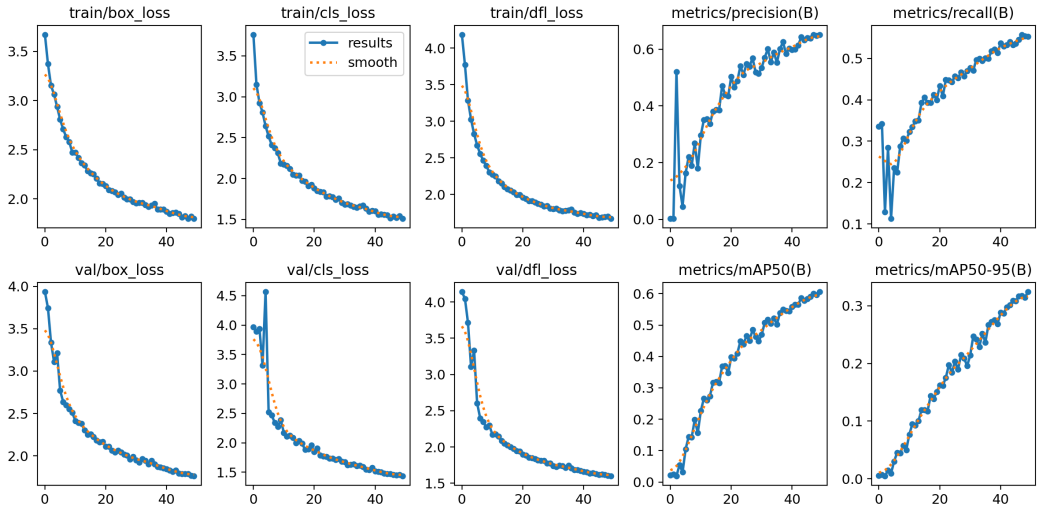


Figure 1: Labels

Figure 2: Predictions

6 REPOSITORY

The code and resources related to this project are available in our GitHub repository at `https://github.com/davutkulaksiz/Computer-Vision-Term-Project`. The repository contains the implementation of our fire detection system using transfer learning with ResNet and the object detection model using YOLOv8. It also includes the necessary dataset, pre-trained models, and scripts for training and evaluation.

Additionally, the repository includes videos demonstrating the fire and smoke detection capabilities of our models. These videos showcase the detection results on various real-world scenarios and can provide valuable insights into the performance and potential applications of our fire detection system.

REFERENCES

[1] Chen, J., Liu, Y., & Cao, C. A Fire Detection Algorithm Based on Infrared Image Analysis. In *Proceedings of the IEEE International Conference on Computational Science and Engineering*, 2017.

[2] Wang, J., Wang, H., Li, X., & Fang, Z. Real-time forest fire detection based on image processing. In *Proceedings of the IEEE International Conference on Natural Computation*, 2019.

[3] Li, X., Shen, H., & Lu, X. Wildfire detection from remotely sensed images using a combined color and motion features approach. *Journal of Applied Remote Sensing*, 11(4), 046003, 2017.

[4] Hoang, T. D., Nguyen, D. H., Nguyen, N. V., Nguyen, D. M., & Le, T. M. Real-time forest fire detection using deep learning. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 3950-3955. IEEE, 2019.

[5] Venâncio, P. V. A. B., Lisboa, A. C., & Barbosa, A. V. An automatic fire detection system based on deep convolutional neural networks for low-power, resource-constrained devices. *Neural Computing and Applications*, 2022.