Fire Detection and Segmentation using YOLOV5 and U-NET

Jawher DRIDI, Mohamed BOUZID, Engineering Students at Ecole Polytechnique de Tunisie
Supervised by : **Rafik GHALI** and **Wided SOUIDENE**

***Abstract*—The environmental crisis the world faces nowadays has never been greater or more complex. A large area of the earth is covered by forest and urban woodlands are endangered by forest fires that have increased rapidly in the last years in Tunisia and worldwide. Thanks to the fast development of sensors and technologies as well as computer vision algorithms, new approaches for fire detection have been proposed. However, these approaches face several limitations that need to be resolved, precisely, presence of fire-like objects, high false alarm rate, detection of small size fire objects, and high inference time. An important step for vision-based fire analysis is the segmentation of fire pixels. Hence, we propose, in this project, a novel architecture, combining YOLOv5 and U-net architectures, for fire detection and fire segmentation. The proposed method is based on a database of wildland fires mixed with a database of fire-like objects. The experimental results we obtained, proves that the novel architecture provides good performances with a small inference time.**

***Index Terms*—Wildland fires, Forest Fires, Fire detection, Fire segmentation, Deep learning, Computer Vision, YOLOv5, U-Net.**

I. Introdution and Motivation

Fires and specifically forest fires are one of the most dangerous and challenging natural disasters today that can threaten humanity and the environment. Fires that are not controlled can make huge damage to human properties and areas of vegetation. More than 350 million hectares are affected by fires every year worldwide. That is why we should consider it and try to find tools to avoid it. So, we should first detect fire, then we deal with it.

In general, wildfires are detected using traditional sensors such as gas detectors, smoke detectors, flame detectors, and temperature detectors, but these traditional techniques are not efficient in this type of problem. Indeed, they have small coverage areas, and they do not respond in real-time. Fortunately, the coming of new technologies such as image processing and computer vision made it easy for researchers to detect and segment fires with high accuracy, high coverage area, and less error. The process is done using camera sensors.

Through the years, researchers have found a lot of techniques that allowed them to detect and segment fires with high accuracy using image processing and computer vision methods. As fire is distinguished by its color, the color analysis technique has been widely used. This technique transforms the image into another color space, such as YCbCr[1] or YUV[2], and then classifies its pixels into fire or non-fire based on comparing pixel values to some thresholds. To segment fire, many techniques based on Deep Learning have been used such as U-net and they have given a good performance. Deep Learning techniques helped researchers a lot to extract relevant features that best represent the problem. Indeed, Neural Networks have been successfully used in several fields such as image classification, speech recognition, face recognition, self-driving cars, cancer detection, etc. For all these applications, deep learning proved its efficiency in detecting and segmenting different classes of objects. For the task of detection, YOLOV5 has Proved an excellent tradeoff between accuracy and inference time. For the task of segmentation and specifically the segmentation of fire, U-net has given excellent results and performance on segmenting fires.

In this context, we present in this project a fire segmentation system based on deep learning using a trained YOLOV5 model and a trained U-net model concatenated sequentially. For this purpose, we first feed the original images of fires with their annotations to the YOLOV5 model then we crop the fire class using the bounding boxes obtained via the detection. Finally, we feed these cropped images to a trained U-net model using original images with their annotations and we obtain our segmented images with their bounding lines in the same image. More specifically, this paper makes two main contributions. Firstly, We propose a novel architecture capable of detecting and segmenting fires in an operationally and time-efficient manner aiming to overcome the limitations of state-of-the-art techniques. Secondly, Our model has high performance on high and small-size fire objects and its capability to distinguish between fire and fire-like objects. The remainder of the paper is organized as follows: section 2, briefly describes the related work of Deep Learning techniques for fire segmentation. Section 3 describes the proposed deep learning architectures. In section 4, the implementation and the experimental results are presented. Finally, section 5 concludes the paper.

II. Related Work

In this related work review, we first deal with Faster ***EfficientDet*** architecture for object and fire detection, then, we introduce Retina for fire detection. After we put the emphasis on 2 newly designed architectures that have been successfully used for object segmentation, we have selected them, and deployed them for fire segmentation: the U-net architecture and the FCN architecture.

A. Object Detection Techniques : Faster R-CNN

The first detector tested for the detection of FIRES is the algorithm created by S. Ren[3] et al which relies on detection

entirely made with networks of convolutional neurons (figure 1):

- A first convolutional neural network takes as input an image of any size and gives as output the regions in which the objects to be detected could be located.
- The second network takes as input the regions proposed by the first network and searches if they contain the object to be detected.

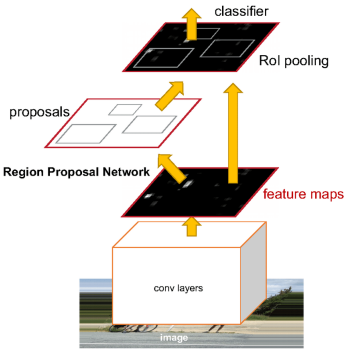

Fig. 1: Faster R-CNN Architecture

B. Image Segmentation Techniques : Fully Convolutional Network (FCN)

The definition of FCN is similar to U-net with a slight difference that is upsampling method. In FCN, the same level downsampling feature map and upsampled feature map are simply added and upsampled right away. On the other hand, in U-Net, it is concatenated and then goes through some conv layers for additional processing[4].

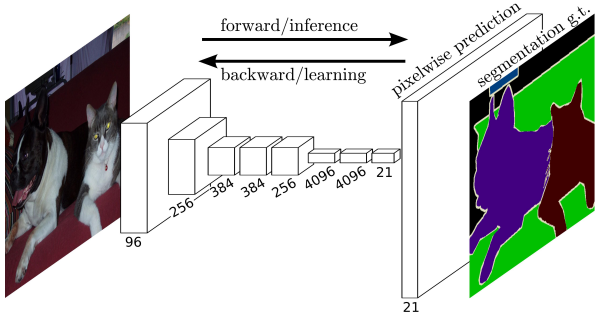

Fig. 2: FCN Architecture

In this section we have detailed the two architectures that are related to our fire detection and segmentation work, in the next section, we provide a detailed architecture of our original technique.

III. Proposed Method

In this section, we will detail our proposed architecture that will take an original image as input and provide us with its mask as output.

A. Proposed Architecture

The proposed fire segmentation architecture is based on combining two convolutional neural networks models in order to perform the segmentation of wildfires. The first model is YOLOv5 and the second one is U-net. In the proposed architecture, these two convolutional neural networks are integrated. First, the network is fed with RGB color images of forest fires, that are processed by YOLO to get the bounding box around the fire area[5]. Once we get these, a Crop layer is applied to the image obtained from YOLOv5 results so that we get only the part limited by the bounding box. Then, these cropped images are fed into the U-net. The result is a binary mask representing the fire pixels in the image.Finally, we took the bounding lines obtained and we put on the original images. It is important to mention that we trained U-net offline and then used the trained weights to segment the cropped images. The model is presented in Figure 3.

Fig. 3: YOLOV5 with Mask Architecture

B. YOLOv5

YOLOv5 (You Only Look Once) is a single-stage object detector that has three important parts :

- Model Backbone: This part is used to extract important features from the given input image. The cross Stage Partial Networks are uses as a backbone to extract rich informative features from the input image
- Model Neck: It mainly consists of generating feature pyramids. Feature pyramids are used to generalize well on object scaling. It helps to identify the same object with different scales. This is useful for performing well on unseen data
- Model head: This is the final detection part. YOLO applies anchor boxes on features and generates final output bounding boxes with a class score.

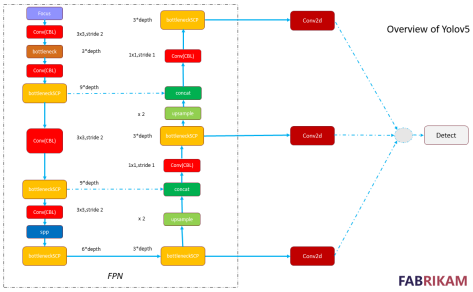

Fig. 4: U-net Architecture

C. U-NET

U-Net[6] network is a deep convolutional network that has been successfully used in medical image segmentation. Unlike

traditional deep learning models which are data-hungry, U-Net can still be trained with a small amount of data. The U-Net can be seen as a two-stage algorithm. Each stage with four blocks and one shared block. Each block is built of two consecutive 2D convolution layers. The main difference between the two stages is the transition between blocks. 2D max-pooling is used for the first stage transitions and 2D transpose convolution is used for the second stage. The max-pooling causes the size of the image to decrease by a factor of 1/4. The number of filters used in the convolutions double compared to that of the previous block. Whereas the transpose convolution has the opposite effect. The size of the image quadruples and the number of filters is halved. In total, the image is reduced four times and then is expanded back to its original size. After the images have passed through the network, one last convolution is performed, and a sigmoid activation function is used. A 1x1 kernel produces our binary mask, a result of the segmentation process. U-Net uses a set of input images and their corresponding binary masks. During training and based on the desired output (the mask), the network learns how to segment the pixels of the images into the different class labels. In our work, we use 2 classes (fire and non-fire).

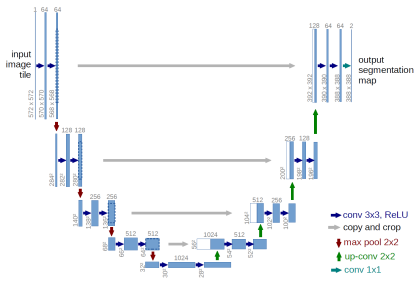

Fig. 5: YOLOV5 Architecture

In this section, we explained the YOLOV5 and U-net architectures and the way they will be concatenated sequentially to form our final architecture.

IV. Implementation and Results

In this section, we detail the implementation settings we followed to train and test our proposed techniques. Namely, the data preparation step (dataset and data augmentation), then finally the overall training materials used in our framework.

A. Dataset and Annotations

1) Dataset: Corsican fire database is the dataset of wildfire images collected from different research teams in the world. It includes wildfire image sequences acquired in different areas, under numerous conditions like burning vegetation type, distance to fire, the brightness of fire, and climatic conditions. To diversify our dataset and make improve the model capability of distinguishing between fire and fire-like objects, we added to the Corsican Fire database numerous images that include fire-like colored objects in various resolutions such as scenarios such as lights, sunrise, sunset, and fire fighter's clothing. At all, the dataset contains about 1300 images. They include fire, non-fire, and fire-like images with different resolutions.

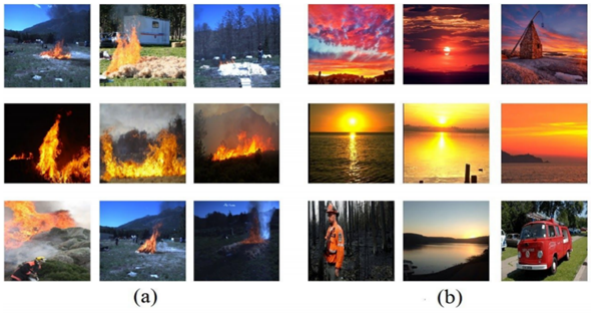

Fig. 6: Examples from the Dataset: (a) Forest Fire Images (b) Fire-like Images

2) Annotations: For the detection dataset, each image is associated with a txt file corresponding to the ground truth class and the bounding boxes.Precisely, each annotation contains the object class and its coordinates in this format: <object-class> <x> <y> <width> <height> where:

- <object-class> - 0 or 1 since we have only two classes (Fire, not fire)
- <x> <y> <width> <height> - float values relative to the center of rectangle (x,y) and the width and the height of image. Value range from [0.0 to 1.0]

For the segmentation dataset, each image is associated with a binary mask corresponding to a segmentation ground-truth (area of fire pixels). A binary mask is an image with only Black and White : Black corresponds to the pixels of non-fire and white corresponds to the pixels of fire.

Fig. 7: Binary Mask Example

B. Data Augmentation

It is important to use data augmentation techniques if want to improve the performance of our model and avoid overfitting. Mainly, it consists of applying transformations on the image such as geometrical transformations (rotation, scaling, padding, and flip translation) and photometric transformation (brightness, contrast, and shear)[7] :

- Flipping: The main applications have three horizontal flips, vertical flips, and horizontal flips.
- Cropping: makes the network less sensitive to scales, so it can identify smaller objects.
- Image translation: Consists of translating the object present in the image.
- MixUp: convex overlaying of image pairs and their labels.
- Mosaic: Mosaic data augmentation combines 4 training images into one in certain ratios.

- Rotation, shear, image scale, etc.

As for our problem, in the case of fire, we carefully chose the techniques we should use. For instance, we did not consider using color space Adjustments (HSV augmentation) because we want to keep the fire color information as it is. Besides, we excluded rotation, Shear because for example we cannot find a 90 degree rotated fire in real life. Also, it is reasonable to flip the fire image horizontally, but it would not be reasonable to flip it vertically. As in the real world, we would not be seeing many images of fire flipped upside-down. In conclusion, we used image Translation, image scale, mosaic and mix up, and horizontal flip. Using these data augmentation techniques helped increase the performance.

C. Training

During the implantation, we trained the two models using Pytorch for the detection part and Tensorflow for the segmentation on a machine with GPU NVIDIA Tesla P100 16 GB. Moreover, we divided our dataset into three subsets as presented in this Table.

	Number of positive images	Number of negative images	Number of annotated objects
Training set	883	107	1367
Validation set	100	15	237
Testing set	155	30	221

Fig. 8: Dataset Subsets

1) Detection Training: In YOLOv5, the input data are PNG images and TXT files containing details of annotated objects. Our training was conducted using Binary Cross-Entropy with Logits Loss function from PyTorch for loss calculation of class probability and object score, a learning rate set to 0.01, a batch size of 8, many epochs set to 300 epochs, and an image size set to 416x416. Note that the training time changes depending on the models since we trained the small YOLO and the Large YOLO.

2) Segmentation Training: In U-NET, the input data are PNG images alongside their corresponding binary masks. We used a combination for the loss function of Dice coefficient and Binary Cross entropy as follows :

dicepbce = 1e-3*binarycrossentropy - dicecoef.

We resized the images to 256x256 and we used 10 Epochs of training due to the lack of high resources especially of GPU. We used a learning rate of 1e-4 for the Adam optimizer with a batch size of 4.

D. Metrics

The evaluation metrics used in this work are divided into detection metrics and segmentation metrics.

1) Fire Detection Metrics:

- Precision: the fraction of relevant instances among the retrieved instances; TP/(TP+FP)
- Accuracy: Accuracy is the quintessential classification metric; (TP+TN)/(TP+FP+FN+TN)
- MAP(mean average precision: mAP (mean average precision) is the average of AP, with AP being the area under the precision-recall curve above.

2) Fire Segmentation Metrics:

- Dice Coefficient : The Dice Coefficient is a statistical indicator that measures the similarity of two samples. DSC = 2TP /(2TP + FP + FN)
- Accuracy : (TP+TN)/(TP+FP+FN+TN)

E. Results

1) Detection Results: In this table, we present the performance of our model for both YOLOv5s(Small version) and YOLOv5x (Extra-large version) with and without TTA (Test time augmentation). TTA is simply applying different augmentation techniques to test images, then feed these different transformed images to the trained model and average the results to get the more confident answer.

	TTA	Recall	MAP@0.5
YOLOv5 Small	ON	0.842	0.732
	OFF	0.805	0.686
YOLOv5 Extra large	ON	0.869	0.718
	OFF	0.796	0.654

Fig. 9: Experimental Results

Fig. 10: Visualization of Fire detection

2) Segmentation Results: The table below presents the best scores that we got for our model. Also, the graph illustrates the evolution of the metrics with the number of Epochs.

dice_coef	binary_accuracy
84%	96%

Fig. 11: Experimental Results

We notice that the resulting segmented fire images and their bounding lines are similar to the original images and this shows the efficiency of our new model. Some small holes are labeled as fire, this is particularly visible with images in the 2nd row.

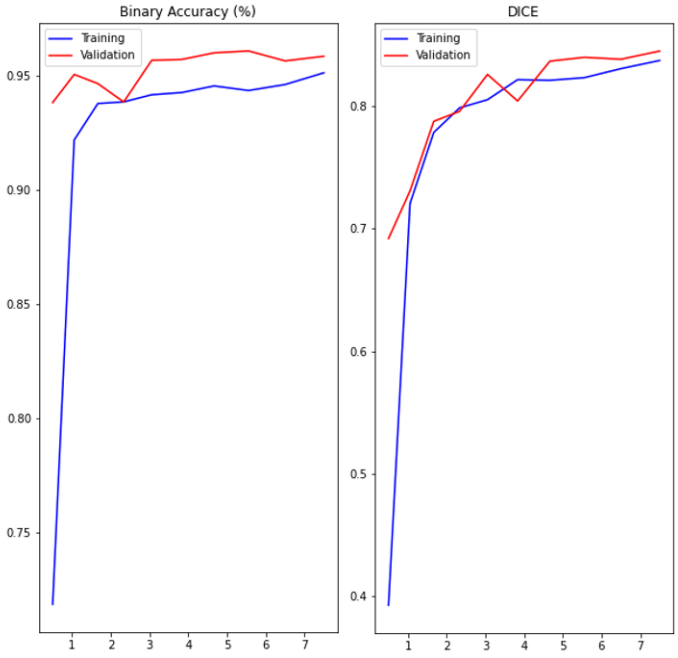

Fig. 12: Metrics Evolution

We can see that our model performs very well in detecting fire pixels and segmenting fire surfaces as it is shown in the figure 13.

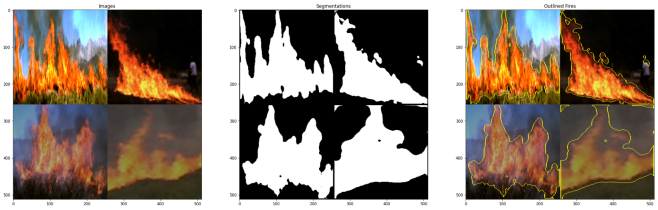

Fig. 13: Final Results

V. Conclusion

In this project we introduced a new technique of Fire detection and segmentation based on YOLOV5 and U-NET. we linked together the good performance of the first method that makes a tradeoff between accuracy and inference time, and the performance of the second method which is very robust for segmentation tasks. Moreover, we enhanced the performance by using the bounding boxes obtained from the first method to crop the output images and feeding them after that to our second model. This last tip improves the performance of our new method. In addition, we used a dataset that contains the fire class with another class that contains photos of object objects that are so close to fire and that can be a cause of confusion for the model. using this approach our model will be more efficient than before.

VI. Acknowledgement

I would like to express my gratitude and thanks to the person named below who gave us the opportunity to do this project, Mrs. Wided SOUIDENE, my professor at Ecole Polytechnique de Tunisie who taught us and gave us information about the different Image Processing Methods. I also would like to express my gratitude and thanks to Mr. Rafik GHALI, my supervisor, for all the advice given throughout the project as well as his expertise in the field AI and Computer Vision and for the help and knowledge provided on the different tasks of the project and for the technical skills and tips, he showed us during this project.

References

[1] Turgay Celik and Hasan Demirel, "Fire detection in video sequences using a generic color model," *Fire safety journal*, vol. 44, no. 2, pp. 147–158, 2009.

[2] Giuseppe Marbach, Markus Loepfe, and Thomas Brupbacher, "An image processing technique for fire detection in video images," *Fire safety journal*, vol. 41, no. 4, pp. 285–289, 2006.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[5] Rafik Ghali, Marwa Jmal, Wided Souidene Mseddi, and Rabah Attia, "Recent advances in fire detection and monitoring systems: A review," in *International conference on the Sciences of Electronics, Technologies of Information and Telecommunications*. Springer, 2018, pp. 332–340.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[7] Connor Shorten and Taghi M Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.