University of Salerno

ON FIRE PROJECT

GROUP 36
Adinolfi Teodoro
Amato Emilio
Bove Antonio

PROBLEM DESCRIPTION

Fire detection is an interesting challenge due to the **dynamic nature of fire and smoke;** this task isn't easy at times even for an human beeing

- Smoke can have any **form** or **colours**
- We can see **flames and smoke** or **only smoke**
- **Smoke** and **clouds** are similar
- For a camera **light and fire could be similar**.

INITIAL APPROACHES

The approaches initially proposed involved the use of both single convolutional networks (CNNs), such as

- EfficientNet
- ShuffleNetV2
- ResNet
- MobileNetV3

but also hybrid networks composed of the last presented, in combination with recurrent networks (CNNs + RNNs).

CNN

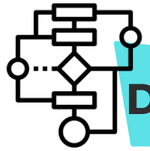
Convolutional Neural Networks are powerfull tools for **image classification.**

Dataset

- OnFire training-set
- OnFire training-set with additional images

Training

Trained with varius configuration of hyperparameters and with different optimizers



Detection

N consecutive fire detections

CNN & RNN

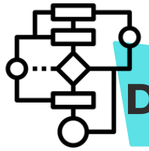
Convolutional Neural Networks can obtain spatial information from each frame, Recurrent Neural network could get dynamic information such as smoke and fire pattern

Dataset

- OnFire training-set

Training

Trained with varius configuration of hyperparameters and with different optimizers



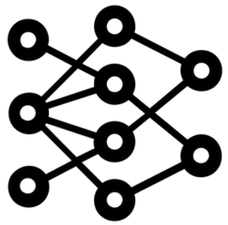
Detection

N consecutive fire detection

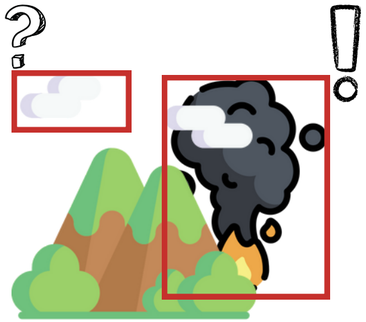
FINAL APPROACH

OBJECT DETECTION + CLASSIFICATION

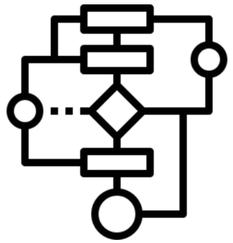
Take a fire picture



The network detects fires



Detection could be exact or bad



With spatial and temporal information we can perform better decision

Fire!

YOLOv5

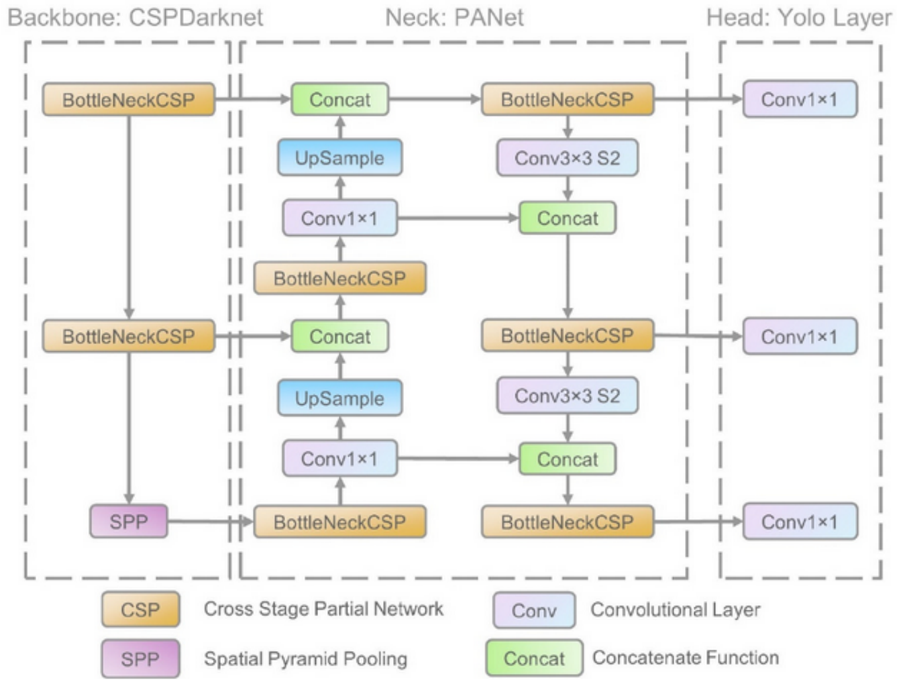
YOLOv5 is a model in the You Only Look Once (YOLO) family of computer vision models that uses a single deep convolutional neural network to simultaneosuly predict class probabilities and bounding box positions directly from images in a single pass

YOLOv5

Architecture

YOLOv5 is a single-phase object detector and its architecture consists of three components:

- **Backbone,** to extract rich feature representation for images
- **Neck**, to extract feature pyramids
- **Head**, to perform the final stage operations

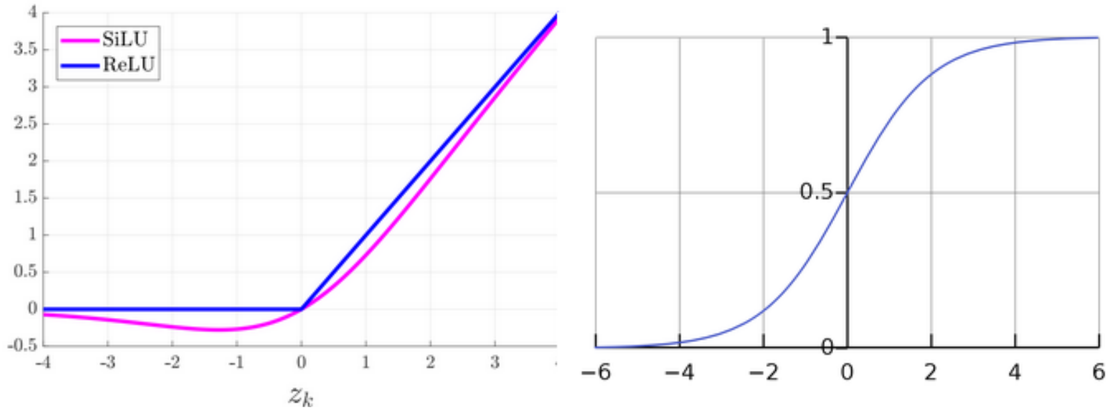


YOLOv5

Activation function

SiLU stands for Sigmoid Linear Unit and,
for YOLOv5 architecture, it has been used
with the convolution operations in the hidden layers.

The **Sigmoid** activation function has been used
with the convolution operations in the output layer.



YOLOv5

Loss function

YOLOv5 returns **three outputs**: the classes of the detected objects, their bounding boxes and the objectness scores.

Thus, it uses **BCE (Binary Cross Entropy)** to compute the classes loss and the objectness loss.

While **CIoU (Complete Intersection over Union)** loss to compute the location loss.

DATASET COLLECTION

We will now analize how we built the dataset used for the training

1 Annotations

YOLOv5 annotations indicate object locations and classes. They're critical for training the model to recognize and position objects accurately.

2 Processing

Image pre-processing is key to ensuring consistency in input data, aiding in accurate feature extraction and improved model performance.

3 Data Augmentation

Data augmentation artificially enhances dataset diversity, improving model robustness by creating varied image versions to train on.

DATASET COLLECTION

OnFire Dataset

Dataset	Fire	Smoke	Fire Smoke	No Fire	Smoke Labels	Fire Labels	Total Images
Train	3192	8187	478	1195	9978	4255	16287
Validation	772	2057	112	294	2444	1011	3235

DATASET COLLECTION

Additional Dataset

Dataset	Fire	Smoke	Fire Smoke	No Fire	Smoke Labels	Fire Labels	Total Images
Train	3270	4071	6722	2239	12707	22122	16302
Validation	836	1015	1688	426	3181	5490	3965

DATASET COLLECTION

Full dataset

Dataset	Fire	Smoke	Fire Smoke	No Fire	Smoke Labels	Fire Labels	Total Images
Train	6462	12258	7200	3434	22685	26377	32589
Validation	1608	3072	1800	720	5625	6501	7200

DATASET COLLECTION

Processing

The 640x640 pixels square format:

- simplify the data pipeline and enhancing GPU utilization
- maintains object aspect ratios, avoiding potential distortions that could impair detection accuracy
- represents a balance between computational efficiency and detection precision, optimizing detail capture and processing speed

DATASET COLLECTION

Data Augmentation

Data augmentation with YOLOv5 enhances dataset diversity and model robustness, addressing object detection challenges.

horizontal flip

mosaic parameter set to 1.0: all training images are amalgamated into a mosaic pattern, teaching the model to recognize objects of various sizes and contexts. These techniques boost the model's generalization on unseen data.

fliplr parameter: images have a 50% chance to be mirrored, teaching the model to recognize objects in different orientations.

TRAINING

The training of the selected model was conducted using the script **train.py**, which implement the entire workflow for model training.

imgsz	batch	epochs	optimizer	lr0	lrf	momentum
640	64	200	SGD	0.01	0.01	0.937

1. **Command-Line Argument Parsing**
2. **Data Loading**
3. **Model Configuration**
4. **Training Phase**

TRAINING

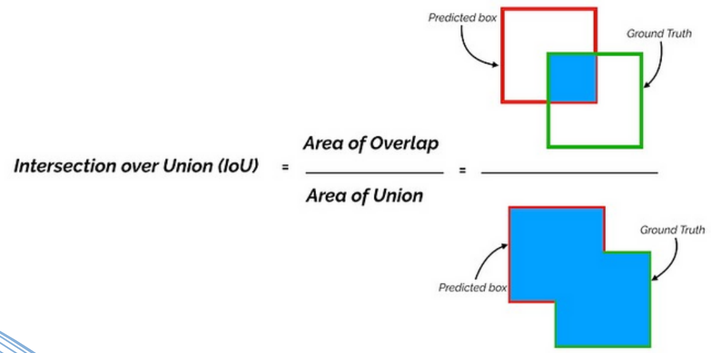
- **Precision**

$$P = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- **Recall**

$$R = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- **mAP_0.5 at IoU**

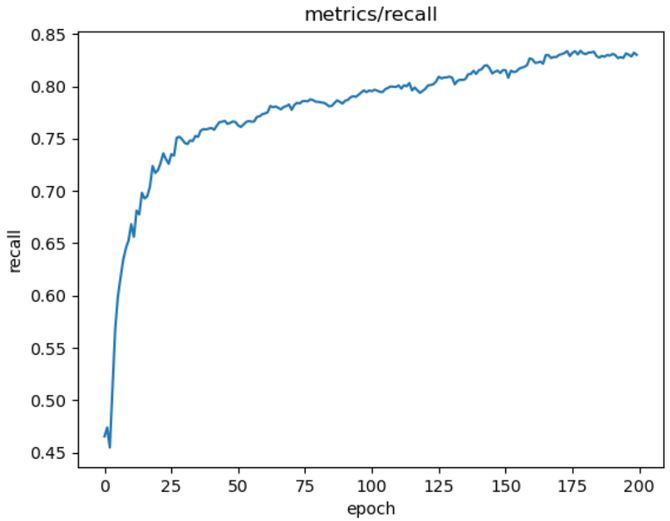


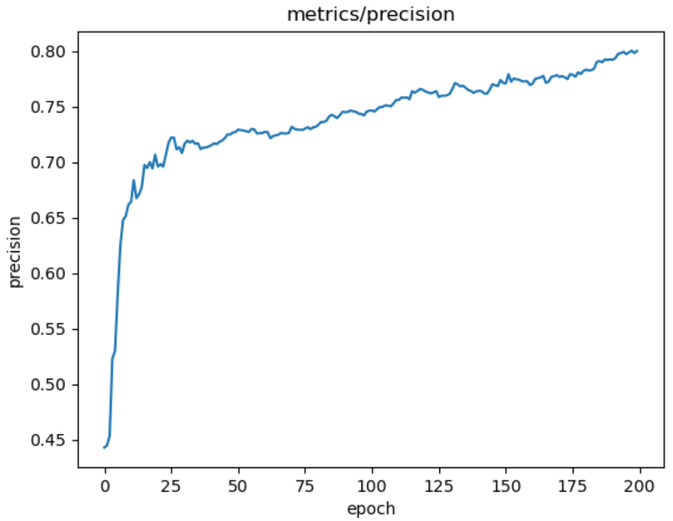
5 **Evaluation Metrics**

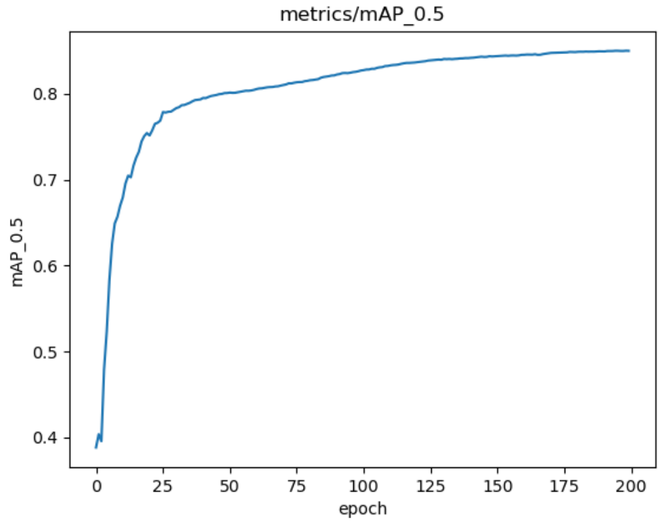
6 **Model Saving**

TRAINING

Results







Overall, the three graphs indicate a **good balance** between **precision** and **recall**, and an **increasing mAP_0.5**, confirming the **efficacy of the model**.

DETECTION ALGORITHM

Fire Threshold Window

Each video can have a fixed FPS...

- **Idea**

  The model **must** see continusley fire for at least 0.5 sec

- **Implementation**

$$\text{Fire Threshold} = \lfloor \text{FIRE\_DETECTION\_TIME} \cdot \text{FPS} \rfloor$$

  Number of frame in which we need that the model detect something

DETECTION ALGORITHM

Accumulation Matices

We can do further processing for each detection

- **Idea**

  The camera is **static** so the bounding box of a fire should be always at the same point

- **Implementation**

  For each pixel inside the detection bounding boxes we accumulate the confidence and use this information for compute a weighted arithmetic mean (more importance is given to lower confidence)

DETECTION ALGORITHM

Fire Detection

- **Idea**

  If after 0.5 sec the net continusley saw fire **and** the wheighted mean of at least one pixel is higer than a fixed threshold we can rely on the network's ability to have detected a fire

- **Implementation**

  After we obtained a matrix with the wheighted mean of each pixel we can check if the max value inside this matrix is higer or equal to the confidence treshold, if so fire is declared.

EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithms, we created our own dataset to use as a test set which consists of 173 diverse videos depicting a wide range of scenarios. Among these videos:

- 60 videos do not contain any fires
- 113 videos contain flames and/or smoke

EXPERIMENTAL RESULTS

Arithmetic Mean

model_conf	fire_limit	eval_limit	TP	FP	FN	P	R	accuracy	F1-score
0.25	3	0.4	97	17	13	0.850	0.881	0.826	0.865
0.25	4	0.4	97	17	13	0.850	0.881	0.826	0.865
0.25	5	0.4	97	17	14	0.849	0.872	0.820	0.860
0.35	3	0.4	105	20	5	0.840	0.954	0.855	0.893
0.35	4	0.4	104	18	6	0.852	0.945	0.861	0.96
0.35	5	0.4	103	18	7	0.851	0.936	0.855	0.891

EXPERIMENTAL RESULTS

Dynamic Mean without Dynamic Fire Threshold

model_conf	fire_limit	eval_limit	TP	FP	FN	P	R	accuracy	F1-score
0.25	3	0.4	107	25	3	0.811	0.973	0.838	0.884
0.25	4	0.4	107	19	3	0.849	0.973	0.872	0.906
0.25	5	0.4	107	19	3	0.849	0.974	0.872	0.907
0.35	3	0.4	107	21	3	0.835	0.972	0.861	0.900
0.35	4	0.4	107	18	3	0.856	0.973	0.878	0.910
0.35	5	0.4	107	18	3	0.856	0.973	0.878	0.910

EXPERIMENTAL RESULTS

Proposed algorithm

model_conf	eval_limit	TP	FP	FN	P	R	accuracy	F1-score
0.25	0.4	107	15	3	0.877	0.973	0.896	0.923
0.25	0.45	106	15	4	0.876	0.964	0.890	0.918
0.25	0.5	104	15	6	0.874	0.945	0.879	0.908
0.35	0.4	106	13	4	0.891	0.964	0.902	0.926
0.35	0.45	105	13	5	0.890	0.955	0.896	0.921
0.35	0.5	104	13	6	0.889	0.945	0.890	0.916

THANK YOU FOR YOUR ATTENTION