Segmenting fire from wilidfire using YOLO and Attention U-net (24-1-R-19)

Prof. Miri Weiss Cohen
Avner Ben Shlomo and Dor Filis

Braude College of Engineering, Karmiel, Israel
Avner.Ben.Shlomo@e.braude.ac.il and Dor.Filis@e.braude.ac.il

Table of Contents

Abstract. Forest fires can be caused by various factors, including extreme weather events or electrical failures, resulting in property damage and loss of human life. Our project aims to explore a method for segmenting live streams or videos captured by drones in real-time with maximum accuracy while maintaining optimal performance for displaying the segmentation results in real-time. We have opted to examine a model comprising YOLOv4 for detecting fires in each frame. YOLO will provide bounding boxes indicating the fire's location, which will then undergo padding. These padded boxes will be inputted into an Attention U-Net for segmentation. Additionally, we will explore the appropriate parameters, such as learning rate, epochs, batch size, and binary cross-entropy loss function, to meet all our requirements. We believe that the model we have studied will aid in identifying large forest fires in real-time, thereby contributing to the prevention of property damage and loss of human life.

Keywords: Wildfire, Real-time segmentation, YOLOv4, Attention U-Net.

1 Introduction

Throughout the world today, fires devastate entire communities, farmlands, and animal habitats due to natural causes. Fires are becoming more frequent as a result of the effects of global warming, which is the consequence of rising temperatures on an annual basis. Fires occur more frequently as temperatures continue to rise each year. These fires can persist for days or even weeks, requiring significant time and resources to contain. An early detection of a wildfire is essential for prompt intervention and the prevention of substantial damage. In particular, Strong winds and dense forest areas contribute to the rapid spread of the fire, making the timely identification of a spreading fire essential.

Fig. 1: Photo: Shutterstock, VanderWolf Images [8]

Fig. 2: Fires in the forests of Siberia / Photo: Julia Petrenko / Greenpeace [3]

The following examples illustrate significant wildfires that have caused extensive damage in the last five years (2019-2023): The Australian bushfires between

2019 and 2020, deemed an ecological disaster, destroyed 186,000 square kilometers of forests, killing 34 people and approximately 3 billion animals. 31 people were killed in wildfires in California, USA, which destroyed 16,000 square kilometers of forests and approximately 10,000 structures [3].

A majority of wildfire detection methods currently rely on human observation or technology. In spite of this, human observation has its limitations, requiring a large workforce to monitor a large forest areas. However, relying only on human observation, even with the aid of surveillance tools like binoculars, may not provide effective coverage of the entire area. Technological surveillance methods relying on sensor systems such as heat, noise, or electrical conductivity may face challenges due to background noise for audio sensors or interference for thermal sensors. Here are some approaches to detecting wildfires. **Sensor and Long-Range Camera Systems :** In order to detect wildfires from a distance, these systems use optical and thermal sensors as well as long-range cameras. They offer rapid and extensive wildfire detection, but their fixed nature limits their range, and they may not function optimally under conditions of poor visibility.
Ground-Based Sensor Systems : Sensors embedded in the ground are used to detect changes in electrical conductivity that are not related to tree falls or damage to power lines. While they are capable of identifying wildfires in remote areas based on ground conditions, they may produce false positives due to factors such as decaying trees or strong winds.
Sound-Based Technologies : By using microphones or sound sensors, these systems can detect sharp noises caused by falling trees or cracking branches. Their ability to identify specific frequencies associated with falling trees or fire noises allows them to provide immediate notification to the user. The effectiveness of these devices may, however, be compromised in noisy environments.
Floating Devices (Balloons or Spheres) : Systems such as these deploy

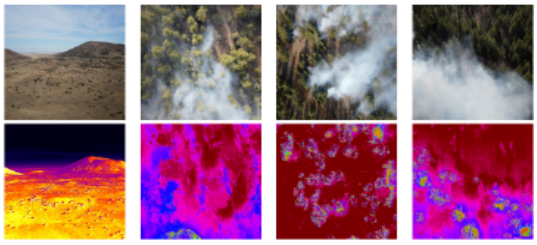

Fig. 3: FLAME 2: Fire detection and modeling - Aerial Multi-spectral image dataset [7]

floating devices equipped with sensors that detect atmospheric changes, such as pollution or smoke. Although they are capable of detecting wildfires quickly in challenging terrain, they are limited in conditions of heavy fog. Obtaining real-time data on wildfires is difficult due to their unpredictable nature. For instance, the FLAME 2 [7] dataset includes videos and images of wildfires captured using two types of cameras: a standard video camera and an infrared camera. This

dual-captured imagery enables effective comparison. The videos are aerial shots taken by drones and come with labeling indicating the presence of fire and smoke. This project proposes a method of detecting wildfires using drones, displaying real-time segmentation on video to mark the location of the fire. To address the wildfire detection challenge, we will demonstrate the accuracy of our model and provide an overview of our deep learning solution. Our solution is expected to aid in early wildfire detection, preventing significant harm to human lives and ecological damage.

Fig. 4: A drone flying over a mountain [1]

2 Literature Review

Wang et al. conducted a study [13] comparing diverse approaches to fire segmentation in forest environments through artificial intelligence, specifically utilizing Convolutional Neural Networks (CNNs). The investigation involved the analysis of images obtained from forested regions, processed through various models to assess their effectiveness in identifying fire segmentation. The researchers employed a substantial image dataset captured by UAVs, primarily drawn from the FLAME dataset by Northern Arizona University, categorizing it into two main image types. The first type, constituting 95.23% of the dataset, featured images of fire within forested settings, while the second type included images with characteristics that could potentially confuse models, such as colors resembling fire or diverse weather conditions. The image analysis process involved several

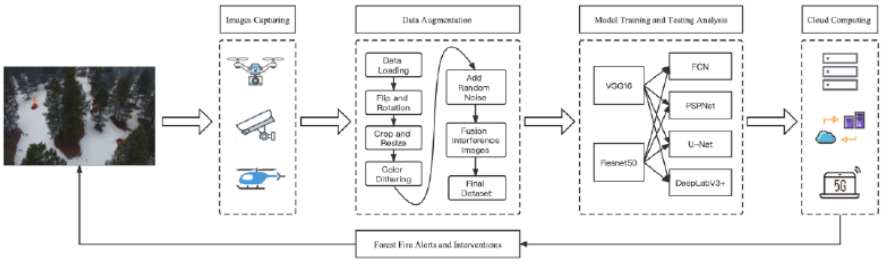

Fig. 5: Overall process of the experiment. [13]

stages like initial UAV photography, data augmentation, and training and testing analysis. Two different backbone networks were used for training the models, and the FCN, U-Net, PSPNet, and DeepLabV3+ models were examined, widely recognized for their effectiveness in image semantic segmentation tasks.

FCN (Fully Convolutional Networks) retains the image structure during processing by using convolutional layers. These layers perform filter computations on the image to identify various features. The model assigns relevant features to each pixel, resulting in a segmentation map.

U-Net is a deep image processing model designed for semantic tasks like object marking or boundary detection. It employs convolutional layers to extract features, followed by dilation layers for high-resolution reconstruction. The model maintains skip connections between encoding and decoding stages to preserve information across resolutions. [13]

PSPNet (Pyramid Scene Parsing Network) excels in semantic tasks for large images. It employs the pyramid structure to focus on features at different scales. The model utilizes Atrous Convolution and Spatial Pyramid Pooling techniques to enhance feature recognition. ASPP layers capture features at various sizes, producing accurate semantic maps.

DeepLabV3+ is a sophisticated image processing model emphasizing semantic tasks. It employs advanced techniques like Atrous Convolution and Spatial Pyramid Pooling to enhance feature recognition. The model uses dilution layers for high-resolution output and combines information from various processing stages for precise results. The study's results indicate that the U-Net model

Mode	Ref	Methodology	Smoke/Flame	Dataset	Accuracy (%)
Classification	Treneska S and Stojkoska B.R. [33]	VGG16	Flame	FLAME: 8617 images	80.76
		VGG19			83.43
		Resnet50			88.01
	Chen Y. et al. [34]	CNN-17	Flame/Smoke	Private: 2100 images	86.00
	Shamsoshoara A. et al. [35]	Xception	Flame	FLAME: 48,010 images	76.23
Segmentation	Bochkov V.S. and Kataeva L.Y. [36]	wUUNet	Flame	Private: 6250 images	94.09
	Harkat H. et al. [37]	DeepLabV3+	Flame/Smoke	Corsian: 1775 images	97.53
	Frizzi S. et al. [38]	U-Net	Flame/Smoke	Private: 366 images	90.20
		CNN based on VGG16			93.40

Fig. 6: Comparison between fire classification and segmentation method. [13]

with Resnet50 achieved the highest accuracy of 99.91% but operated relatively slowly. On the other hand, DeepLabV3+ with Resnet50 achieved a relatively satisfactory accuracy of 99.89% while maintaining faster execution time, making it suitable for real-time applications.

Chou et al. in their study[5], aimed to tackle the challenge of detecting cancer in the gastrointestinal (GI) tract, encompassing the stomach, large bowel, and small bowel. The primary objective was to efficiently trace healthy organs in medical scans, thereby improving cancer treatment precision by focusing on affected areas exclusively. The current practice involves manual scanning and

marking of cancerous lesions for radiation. This research sought to take these scans and implement segmentation techniques to achieve precise marking accuracy.

Two segmentation methods were explored: U-Net and Mask R-CNN. The U-Net model consists of two main parts. The first part, the down-sampling, comprises four stages that increase the image's resolution while extracting its features. The second part, the up-sampling, reconstructs the original image size and presents the segmentation. The Mask R-CNN model operates in two main stages as well. The first stage involves object detection using a method similar to Faster R-CNN, with improved specificity to enable precise semantic segmentation of object boundaries. The second stage is carried out by a convolutional layer that includes upsampling, increasing the spatial resolution of the image and generating the segmentation masks. The research employed the "anonymized MRIs of

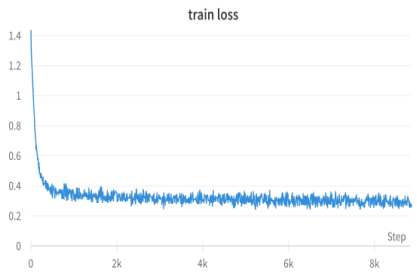

Fig. 7: Training loss [5]

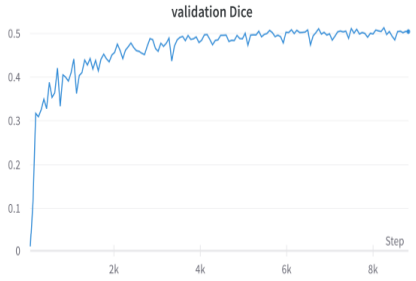

Fig. 8: Validation Dice [5]

patients treated with MRI guided radiotherapy provided by the UW Madison Carbone Cancer Center" dataset available on Kaggle, containing 85 cases with 38,496 organ scans presented in PNG format. After preprocessing, the data was split into 70% training set, 20% validation set, and 10% testing set.

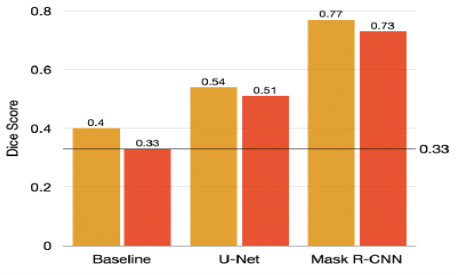

Fig. 9: Dice Score Comparison Across Models [5]

To evaluate the models' results, the Dice score was used to compare the predicted masks against the ground truth. The results indicated that the U-Net model achieved a score of 0.54 on the training set and 0.5134 on the validation set. In contrast, the Mask R-CNN model achieved scores of 0.7732 on the training set and 0.7266 on the validation set. It is evident that the Mask R-CNN model operated with higher accuracy than U-Net. This might be due to the fact that the U-Net's loss function is relatively simplistic compared to the Mask R-CNN.
Muksimova [9], aimed to tackle the challenging task of accurately segmenting forest fires and smoke in real-time images captured by Unmanned Aerial Vehicles (UAVs). To achieve this goal, they employed a sophisticated model known as EfficientNetv2, which is a neural network architecture composed of an encoder-decoder framework, further enhanced by an attention gate mechanism.
EfficientNetv2 is a state-of-the-art deep learning architecture known for its efficiency and effectiveness in various computer vision tasks. It combines the strengths of an encoder, responsible for feature extraction, and a decoder, which refines and reconstructs segmented objects. The attention gate mechanism is a critical component that helps the model focus on relevant information while excluding irrelevant details, thus improving the accuracy of wildfire segmentation.
To train and evaluate their model, the researchers compiled a comprehensive dataset consisting of 37,526 images depicting wildfires and smoke. These images were collected from various online sources, including YouTube videos, and are publicly accessible via the following link: https://github.com/ShakhnozaSh/Wildfire-NET. The results of the research indicated that the proposed EfficientNetv2-

Method	Backbone	Time	FPS	AP^{mask}	$AP50^{mask}$	$AP75^{mask}$	APS^{mask}	APM^{mask}	APL^{mask}
SOLOv1 [47]	Res-101-FPN	43.2	10.4	37.8	59.5	40.4	16.4	40.6	54.2
SOLOv2 [48]	Res-101-FPN	42.1	31.3	38.8	59.9	41.7	16.5	41.7	56.2
Blend Mask [49]	Res-101-FPN	72.5	25	38.4	60.7	41.3	18.2	41.2	53.3
Retina Mask [50]	Res-101-FPN	166.7	6.0	34.7	55.4	36.9	14.3	36.7	50.5
FCIS [51]	Res-101-C5	151.5	6.7	29.5	51.5	30.2	8.0	31.0	49.7
MS R-CNN [52]	Res-101-FPN	116.3	8.6	38.3	58.8	41.5	17.8	40.4	54.4
YOLACT- 550 [46]	Res-101-FPN	29.8	33.5	29.8	48.5	31.2	9.9	31.3	47.7
Mask R-CNN [38]	Res-101-FPN	116.3	8.6	35.7	58.0	37.8	15.5	38.1	52.4
PA-Net [53]	Res-101-FPN	212.8	4.7	36.6	58.0	39.3	16.3	38.1	53.1
YOLACT++ [54]	Res-101-FPN	36.7	27.3	34.6	53.8	36.9	11.9	36.8	55.1
Proposed method	Res-101-FPN	26.2	33.9	39.4	63.2	40.5	16.3	42.8	56.1

Fig. 10: Quantitative comparison of the proposed method with existing methods in terms of accuracy and runtime. [9]

based model achieved superior accuracy and reliability in the identification and segmentation of wildfires and smoke. Furthermore, the model's ability to operate in real-time offers significant potential for enhancing wildfire monitoring and detection efforts, ultimately contributing to the mitigation of environmental damage caused by wildfires.
Ghali [6] explores techniques for the classification, detection, and segmentation of wildfires in nature. These techniques are implemented using deep learning models. The main goal of these models was to identify and detect wildfires and

perform segmentation of fire areas. The key challenges they addressed were complex backgrounds and smoke coverage in some fire areas. The article presented several commonly used datasets for tasks of wildfire detection and classification, including BowFire, FLAME, CorsicanFire, FD-dataset, ForestryImages, VisiFire, Firesense, MIVIA, FiSmo, DeepFire, FIRE, and FLAME2. Each of these datasets can contain images or videos of fires in urban or natural settings. The article then discussed image processing and the different models' capabilities to identify wildfires. Most of the models discussed in the article achieved an accuracy of around 90percent. The paper also addressed the real-time wildfire detection challenge. Existing models were divided into three categories: Wildfire Classification, Wildfire Detection, and Wildfire Segmentation. Each category was evaluated using different hardware, measuring the frames per second (FPS) the model could process.
For Wildfire Classification, the models ranged from FT-ResNet50 with 18.1 FPS as the lower threshold to EfficientNet-B5 and DenseNet201 with 55.55 FPS as the upper limit. In Wildfire Detection, the lower threshold was set at Modified Yolo v3 with 3.2 FPS, and the upper limit was FCDM with 64 FPS. In Wildfire Segmentation, the lower threshold was MedT with 0.37, and the upper limit was wUUNet with 63 FPS. The results presented in the article demonstrate the reliability and robustness of these models, showing their high potential for performing these tasks. The aim of the work by Chen [4] is to solve the problem of forest fire detection and traredcking using drones that capture images or videos from two cameras. The first is a regular RGB camera, while the second is an infrared

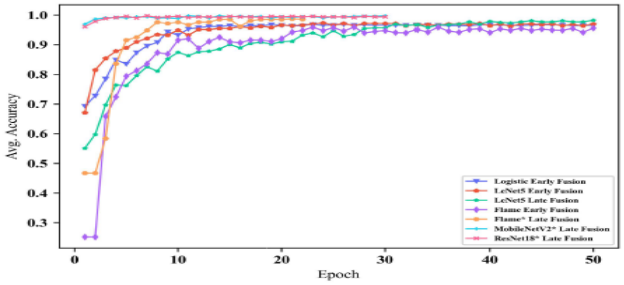

Fig. 11: Averaged accuracy via time on test set of different models with input of RGB-thermal pairs. TThe models with the sign '*' denote the fine-tuned pretrained models, while the remaining models denote the models training from scratch. [4]

camera that shows a thermal image highlighting variations in heat displayed in the picture. The dataset used in the study is known as FLAME2, containing both regular and thermal images. The dataset was divided into four categories: images containing both fire and smoke (25,434), smoke without fire (0), fire without smoke (14,317), and neither fire nor smoke (13,700), totaling 53,451 pairs of images with both regular and thermal content. The model used in the article is a model that takes either a thermal or regular image and passes through

a convolutional layer, followed by a BatchNorm+ReLU layer, then splits into two separate convolution layers. One of them undergoes two more iterations in the BatchNorm+ReLU layer and the separate convolution layer, finally merging with the second convolution layer. The BatchNorm+ReLU layers and convolution are passed through once, finally ending with a global pooling layer, dropout, and softmax.

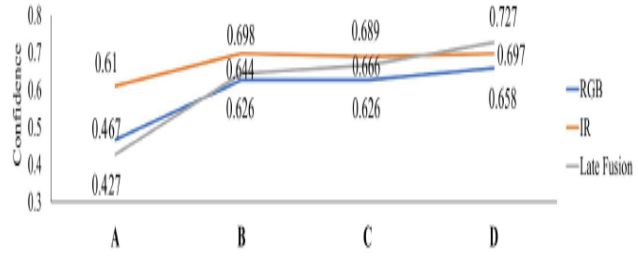

Fig. 12: Detection confidence for sequential frames by pre-trained ResNet18 with RGB, IR, and Late Fusion, respectively. All of them are detected as YN class. The confidence is calculated as the softmax probability of the predicted class. [4]

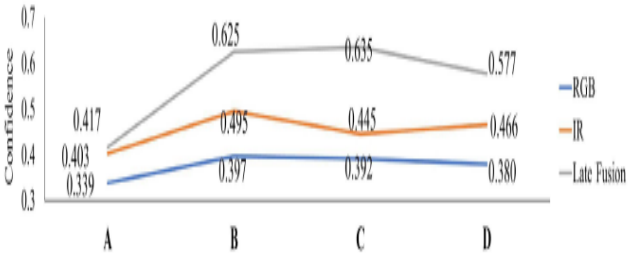

Fig. 13: Detection confidence under Gaussian noise for sequential frames by pre-trained ResNet18 with RGB, IR, and Late Fusion, respectively. All of them are detected as YN class. The confidence is calculated as the softmax probability of the predicted class. [4]

The data was split into 80% for training and 20% for testing. The model's accuracy, compared to the labeling of two experts, is 94%. This study represents a novel development in creating a real-time fire detection and tracking method.

3 Background

In this work, our goal is to identify and precisely locate wildfires using drones equipped with cameras. Employing segmentation techniques, our objective is to accurately detect the affected areas. To achieve this, we will utilize well-known models such as YOLOv4 and Attention U-Net. Moving forward, we will provide a detailed overview of these models and their functionalities.

3.1 YOLOv4 (You Only Look Once)

Object detection is an important task in computer vision, addressing the challenge of precisely identifying and locating objects of interest in images or videos and label them with an appropriate class label. The object detection model learns to identify and locate objects by minimizing a loss function that measures the difference between predicted and ground-truth labels and bounding boxes. These models are used in applications, such as autonomous driving, surveillance, and robotics.
In recent years, there has been significant progress in the field of object detection. object detectors are classified into two categories viz. two stage and single stage object detectors. The two stage approach in her first step predict the bounding boxes and in second stage predict the class and bounding box information. This approach is slower than other detectors but has high accuracy. One-stage object detection in one forward pass predict everything, These detectors are much faster than two-stage.
YOLO is the state of the art object detection model that utilizes deep learning, initially developed by Joseph Redmon and Ali Farhadi in 2015. Since its inception, there have been eight different versions, with the latest being YOLO V8 released in 2023. The main differences between the versions are development of new features in order to improve runtime efficiency.
YOLO, short for "You Only Look Once," is a convolutional neural network that not only predicts bounding boxes but also calculates class probabilities for all objects present within an image. As this algorithm identifies the objects and their positioning with the help of bounding boxes by looking at the image only once, hence they have named it as You Only Look Once, reflecting its efficiency, which makes it highly suitable for real-time object detection tasks. The main challenge lies in the accurate identification of multiple objects along with their exact positioning present in a single image In Yolo object detection process, the

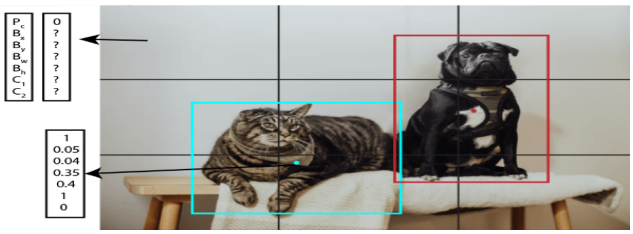

Fig. 14: The vector [14]

image is divided into S × S grid cells, each grid cell predicts B bounding boxes along with their positions and dimensions, probability of the classify. The fundamental concept behind detection of an object by any grid cell is that the center of an object should be inside that grid cell.
It is necessary to resolve the problem of detecting the same object in multiple grid cells or in multiple bounding boxes of the same grid cell. Non max suppression internally uses an important concept of Intersection over Union (IoU) which can be computed for two boxes. First, we select the box having the maximum

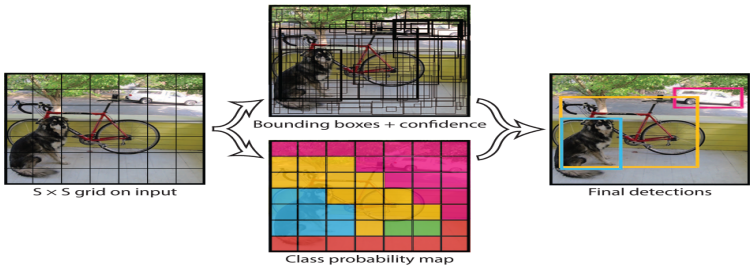

Fig. 15: The Model [11]

class score. All other bounding boxes overlapped with the chosen box will be discarded having IoU is greater than some predefined threshold. We repeat these steps until there are no bounding boxes with lower confidence scores than the chosen bounding box. In YOLO v4, the processing begins with the input image, which then goes to feature extraction within the backbone. Following this, the features are passed to the neck, responsible for aggregating them to generate optimal features for predictions. Finally, the processed features are forwarded to the head, where both object detection bounding box prediction and class prediction take place. In YOLO v4, the backbone employed is the CSP Darknet, chosen for its effectiveness in feature extraction. For the neck, a combination of SPP and PAN is utilized to enhance feature aggregation, while the detection head employs the YOLO architecture, ensuring accurate and efficient predictions.
Backbone: CSPNet employs an approach where the input channels are divided into two halves. One half serves as the input to the computation block, while the other half is concatenated with the block's output afterward. This computation block can be a DenseBlock. Unlike traditional architectures, where layers are connected sequentially, DenseBlock ensures that every layer is connected to all others. In this setup, half of the features go through the dense block, while the remaining half bypasses it, feeding directly into the transition layer. Notably, in CSPDarknet, the activation function has been replaced with Mish, a novel activation function known for its smoother gradients and improved performance.
Neck: The neck plays an important role in collecting information from different levels of feature maps coming from the backbone and merging them effectively for detection purposes. It's crucial for small object detection, as it combines spatial information required for their accurate identification. As we propagate

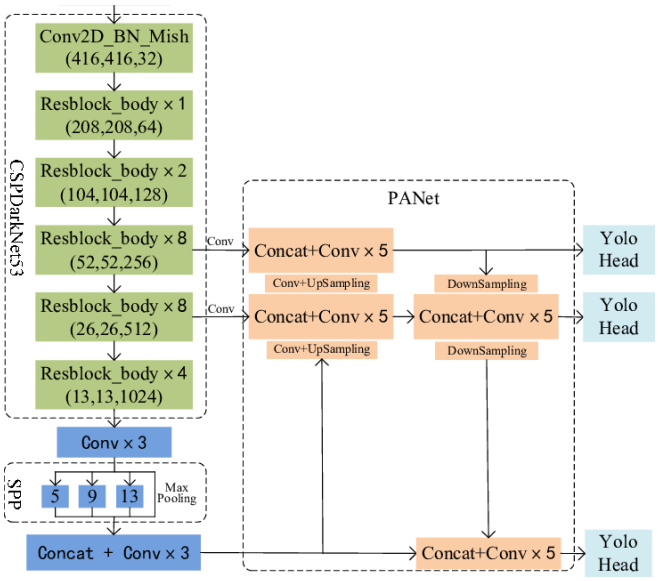

Fig. 16: Architecture of YOLOv4 [2]

through the network and ascend to higher levels, we gain more semantic information, which is also essential for detecting small objects. What FPN (Feature Pyramid Network) accomplishes is to combine the high-level feature maps with lower-level ones, generating a new set of feature maps. PAN (Path Aggregation Network) extends upon FPN, introducing an additional branch of the bottom-up path. This innovation shortens the information path. Additionally, SPP (Spatial Pyramid Pooling) is integrated on top of the backbone to extract more significant context features. In YOLOv4, instead of a polling layer, an SPP layer is employed after the final convolutional layer. This involves dividing the final feature map into a fixed number of cells and performing pooling for each cell, resulting in a fixed-length feature vector.

3.2 Attention U-Net

In order to mark the wildfires in the project, we decided to use the Attention U-Net model to segment the wildfires. The U-Net model is well-known for its excellent segmentation results, and when combined with Attention, it performs even better. Now, let's explain what segmentation is, how the U-Net architecture works, and how we can integrate Attention blocks to enhance its performance even further.

Segmentation: There are tasks in deep learning that require us to classify objects based on their context within an image. Segmentation is one method for accomplishing this. In segmentation, an image is divided into regions and colored according to their color. An object is represented by a region, while a class is represented by a color. In our project, we intend to highlight the fire in one color and the forest in another so that the appropriate classification can be displayed.

Pixels with similar features are grouped together by segmentation. We will first pass the image through filters in order to understand its features and then we will enter them into clusters based on the common denominator. As a first step, we pass the image through filters to identify its features, find commonalities, and then classify it according to these features. This presentation will provide an overview of the U-Net model for segmentation and demonstrate how it works.

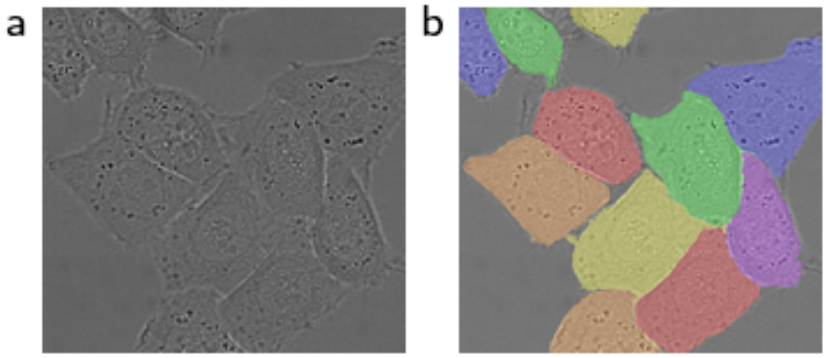

Fig. 17: Segmentation using U-net [12]

U-Net: The U-Net architecture was first introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in 2015 [12]
At first, it was developed to address the segmentation problem in the medical field, but was quickly adopted for a number of other applications. It consists of an encoder, a decoder, and a bottleneck. In the following sections, we will discuss each of these components in greater detail.

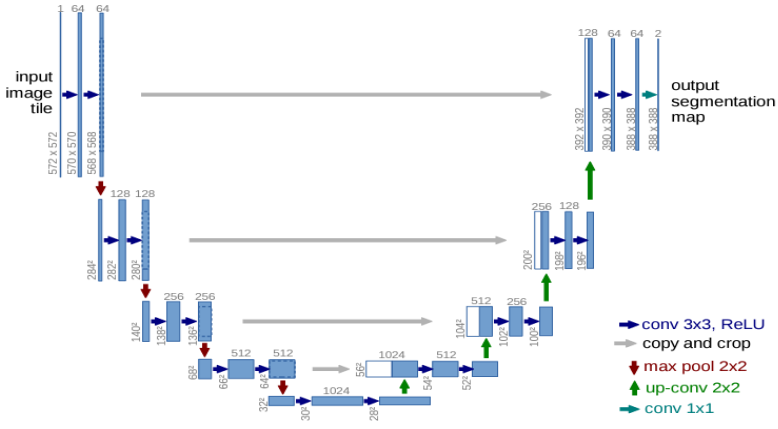

Fig. 18: U-net architecture[12]

The encoder is responsible for extracting the features from the image. In order to accomplish this, it uses downsampling. This can be viewed as a descending process. During each stage, we receive data in the form of dimensions (w,h,d). A convolution of 3X3 with ReLU activation function is performed twice in each such block. Two recipients receive the result. The first is to a skip connection, which we will explain later, and the second is to a max pool of 2X2, which will be the input for the next stage. By reducing the width and length of the image and increasing the depth of the filters in each block, we reduce the dimensions of the image. The U-Net designer can decide how many downsampling blocks he wants to add based on the resolution we enter as our initial input. The number of these blocks was 4 in the original paper that introduced the U-Net, when the input images were (572,572,1) and the output of the last block was (32,32,512). After the downsampling, the bottleneck is actually what connects the upsampling and the downsampling. Convolution of 3X3 with the ReLU activation function is performed twice here, followed by upsampling.
The final step is the decoder, also known as upsampling. At this stage, we also divide into blocks. There are two UPConv 2X2 operations in each block. Essentially, it enlarges the low-resolution image. The second operation is the skip connection. Parallel to each upsampling block, there is a downsampling block with the same dimensions. The actual process involves concatenating all the filters we obtained during downsampling with the UPConv result, thereby doubling our features. It is this characteristic that distinguishes the U-Net from other models. A convolution of 3X3 is then performed twice with an activation function and sent to the next block. As a result of downsampling, the number of blocks will always be symmetrical. Lastly, before we output the image to the output, we add another convolution layer of 1X1. By using this layer, we reduce the depth of the features while maintaining the height and width of the image, and thus we output the image in exactly the same dimensions as the input image.
The architecture has demonstrated good performance since it was presented in 2015, beating the sliding-window convolutional network at the same time with a warping error of 0.0003529 and a random error of 0.0382.

Rank	Group name	Warping Error	Rand Error	Pixel Error
	** human values **	0.000005	0.0021	0.0010
1.	u-net	**0.000353**	0.0382	0.0611
2.	DIVE-SCI	0.000355	0.0305	0.0584
3.	IDSIA [1]	0.000420	0.0504	0.0613
4.	DIVE	0.000430	0.0545	**0.0582**
⋮				
10.	IDSIA-SCI	0.000653	**0.0189**	0.1027

Fig. 19: U-net result [12]

Attention: In essence, attention consists of focusing on what interests us and ignoring what does not. As part of the U-Net training process, attention is used to emphasize relevant activations. Furthermore, it reduces processing time by not referring to irrelevant information and allows the network to be more generalized. Attention can be classified into two types as follows [10]:
The method of hard attention involves dividing the image into smaller portions. Due to the small area covered by the model each time, the model is incapable of differentiating, and this requires strong learning that cannot be achieved using backpropagation. A network can either focus on something or not; there is no middle ground.
In soft attention, areas in the image that are more relevant are given a higher weight than those that are unrelated. Backpropagation is possible due to the weight method. Backpropagation allows us to learn from our network, to update the weights continuously and to become more accurate, allowing us to pay more attention to relevant areas.
So how does the Attention block look? It has 2 inputs: the first is X, the second is G (gating signal). First, we will pass G through a convolution of 1X1 and a certain number of filters that we want to get immediately. We will explain how to determine the number of filters for input X. We will perform a convolution of 2X2 to reduce the dimensions of the length and width of the image by half and leave the number of filters the same. The number of filters of X is the number of filters we want G to have after the convolution. This is because the next step will be to connect them with the weights, which actually helps us to emphasize the large weights. Then we will pass them through the ReLU activation function. Then the result will pass through a convolution layer of 1X1 with a number of filters of 1, which will reduce all the depth of the filters to 1. Then we will pass them through a sigmoid activation function that will limit the values of the weights between 0 and 1. We will perform upsampling to bring the result to the same dimensions as the input X. This is in order to multiply the layers between them, and thus we will get the original dimensions of X and will not interfere with the flow of the network in which we implement the block.
Attention U-Net: Let us now see how we can combine the U-Net architec-

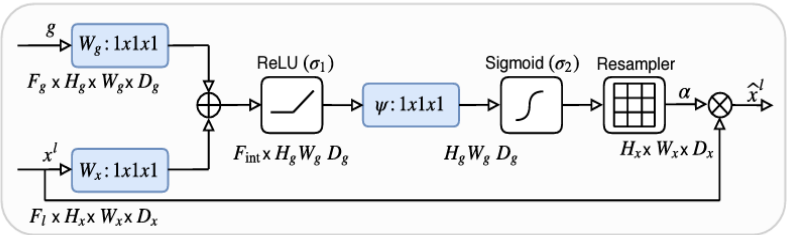

Fig. 20: Attention block [10]

ture and the Attention block. As described above, the skip connections in the original U-Net pass filter results from downsampling to upsampling. Because of

the symmetric structure, after every two convolution layers in downsampling, we receive a result that is sent to the parallel upsampling process. Here, we use the skip connection as input X instead of directly concatenating to the parallel block, and input G comes from one layer below in upsampling since we are doing upsampling to it, and we concatenate the result exactly as we did with the skip connection in the original U-Net. In the paper that published the new architecture in 2018, [10] you can see a significant improvement in the performance of the Attention U-Net when compared to the original U-Net.

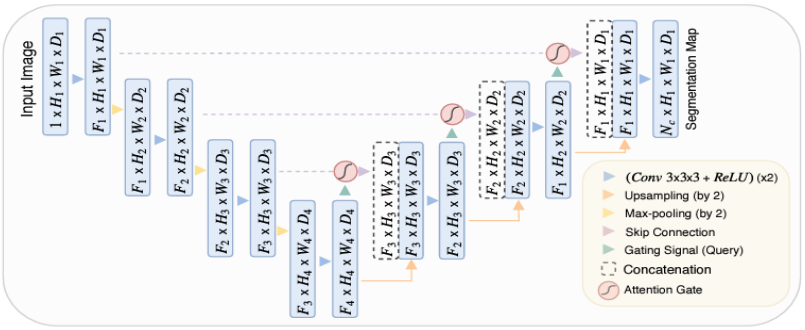

Fig. 21: Attention U-Net architecture[10]

4 Proposed approach

The model we present is designed to detect forest fires in real time. Our research aims for the model's performance to be both fast and accurate in marking fire in video footage efficiently. We will utilize a dataset called FLAME2, containing images and videos captured during a forest fire in Arizona in 2021. Each pair within the dataset consists of an image and video in either RGB or IR format. The dataset is labeled as "Fire/NoFire" to facilitate model evaluation and training. Throughout our research, we will explore various hyperparameters to optimize the model's performance. We will compare training loss versus validation loss, or training accuracy versus validation accuracy over epochs, to ensure accuracy while preventing overfitting. Our model will be constructed by combining two established architectures. Firstly, we will use YOLOv4 to swiftly identify fire locations within images. Subsequently, we will transfer the identified regions, termed as the "head" of YOLOv4, to the Attention U-Net model for segmentation. This approach ensures both speed and accuracy in real-time fire detection.

4.1 Model architecture

The architecture of our proposed model consists of integrating the YOLOv4 framework to identify fire, followed by segmenting the result using the Attention U-Net, as explained earlier in our research. YOLOv4 excels at swiftly and

efficiently detecting objects in images, requiring no internal architectural modifications. We will input video frames as received from a drone. The output of YOLOv4 provides a vector containing parameters defining a bounding box around detected objects, along with confidence scores. We will assess the model's detection performance and establish a threshold percentage for detection, beyond which we transfer the identified object to the Attention U-Net model. Each YOLOv4 detection will yield a bounding box, which we'll extract from the original image and pass to the subsequent model. Given potential variations in box sizes, we'll standardize boundaries for input into Attention U-Net. Regarding the Attention U-Net, we'll maintain its structure, featuring three downsampling blocks, a bottleneck block, and three upsampling blocks. For each upsampling block, we'll incorporate an Attention block, merge its output with the model's, restore it to the original frame, and deliver the segmented image to the user. We anticipate that this model will provide highly accurate results with swift identification and segmentation, rendering it suitable for real-time systems.

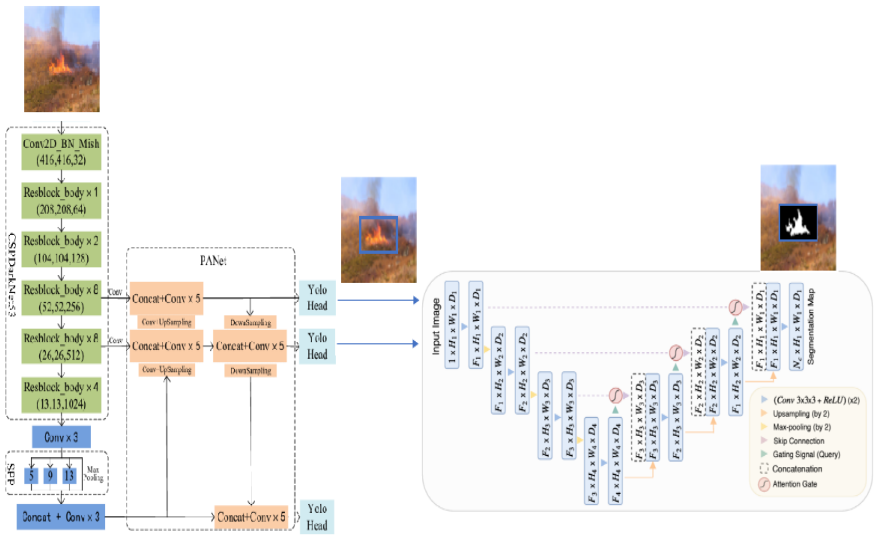

Fig. 22: Proposed Model

4.2 Hyper parameter

To train our model effectively, we need to define and optimize hyperparameters, which significantly influence both the accuracy and speed of the model. These parameters include Learning Rate, Epochs, Batch Size, Loss Function, and Evaluation Metric. Throughout our research, we'll explore and set values for these

parameters to align with our desired results and performance goals. Adjusting these parameters will also help prevent overfitting, ensuring the model generalizes well to new fire detection scenarios beyond the training dataset.For each parameter, we'll explore a range of values and select those that result in the best performance according to our evaluation metrics. By systematically adjusting these hyperparameters, we aim to optimize the model's accuracy, speed, and generalization ability for real-time fire detection.

Learning Rate is a crucial hyperparameter that determines the step size taken during the optimization process. A lower learning rate increases the likelihood of reaching the optimal point but slows down progress, while a higher learning rate speeds up training but risks overshooting the optimum.

Finding the right balance is essential, ensuring efficient progress without sacrificing accuracy. In our case, we'll set the learning rate within the range of $10^{-6} \leq x \leq 10^{-5}$, considering the optimization function ADAM.

This range allows us to explore both slower and faster learning rates, facilitating the discovery of an optimal value. By experimenting within this range, we aim to strike a balance between training speed and accuracy, ultimately guiding the model towards effective real-time fire detection.

Epochs represents a complete pass of the entire training dataset through the model during the training phase. It's crucial to determine the number of epochs for training, balancing between underfitting and overfitting.

An undertrained model may result in inaccurate predictions and suboptimal performance if there are too few epochs. Conversely, an excessively high number of epochs can cause overfitting, where the model memorizes the training data too well and fails to generalize to new data.

In our research, we'll explore the model's behavior across a range of epochs, specifically from 50 to 80. After each set number of epochs, we'll evaluate the model's performance and analyze its learning progress. By monitoring metrics such as training and validation loss, as well as accuracy, we'll assess how well the model is learning and whether additional epochs are beneficial.

This iterative approach allows us to determine the optimal balance between training efficiency and model generalization. Based on the observed results, we'll decide the final number of epochs that best aligns with our goals for accurate and efficient real-time fire detection.

Batch size is a critical parameter that influences how many samples are processed in each iteration during model training. A smaller batch size leads to more frequent weight updates but may introduce instability, while a larger batch size results in slower convergence but greater stability.

In our research, we'll explore batch sizes of 32, 64, and 128. Each option presents trade-offs between training speed and stability. Additionally, we'll leverage cloud technology to harness advanced hardware capabilities, enabling efficient computation without the need for physical infrastructure. This approach ensures we can effectively train our model while optimizing resource utilization and computational efficiency.

Loss functions: Dice Score is a metric that shows us how similar two datasets

are. For example, in our project, we want to determine how closely the segmentation provided by our model matches the actual data mask. The Dice Score ranges from 0 (indicating no overlap or proximity) to 1 (representing perfect overlap). We can calculate the Dice Score using the following formula:

$$DS = \frac{2 \times \text{number of common elements}}{\text{number of elements in set A} + \text{number of elements in set B}}$$

For our project, we have chosen the Binary Cross-Entropy loss function. It helps the model recognize its mistakes by penalizing incorrect predictions. The Binary Cross-Entropy function takes two parameters. The first parameter, Y, represents the class number. Since it is binary, Y can only be 0 or 1. The second parameter, P, represents the probability that we are certain the prediction belongs to a particular class. We feed this probability into the logarithm function, which returns a value between 0 and 1. If the value is close to 0, it indicates a good prediction and the model should not be penalized. Conversely, if the value is close to 1, it suggests a poor prediction and the model should be penalized. The negative sign before the logarithm is used to ensure the result is positive, as the logarithm of a number between 0 and 1 would normally yield a negative value.

$$\text{BCE}(y, p) = -\left(y \log(p) + (1 - y) \log(1 - p)\right)$$

4.3 GUI

Our interface features several options, as depicted in Fig.23. Initially, users can choose to load a photo or video, or connect to a live camera feed. Upon selecting the desired option, they can click the "Start" button to initiate the model's analysis of the frames or the uploaded media. An additional option allows users to toggle segmentation on or off. When segmentation is deactivated, users can view the original, real image without any overlays. In addition, our interface features a progress bar that represents the model's confidence level in identifying the fire. The advancement of the bar reflects the certainty of the detected fire, with a higher progression indicating greater confidence in the detection. This visual indicator offers users real-time feedback on the model's confidence, enabling quick and informed decision-making regarding the fire's intensity.

5 Expected Achievements

In this project, we will develop a model designed to efficiently and rapidly detect wildfires as they occur. Our target is to achieve an accuracy between 95 and 98 percent, coupled with rapid detection to analyze real-time photographs. To evaluate our model, we will compare its performance with Attention U-Net, focusing

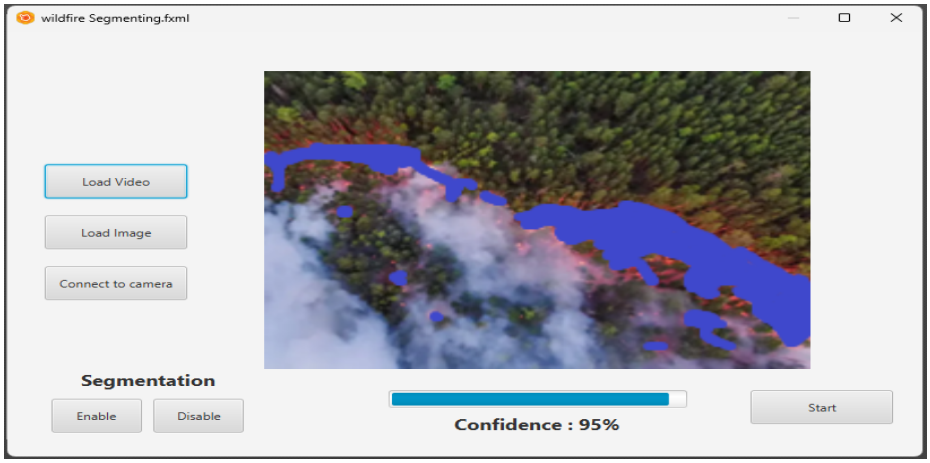

Fig. 23: GUI

on both accuracy and speed. We will fine-tune parameters such as learning rate, epochs, loss function, and batch size based on the results we obtain. Our assessment will be conducted using a dataset comprising video images, allowing us to assess the model's effectiveness in detecting wildfires.

6 UML

We've opted to showcase the project through two diagrams: the first being the Use Case diagram delineating the project's processes, and the second being the Flow Chart.

Use case

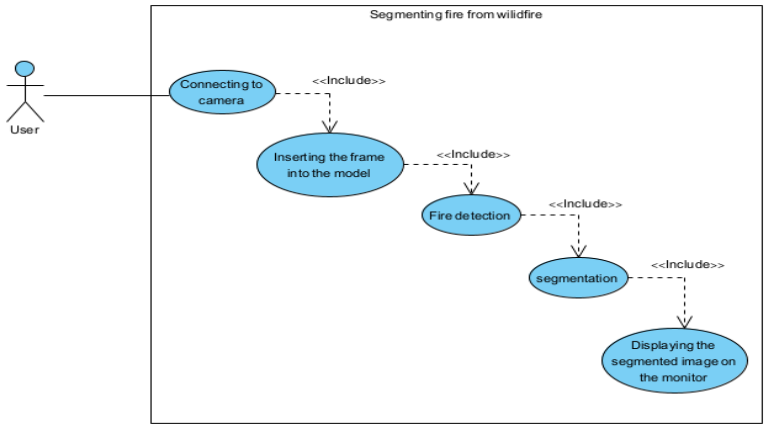

Fig. 24: Use case

Flow Chart

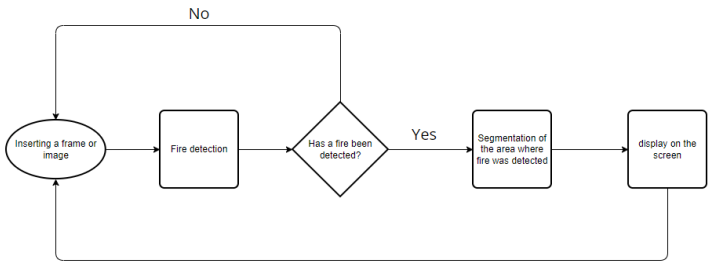

Fig. 25: Flow Chart

7 Evaluation / Verification Plan

Case	Test Case	Expected Result
1	Press Load Video / Image / Camera	The image or video will load to the screen
2	Press Start	The model will segment the image or the frame will display it on the screen, and if it is a video it will segment the next frame
3	Press Enable / Disable	Enable and disable segmentation on the screen
4	Invalid file upload	Notification will pop up asking you to replace a file
5	There is no connection to the camera	Notification will pop up on the screen asking you to connect the camera properly
6	Exiting the program before the end of the video	The program will verify if the user is sure he wants to exit
7	Press Enable / Disable when a file has not yet been uploaded	Error message will appear saying that there is no file loaded into the system
8	Attempt to upload a new file while segmenting in real time	An error message will appear saying that segmentation is in progress and a file should be uploaded only when finished

Table 1: Test Cases and Expected Results

References

1. The best forestry drones for forest management (2021), `https://www.irisonboard.com/best-drones-for-forest-management/`
2. , M., Ji, K., Xiong, B., Zhang, L., Sijia, F., Kuang, G.: Light-yolov4: An edge-device oriented target detection method for remote sensing images. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **PP**, 1–1 (10 2021). `https://doi.org/10.1109/JSTARS.2021.3120009`

3. Ashkenazi, S.: Not only the corona virus: ten climatic disasters in 2020 that will be hard to forget (2021), https://www.globes.co.il/news/article.aspx?did=1001355530
4. Chen, X., Hopkins, B., Wang, H., O'Neill, L., Afghah, F., Razi, A., Fulé, P., Coen, J., Rowell, E., Watts, A.: Wildland fire detection and monitoring using a drone-collected rgb/ir image dataset. IEEE Access **10**, 121301–121317 (2022)
5. Chou, A., Li, W., Roman, E.: Gi tract image segmentation with u-net and mask r-cnn. Image Segmentation with U-Net and Mask R-CNN. Available online: http://cs231n. stanford. edu/reports/2022/pdfs/164. pdf (accessed on 4 June 2023) (2022)
6. Ghali, R., Akhloufi, M.A.: Deep learning approaches for wildland fires remote sensing: Classification, detection, and segmentation. Remote Sensing **15**(7), 1821 (2023)
7. Hopkins, B., O'Neill, L., Afghah, F., Razi, A., Rowell, E., Watts, A., Fule, P., Coen, J.: Flame 2: Fire detection and modeling: Aerial multi-spectral image dataset (2022). https://doi.org/10.21227/swyw-6j78, https://dx.doi.org/10.21227/swyw-6j78
8. Klein, A.: The 10 biggest natural disasters in the last hundred (and a little) years (2022), https://www.globes.co.il/news/sparticle.aspx?did=1001416041
9. Muksimova, S., Mardieva, S., Cho, Y.I.: Deep encoder–decoder network-based wildfire segmentation using drone images in real-time. Remote Sensing **14**(24), 6302 (2022)
10. Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., et al.: Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:1804.03999 (2018)
11. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
12. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. pp. 234–241. Springer (2015)
13. Wang, Z., Peng, T., Lu, Z.: Comparative research on forest fire image segmentation algorithms based on fully convolutional neural networks. Forests **13**(7), 1133 (2022)
14. Zvornicanin, E.: What is yolo algorithm? (2023), https://www.baeldung.com/cs/yolo-algorithm