

Large language models and deep learning for computer vision and time series: From academia to industry

Computer vision (cool) things to learn and do

Felipe Ferreira de Freitas

felipe.defreitas@edfenergy.com

[my github](#)

[my other github](#)

Felipe Ferreira de Freitas

- Graduate in Medical Physics(Catholic University of Pernambuco).
- Msc. in Particle Physics.
- Phd. in Particle Physics.
- Visiting research fellow at University of Sussex.
- Postdoctoral Researcher at Institute of Theoretical Physics-CAS.
- Postdoctoral Researcher at Aveiro University.
- Junior Researcher at Aveiro University.
- Machine Learning Scientist @ Digital Innovation EDF UK

Research topics:

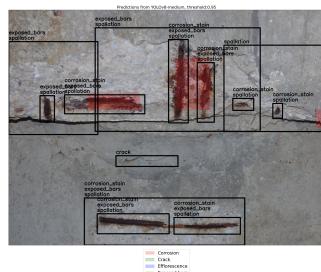
Quantum Machine Learning.

Development of DL methods.

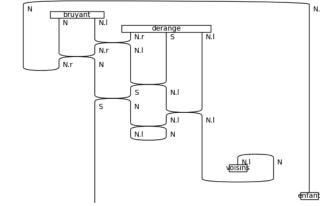
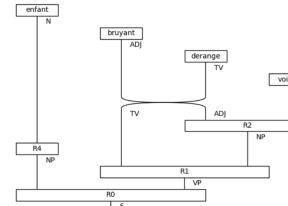
Computer vision methods for various applications.

F. F. Freitas
NLP and Large Language models.

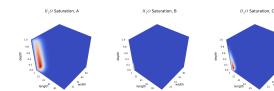
Deep learning methods in simulations.



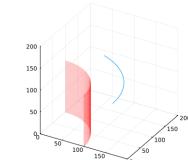
F. F. Freitas



F. F. Freitas



F. F. Freitas



F. F. Freitas

Disclaimer!!

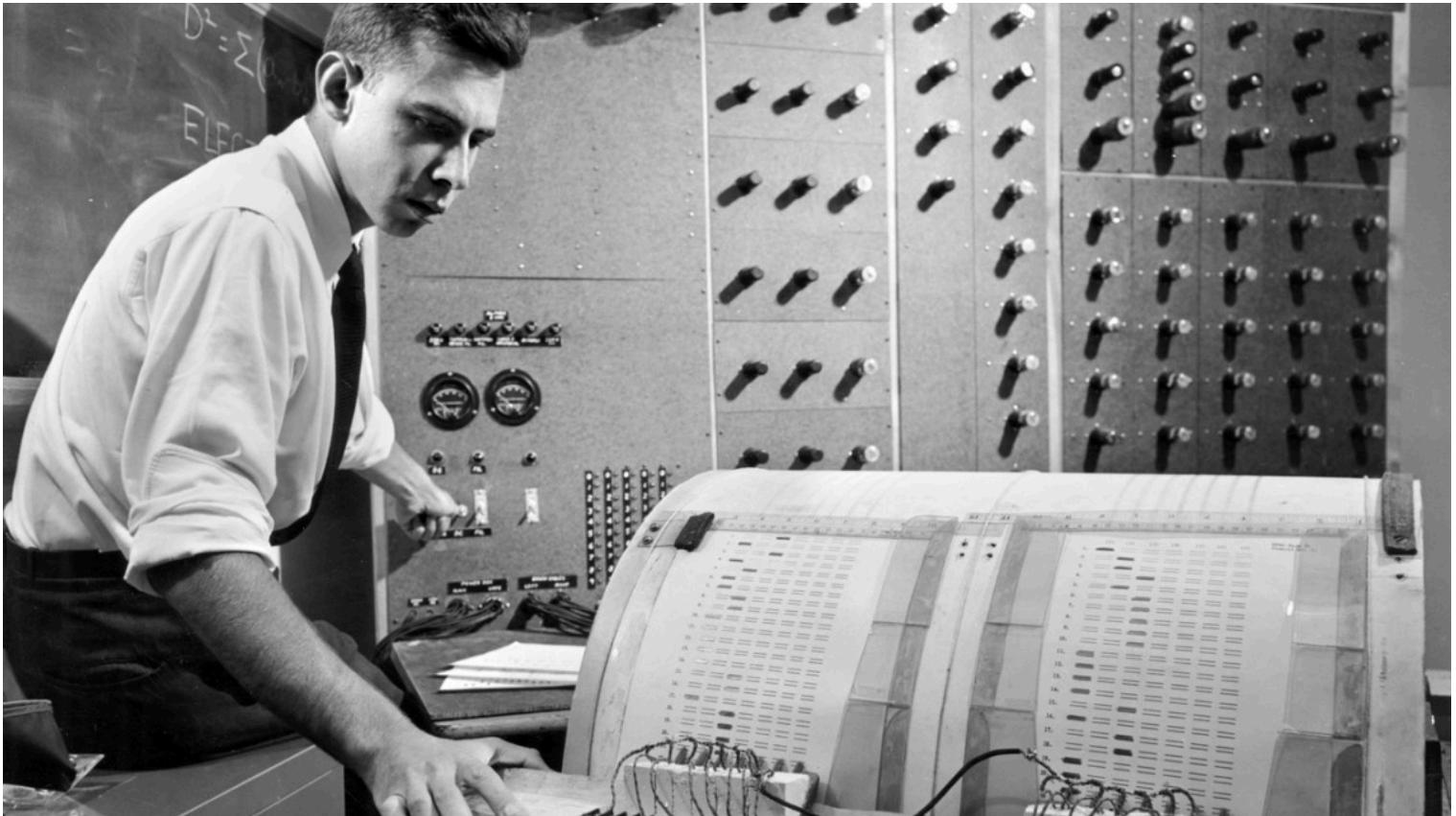
It is a very short lecture session and I can't cover everything

I can only give you the keys to the castle, but you have to open
the doors



A brief (and quick) history of AI:

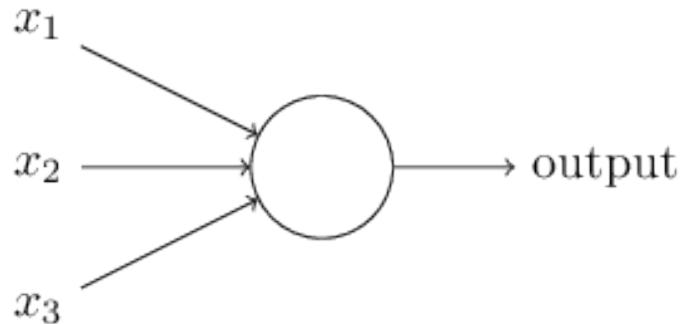
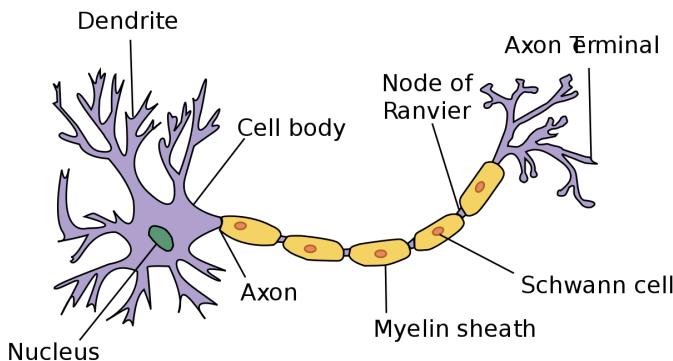
1957, The Perceptron: Frank Rosenblatt invents the perceptron while working at the Cornell Aeronautical Laboratory.



image

Rosenblatt Perceptron

- Rosenblatt's perceptron is based on an approximation of how a neuron works:
 - linear combination of inputs
 - an output if the cell is "activated"

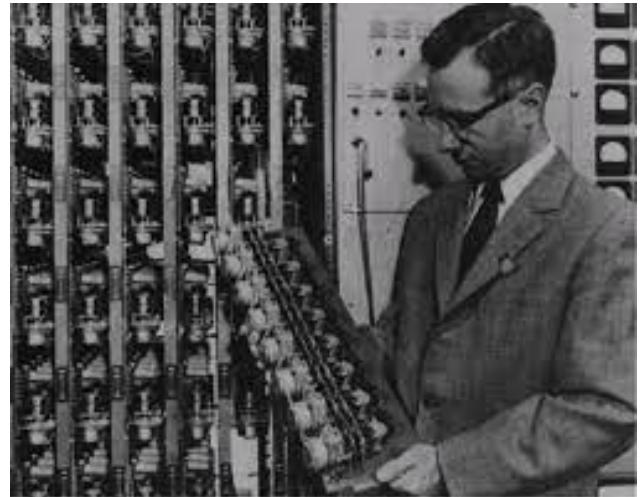
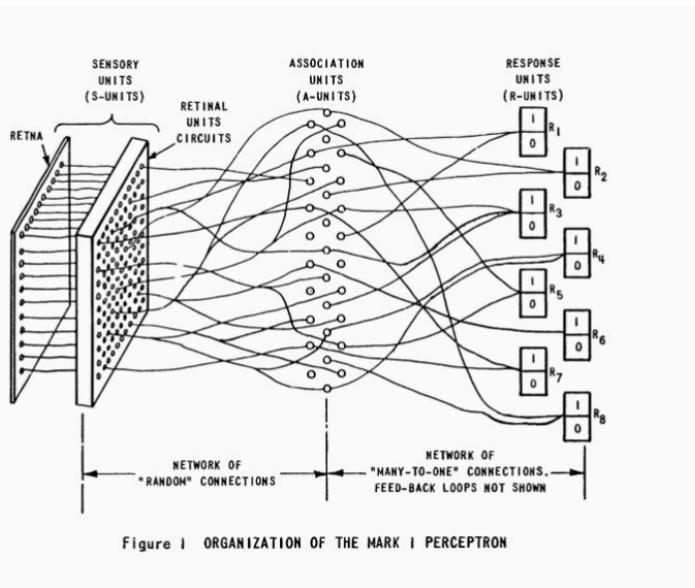


<http://neuralnetworksanddeeplearning.com/chap1.html>

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

- One way to think about the perceptron is that it is a device that makes decisions by evaluating evidence.

Rosenblatt Perceptron



- A 20×20 array of cadmium sulfide photocells to make a 400 pixel image.
- Each pixel in this case would be a neuron, and its values could be numbers between 0 (pixel off) and 1. (pixel completely on).
- All neurons are connected to a single neuron that determines the final output.

Rosenblatt Perceptron

In practice, what we have is matrix multiplication \begin{equation*}

$$(a_{1,1} \quad a_{1,2} \quad \cdots \quad a_{1,n})$$

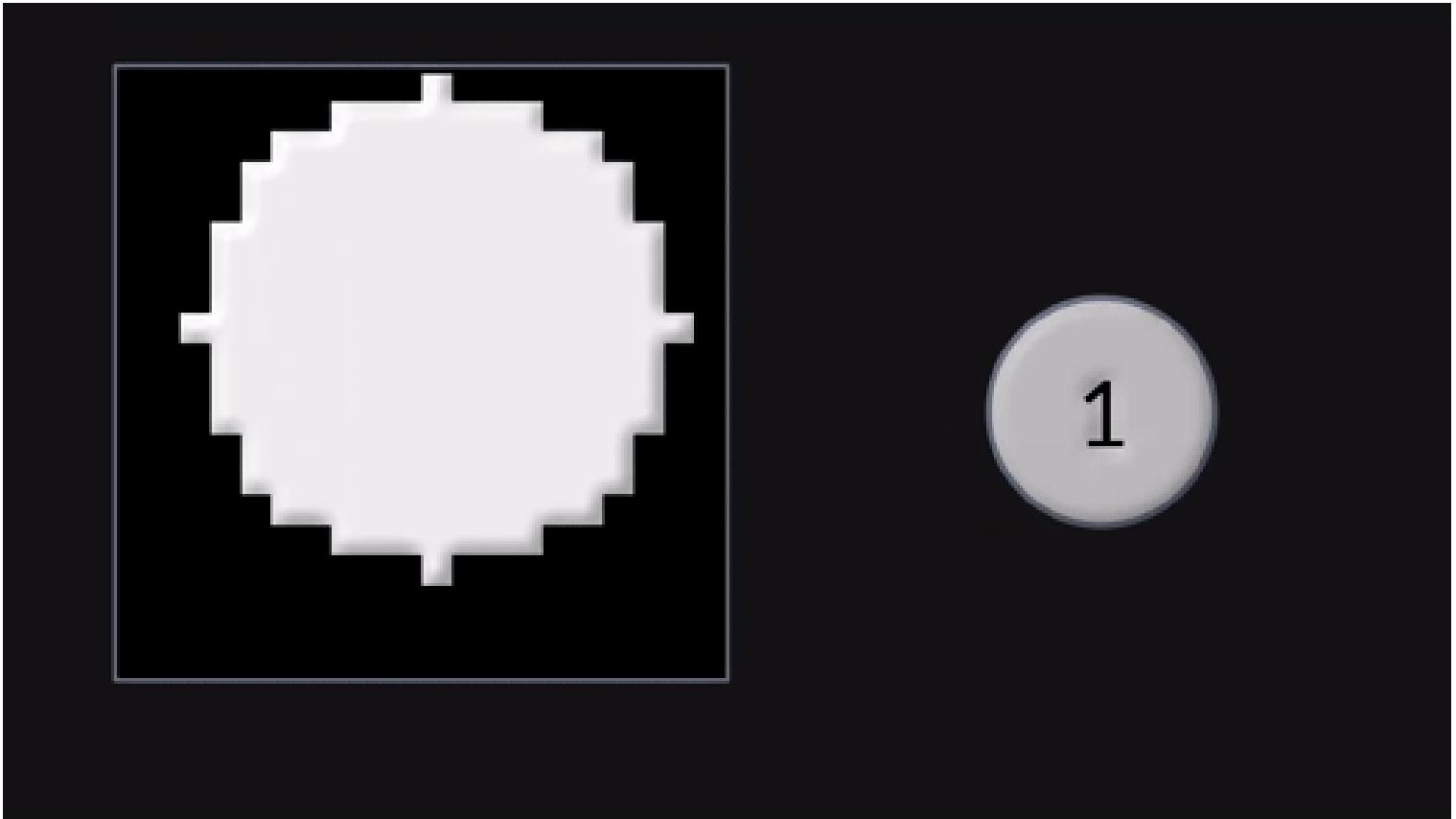
\times

$$\begin{pmatrix} w_{1,1} \\ w_{1,2} \\ \vdots \\ w_{1,n} \end{pmatrix}$$

Rosenblatt Perceptron

$$\begin{pmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,n} \end{pmatrix}$$

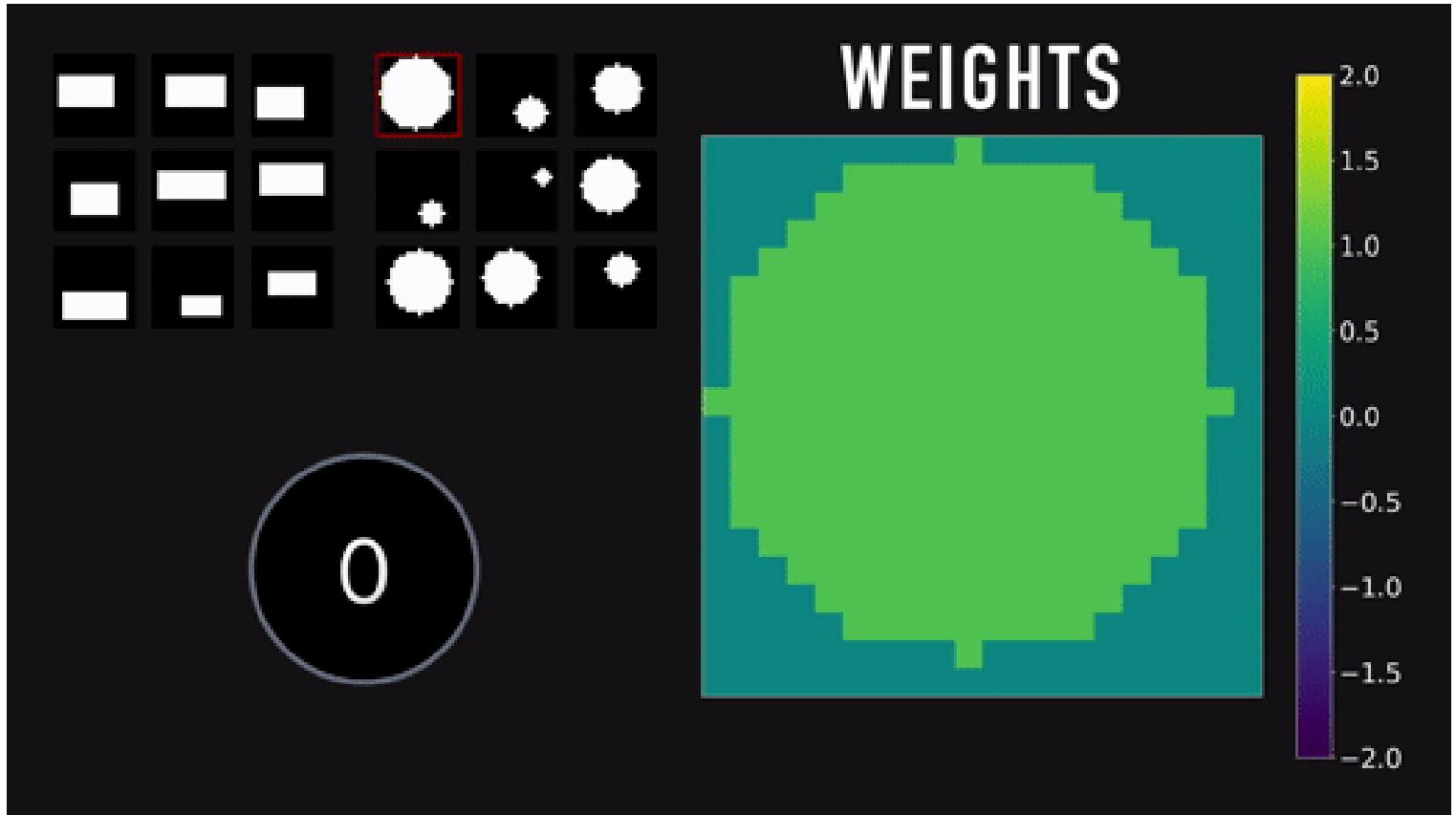
The main objective of the Rosenblatt Perceptron was to perceive
differences between two images, e.g. the difference between a circle
\end{equation*} and a rectangle.



Veritassium Youtube

--- [video](<https://www.youtube.com/watch?v=GVsUOuSjvcg>)

Training the Rosenblatt Perceptron



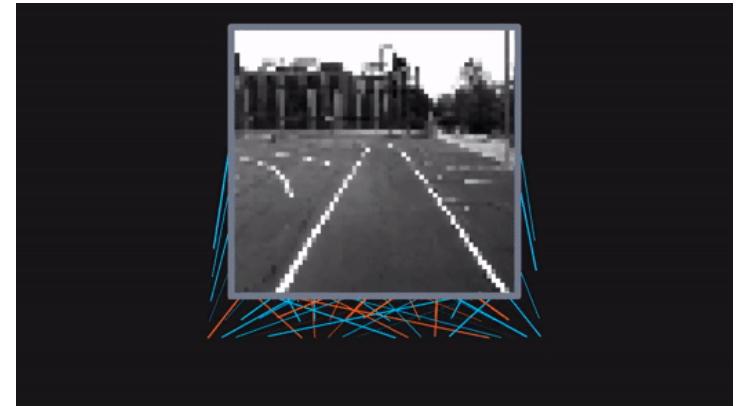
Veritassium Youtube

- If the neuron did not activate when it should have, we add the figure to the weights.
- If the neuron activated when it shouldn't, we subtract the figure from the weights.

In 1986 we had the first
autonomous vehicle:
NAVLAB



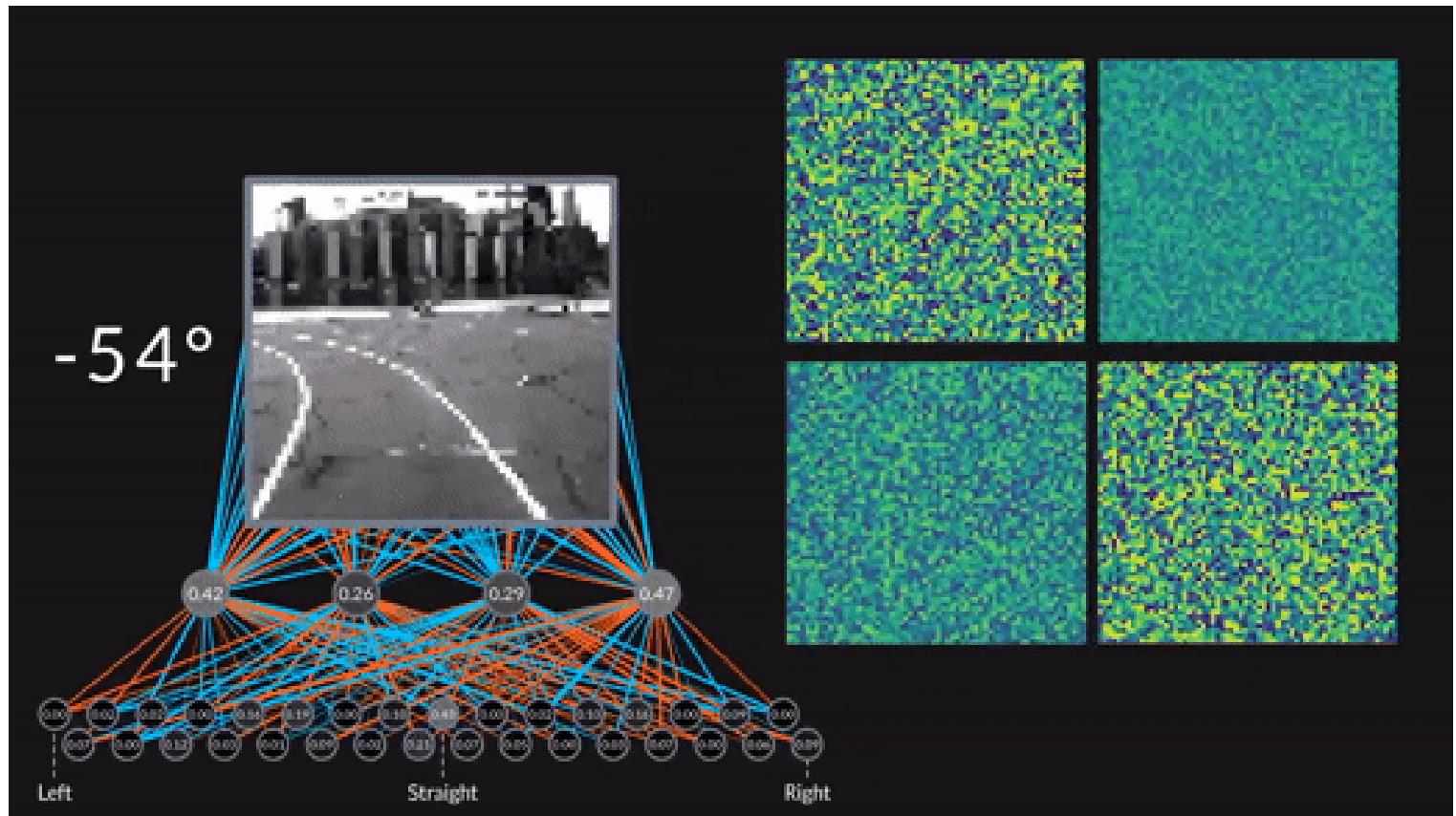
Veritassium Youtube



Veritassium Youtube

- 30 x 32 pixels as video input.
- A hidden layer with 4 neurons.
- An output with 32 neurons to determine the angle and speed of the car.
- To train a person first drove the car.
- The weights were adjusted using the backpropagation technique.
- Maximum speed: 2 km/h (hardware at the time was not efficient at all)

Training ALVIN:



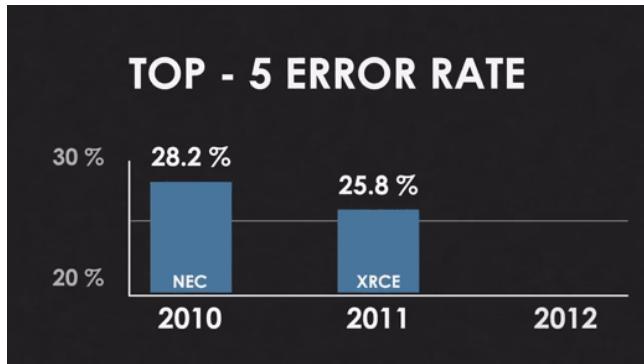
Veritassium Youtube

The second winter of
artificial intelligence

- Despite all these advances, the algorithms at the time were slow and quite inaccurate..
- The discussion revolved around optimizing the algorithms and/or improving the hardware.
- Professor Fei-Fei Li proposed a third solution to the problem: more data.
- In 2006, the ImageNet database was created, with more than 14 million images with 1,000 categories.
- Since 2010, the ImageNet project has held an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where software programs compete to correctly classify and detect objects and scenes.

AlexNet

One way to evaluate the performance of a neural network and look at the 5 categories that the network predicts and compare with the correct answer, this process is called the top-5 error rate.



Veritassium Youtube



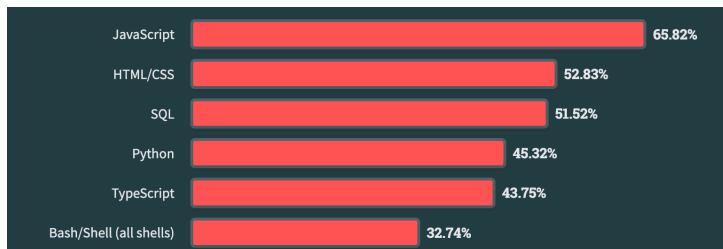
Veritassium Youtube

- Developed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton, AlexNet was a convolutional network with 8 layers, 5 convolutional layers and 3 dense layers, 500,000 neurons, with more than 16 million weights to be adjusted.
- It used very modern techniques (at the time), such as the *ReLU* activation function and was trained on GPUs.

Let's start build our own model

Pick your poison.

We all have our favourite program language, but for the better (or the worse) python is still the current dominant language in AI!



Stack overflow survey 2023



Google trends

Pick your poison.

You
can you tell me what was the main programming language used to develop you?

ChatGPT
As an AI language model developed by OpenAI, I was primarily created using Python. Python is widely used for machine learning and natural language processing tasks, making it a suitable choice for building AI models like me. Additionally, various other programming languages and technologies might have been utilized in the development process for specific components or optimizations, but Python served as the main language for building and training the model.

🔗 ↻ ⌂

ChatGPT 3.5 turbo

You
can you give a brief list of what are the possible other languages used in building you?

ChatGPT
Certainly! While Python was the primary language used to build me, other languages and technologies might have been involved in the development process for specific tasks or components. Some of these languages and technologies could include:

1. **C/C++**: Used for performance-critical components or optimizations, such as certain mathematical computations or low-level operations.
2. **CUDA**: If GPU acceleration was utilized for certain computations, CUDA (a parallel computing platform and programming model) might have been used in conjunction with C/C++.
3. **JavaScript**: For web-based interfaces or applications that interact with the AI model through browsers or web APIs.
4. **Shell scripting**: Used for automation tasks, managing environments, or deployment processes.
5. **TensorFlow or PyTorch**: These deep learning frameworks could have been used alongside Python for training and building neural networks.
6. **Java or Scala**: If components were developed using frameworks like Apache Spark for distributed computing tasks or data processing.
7. **SQL**: For managing and querying databases that store training data or other relevant information.
8. **HTML/CSS**: For designing and styling user interfaces, if applicable.

These are just some examples, and the actual technologies used might vary depending on the specific requirements and architecture of the AI model.

ChatGPT 3.5 turbo

The task and the data

RadioGalaxyNET: Dataset

The RadioGalaxyNET dataset encompasses 2800 3-channel images featuring two radio sky channels and one corresponding infrared sky channel. Both noisy and processed radio images are included in the dataset. It comprises 2800 extended radio galaxies, resulting in a total of 4155 instances of these galaxies due to their proximity in the sky and their appearance in multiple images.

The radio galaxies in the dataset are categorized as:

- 13% FR-I [1]
- 48% FR-II
- 14% FR-x
- 25% R sources

The dataset is divided into three sets: training, validation, and test sets, with a split ratio of 0.7:0.15:0.15

1. Fanaroff–Riley classification

Annotations

The RadioGalaxyNET dataset has the following four sets of labels

- the infrared host galaxy positions.
- the extended radio galaxy classes,
- the bounding boxes enclosing all components of each radio galaxy, and
- the segmentation masks for radio galaxies.

Annotations in the RadioGalaxyNET dataset adhere to the COCO dataset format (Lin et al. Reference Lin, Fleet, Pajdla, Schiele and Tuytelaars2014), facilitating straightforward comparisons of **object detection methods**. Radio annotations for each galaxy are stored as 'categories', 'box', and 'segmentation', while the positions of the infrared hosts are stored as 'keypoints'.

Download data:

!!!!here!!!!



<https://data.csiro.au/collection/csiro%3A61068v1>

Check the data!!

```
In [1]: !ls datasets/galaxyradionet/data/RadioGalaxyNET/
```

```
annotations test train train.cache val val.cache
```

```
In [2]: !ls datasets/galaxyradionet/data/RadioGalaxyNET/annotations/
```

```
test.json train.json val.json
```

```
In [3]: !ls -l datasets/galaxyradionet/data/RadioGalaxyNET/test/ |wc -l
```

```
421
```

```
In [4]: !ls -l datasets/galaxyradionet/data/RadioGalaxyNET/train/ |wc -l
```

```
1960
```

```
In [5]: !ls -l datasets/galaxyradionet/data/RadioGalaxyNET/val/ |wc -l
```

```
421
```

Check the labels

lets use `jason` and `pandas`

In [6]:

```
import json
import pandas as pd
```

Load the json files from the annotations folder

```
In [7]: with open("datasets/galaxyradionet/data/RadioGalaxyNET/annotations/training.json") as train_ann:  
    trn_ann = json.load(train_ann)  
    train_ann.close()
```



```
In [8]: with open("datasets/galaxyradionet/data/RadioGalaxyNET/annotations/testing.json") as test_ann:  
    tst_ann = json.load(test_ann)  
    test_ann.close()
```



```
In [9]: with open("datasets/galaxyradionet/data/RadioGalaxyNET/annotations/validation.json") as val_ann:  
    vld_ann = json.load(val_ann)  
    val_ann.close()
```

What is this thing?

JSON (JavaScript Object Notation) are usually load in python as [dictionaries](#), which we have keys and values:

```
In [10]: type(trn_ann)
```

```
Out[10]: dict
```

```
In [11]: trn_ann.keys()
```

```
Out[11]: dict_keys(['info', 'licenses', 'images', 'categories', 'annotations'])
```

```
In [12]: trn_ann['categories']
```

```
Out[12]: [{  
    'supercategory': 'FR-II', 'id': 1, 'name': 'FR-II'},  
    {'supercategory': 'FR-I', 'id': 2, 'name': 'FR-I'},  
    {'supercategory': 'FR-X', 'id': 3, 'name': 'FR-X'},  
    {'supercategory': 'R', 'id': 4, 'name': 'R'}]
```

We can use such dictionaries to build dataframe to make our work a bit easier!

```
In [14]: df_ann = pd.DataFrame.from_dict(trn_ann['annotations'])
```

```
In [15]: df_ann.keys()
```

```
Out[15]: Index(['segmentation', 'num_keypoints', 'area', 'iscrowd', 'image_id', 'bbox',
       'keypoints', 'category_id', 'id'],
      dtype='object')
```

```
In [16]: df_imgs = pd.DataFrame.from_dict(trn_ann["images"])
```

```
In [17]: df_ann.head(10)
```

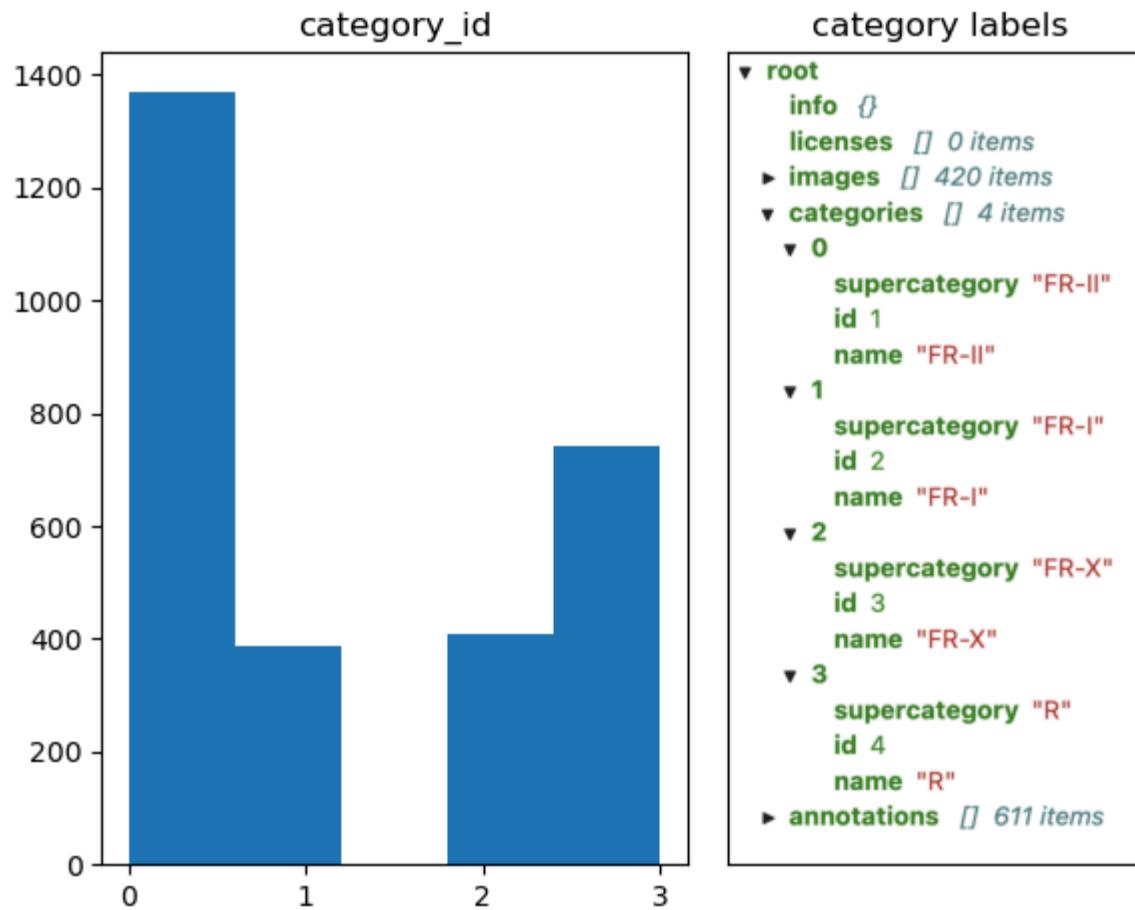
	segmentation	num_keypoints	area	iscrowd	image_id	bbox	keypoints
0	[[194.0, 214.5, 197.0, 213.5, 198.5, 211.0, 18...]	1	3666.0	0	1	[182.5, 197.5, 78.0, 47.0]	
1	[[226.0, 242.5, 234.0, 238.5, 238.0, 233.5, 24...]	1	1360.0	0	2	[208.5, 202.5, 34.0, 40.0]	
2	[[228.0, 229.5, 233.0, 227.5, 235.5, 224.0, 22...]	1	330.0	0	3	[213.5, 214.5, 22.0, 15.0]	
3	[[144.0, 213.5, 144.5, 211.0, 141.5, 209.0, 14...]	1	2613.0	0	3	[129.5, 174.5, 67.0, 39.0]	
4	[[228.0, 231.5, 234.5, 225.0, 232.0, 222.5, 22...]	1	414.0	0	4	[211.5, 213.5, 23.0, 18.0]	
5	[[228.0, 230.5, 231.5, 229.0, 232.5, 225.0, 23...]	1	266.0	0	5	[213.5, 216.5, 19.0, 14.0]	

	segmentation	num_keypoints	area	iscrowd	image_id	bbox	keypoints
6	[[208.0, 238.5, 212.5, 237.0, 214.0, 232.5, 22...]		1 1161.0	0	6	[198.5, 211.5, [225, 225, 43.0, 27.0]	
7	[[148.0, 77.5, 152.0, 76.5, 156.5, 72.0, 154.5...]		1 609.0	0	6	[127.5, 56.5, [149, 66, 29.0, 21.0]	
8	[[233.0, 245.5, 232.5, 243.0, 234.0, 241.5, 23...]		1 1591.0	0	7	[205.5, 202.5, [225, 225, 37.0, 43.0]	
9	[[226.0, 248.5, 231.5, 243.0, 231.5, 237.0, 22...]		1 1275.0	0	8	[211.5, 197.5, [225, 225, 25.0, 51.0]	

```
In [18]: import matplotlib.pyplot as plt
```

```
In [19]: fig, axs = plt.subplots(1, 2, tight_layout=True)

# We can set the number of bins with the *bins* keyword argument.
axs[0].hist(df_ann["category_id"] - 1, align='mid', bins=5)
axs[0].set_title("category_id")
axs[0].set_xticks(sorted(df_ann["category_id"].unique() - 1))
img_legend = plt.imread("./figs/json_labels.png")
axs[1].imshow(img_legend)
axs[1].set_title("category labels")
axs[1].set_xticks([])
axs[1].set_yticks([]);
```



Check some images

for this we can use either [OpenCV](#) or [Pillow](#) (aka pil) for load and check some of the images

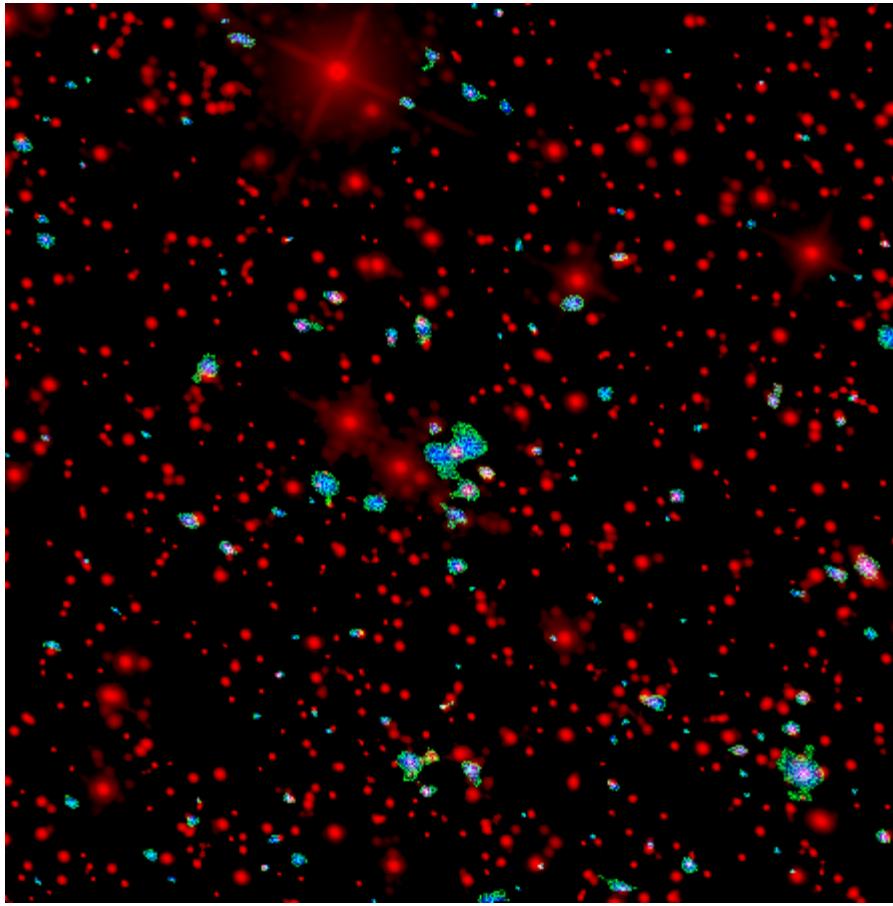
```
In [20]: from PIL import Image  
        from pathlib import Path
```

```
In [21]: sample_path = Path('datasets/galaxyradionet/data/RadioGalaxyNET/test/48')
```

```
In [22]: image_sample = Image.open(sample_path)
```

```
In [23]: image_sample
```

Out[23]:



```
In [24]: image_sample.format_description
```

```
Out[24]: 'Portable network graphics'
```

```
In [25]: image_sample.size
```

```
Out[25]: (450, 450)
```

```
In [26]: image_sample.info
```

```
Out[26]: {}
```

```
In [27]: image_sample.getbbox()
```

```
Out[27]: (0, 0, 450, 450)
```

Explorartory data analysis (EDA)

What we just did was a basic exploratory data analysis, I strong reccomend do this before initiate any work.

1. access and store the data in a convinient place
2. check the integrety and what is the data type you are working
3. show off some images, tables, summary statistics

We could explore a bit more, but for the sake of time lest keep as it is. For further readings -
> [EDA](#)

Our task: Object detection

The goal is to detect and locate objects of interest in an image or video, in our case identify and locate the types of radio galaxies with active nuclei using the images available in the RadioGalaxy dataset.

Process annotations to ultralytics

We are going to use YOLOv8 model available in ultralytics library, we also going to use other useful functions from ultralytics. However, ultralytics requires some small adjustments in our dataset in order to make full use of it. Luck for us the curators of the RadioGalaxyNET did an amazing job by providing very useful python scripts to do this task, we can use these scripts:

1. clone the repository from [here](#)
2. we are going to use the scripts in the folder JSON2YOLO
3. if you intend to use the scripts inside this notebook, I recommend to remove the code after the part:

```
if __name__ == '__main__':
```

4. create a new folder with this structure:

```
yolo/YOL0_RadioGalaxyNET/labels
```

```
In [39]: import sys
```

```
In [40]: sys.path.append("JSON2YOL0/")
```

```
In [41]: from general_json2yolo import convert_coco_json
```

```
In [44]: convert_coco_json('datasets/galaxyradionet/data/RadioGalaxyNET/annotations/test.json',
                         use_segments=False,
                         use_keypoints=False,
                         cls91to80=False,
                         savedir_path='./yolo/YOL0_RadioGalaxyNET/')
```

```
Annotations /home/sagemaker-user/datasets/galaxyradionet/data/RadioGalaxyNET/annotations/test.json: 100%|██████████| 420/420 [00:00<00:00, 12627.92it/s]
Annotations /home/sagemaker-user/datasets/galaxyradionet/data/RadioGalaxyNET/annotations/train.json: 100%|██████████| 1959/1959 [00:00<00:00, 11239.11it/s]
Annotations /home/sagemaker-user/datasets/galaxyradionet/data/RadioGalaxyNET/annotations/val.json: 100%|██████████| 420/420 [00:00<00:00, 11588.76it/s]
```

the `convert_coco_json` function will automatically create the folder with the labels as expected by the ultralytics

```
image_name.txt
```

and in it we have:

```
2 0.505556 0.496667 0.102222 0.08  
0 0.0644444 0.568889 0.0777778 0.0644444
```

which correspond to:

- class id (Class numbers are zero-indexed)
- class x_center y_center width height format.
- Box coordinates normalized xywh format (from 0 to 1).

If your boxes are in pixels, divide x_center and width by image width, and y_center and height by image height.

creat a series of folders with the follow structure:

```
mkdir -p yolo/YOL0_RadioGalaxyNET/images/train  
mkdir -p yolo/YOL0_RadioGalaxyNET/images/test  
mkdir -p yolo/YOL0_RadioGalaxyNET/images/val
```

either move or copy the respective images from the RadioGalaxyNET data we download to their folders in the yolo structure we just create.

⚠⚠⚠ DO NOT USE symbolic links ⚠⚠⚠

We should have an structure like this:

```
In [47]: list_files("yolo2/")
```

```
/  
YOLO_RadioGalaxyNET/  
    images/  
        train/  
        test/  
        val/  
    labels/  
        train/  
        test/  
        val/
```

last but not least!!

we create a [yaml](#) which ultralytics uses to train our model. This file will contains the path information for our images, labels and also the names of our classes and their respective IDs

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file:  
# path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]  
path: ../yolo/YOLO_RadioGalaxyNET # dataset root dir  
train: images/train # train images (relative to 'path') 128 images  
val: images/val # val images (relative to 'path') 128 images  
test: images/test # (optional)  
  
# Classes  
names:  
0: FR-II  
1: FR-I  
2: FR-X  
3: R
```

Now we can start train our YOLOv8 model!!!

- 1) check our torch version (preferable >= 2.0)
- 2) check if we have gpu + cuda available, otherwise ->



```
In [28]: import torch
```

```
In [29]: torch.__version__
```

```
Out[29]: '2.0.0.post200'
```

```
In [30]: # import numpy never hurts
import numpy as np
```

```
In [31]: # do we have gpu?  
!nvidia-smi
```

```
Tue Apr  9 14:23:37 2024  
+-----  
| NVIDIA-SMI 535.129.03                 Driver Version: 535.129.03  
| CUDA Version: 12.2  
|-----+  
| GPU Name                  Persistence-M | Bus-Id      Disp.  
A | Volatile Uncorr. ECC |  
| Fan Temp   Perf          Pwr:Usage/Cap |      Memory-Usag  
e | GPU-Util  Compute M.  |  
| MIG M.  |  
+-----+  
| 0  Tesla T4                Off  | 00000000:00:1E.0 Of  
f |                               0 |  
| N/A  26C      P0            25W / 70W |      2MiB / 15360Mi  
B |           5%     Default |  
| N/A  |  
+-----+  
+-----+  
+-----+  
| Processes:
```

GPU	GI	CI	PID	Type	Process name
GPU Memory					
	ID	ID			
Usage					
<hr/>					
<hr/>					
No running processes found					
<hr/>					
<hr/>					

```
In [32]: # check if cuda is available  
torch.cuda.is_available()
```

Out[32]: True

Perfeito, we can now load ultralytics (assuming we had sucessfully installed)

```
In [33]: import ultralytics
```

!!! important: check if our ultralytics instalation is ok:

```
In [34]: ultralytics.checks()
```

```
Ultralytics YOLOv8.1.45 🚀 Python-3.10.13 torch-2.0.0.post200 C
UDA:0 (Tesla T4, 14931MiB)
Setup complete ✅ (4 CPUs, 15.4 GB RAM, 0.2/32.0 GB disk)
```

And we can use the YOLO class from ultralytics to build our model

```
In [35]: from ultralytics import YOLO
```

Build our model

in this case a YOLOv8 nano, you can check which options are available from [here](#), we are also going to make use of transfer learning

```
In [36]: # build model architecture  
model = YOLO("yolov8n.yaml")
```

```
In [37]: # check the model architecture  
#list(model.modules())
```

```
In [38]: # to download and use pre-trained models  
model = YOLO("yolov8n.pt")
```

Train the model in our data

We can use the train function from the YOLO class, we just need to provide the yaml file we create previously `RadioGalaxyNET.yaml` to the argument data and set the number of epochs we wish to train the model.

Other parameters/arguments can be found [here](#)

```
In [ ]: #?model.train
```

```
In [39]: # Train the
model.train(data='./RadioGalaxyNET.yaml', epochs=50)
```

```
Ultralytics YOLOv8.1.45 🚀 Python-3.10.13 torch-2.0.0.post200 C
UDA:0 (Tesla T4, 14931MiB)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data
= './RadioGalaxyNET.yaml', epochs=50, time=None, patience=100, bat
ch=16, imgsz=640, save=True, save_period=-1, cache=False, devic
e=None, workers=8, project=None, name=train, exist_ok=False, pr
etrained=True, optimizer=auto, verbose=True, seed=0, determinis
tic=True, single_cls=False, rect=False, cos_lr=False, close_mos
aic=10, resume=False, amp=True, fraction=1.0, profile=False, fr
eeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4,
dropout=0.0, val=True, split=val, save_json=False, save_hybrid=
False, conf=None, iou=0.7, max_det=300, half=False, dnn=False,
plots=True, source=None, vid_stride=1, stream_buffer=False, vis
ualize=False, augment=False, agnostic_nms=False, classes=None,
```

```
retina_masks=False, embed=None, show=False, save_frames=False,
save_txt=False, save_conf=False, save_crop=False, show_labels=True,
show_conf=True, show_boxes=True, line_width=None, format=torchscript,
keras=False, optimize=False, int8=False, dynamic=False,
simplify=False, opset=None, workspace=4, nms=False, lr0=0.01,
lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epoch=3.0,
warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5,
dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64,
hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1,
scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5,
gr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=рандаугмент,
erasing=0.4, crop_fraction=1.0, cfg=None, tracker=bot
sort.yaml, save_dir=runs/detect/train
Overriding model.yaml nc=80 with nc=4
```

	from	n	params	module
arguments				
0		-1	1	464 ultralytics.nn.modules.co
nv.Conv		[3,	16,	3, 2]
1		-1	1	4672 ultralytics.nn.modules.co
nv.Conv		[16,	32,	3, 2]
2		-1	1	7360 ultralytics.nn.modules.bl
ock.C2f		[32,	32,	1, True]
3		-1	1	18560 ultralytics.nn.modules.co
nv.Conv		[32,	64,	3, 2]
4		-1	2	49664 ultralytics.nn.modules.bl
ock.C2f		[64,	64,	2, True]
5		-1	1	73984 ultralytics.nn.modules.co
nv.Conv		[64,	128,	3, 2]
6		-1	2	197632 ultralytics.nn.modules.bl
ock.C2f		[128,	128,	2, True]

7		-1	1	295424	ultralytics.nn.modules.co
nv.Conv		[128, 256, 3, 2]			
8		-1	1	460288	ultralytics.nn.modules.bl
ock.C2f		[256, 256, 1, True]			
9		-1	1	164608	ultralytics.nn.modules.bl
ock.SPPF		[256, 256, 5]			
10		-1	1	0	torch.nn.modules.upsampli
ng.Upsample		[None, 2, 'nearest']			
11		[-1, 6] 1		0	ultralytics.nn.modules.co
nv.Concat		[1]			
12		-1	1	148224	ultralytics.nn.modules.bl
ock.C2f		[384, 128, 1]			
13		-1	1	0	torch.nn.modules.upsampli
ng.Upsample		[None, 2, 'nearest']			
14		[-1, 4] 1		0	ultralytics.nn.modules.co
nv.Concat		[1]			
15		-1	1	37248	ultralytics.nn.modules.bl
ock.C2f		[192, 64, 1]			
16		-1	1	36992	ultralytics.nn.modules.co
nv.Conv		[64, 64, 3, 2]			
17		[-1, 12] 1		0	ultralytics.nn.modules.co
nv.Concat		[1]			
18		-1	1	123648	ultralytics.nn.modules.bl
ock.C2f		[192, 128, 1]			
19		-1	1	147712	ultralytics.nn.modules.co
nv.Conv		[128, 128, 3, 2]			
20		[-1, 9] 1		0	ultralytics.nn.modules.co
nv.Concat		[1]			
21		-1	1	493056	ultralytics.nn.modules.bl
ock.C2f		[384, 256, 1]			
22	[15, 18, 21]	1		752092	ultralytics.nn.modules.he

```
ad.Detect [4, [64, 128, 256]]  
Model summary: 225 layers, 3011628 parameters, 3011612 gradient  
s, 8.2 GFLOPs
```

Transferred 319/355 items from pretrained weights

TensorBoard: Start with 'tensorboard --logdir runs/detect/trai
n', view at <http://localhost:6006/>

Freezing layer 'model.22.dfl.conv.weight'

AMP: running Automatic Mixed Precision (AMP) checks with YOL0v8
n...

AMP: checks passed ✓

```
train: Scanning /home/sagemaker-user/yolo/YOL0_RadioGalaxyNET/l  
abels/train.cache... 1959 images, 0 backgrounds, 0 corrupt: 10  
0%|██████████| 1959/1959 [00:00<?, ?it/s]
```

```
val: Scanning /home/sagemaker-user/yolo/YOL0_RadioGalaxyNET/lab  
els/val.cache... 420 images, 0 backgrounds, 0 corrupt: 100%|████  
████| 420/420 [00:00<?, ?it/s]
```

Plotting labels to runs/detect/train/labels.jpg...

optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'mom
entum=0.937' and determining best 'optimizer', 'lr0' and 'momen
tum' automatically...

optimizer: AdamW(lr=0.00125, momentum=0.9) with parameter group
s 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=
0.0)

TensorBoard: model graph visualization added ✓

Image sizes 640 train, 640 val

Using 4 dataloader workers

Logging results to **runs/detect/train**

Starting training for 50 epochs...

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

17	1/50	2.35G	1.333	3.354	1.052	
	640: 100% ██████████ 123/123 [00:27<00:00, 4.41it/s]					

R	mAP50	Class mAP50-95): 100% ██████████ 14/14 [00:02<00:00, 5.15it/s]	Images	Instances	Box(P)	
---	-------	--	--------	-----------	--------	--

347		all	420	631	0.173	0.
	0.195	0.126				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

21	2/50	2.35G	1.109	2.43	0.9916	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.96it/s]					

R	mAP50	Class mAP50-95): 100% ██████████ 14/14 [00:02<00:00, 5.32it/s]	Images	Instances	Box(P)	
---	-------	--	--------	-----------	--------	--

0.49		all	420	631	0.243	
	0.262	0.156				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

18	3/50	2.33G	0.9648	2.138	0.9609	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.94it/s]					

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00, 4.98it/s]

		all	420	631	0.217	0.
444	0.256	0.209				

ces	Epoch Size	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
-----	------------	---------	----------	----------	----------	--------

21	4/50 640: 100%	2.34G ██████████ 123/123 [00:24<00:00, 4.95it/s]	0.8947	1.931	0.9461	
----	----------------	--	--------	-------	--------	--

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00, 5.27it/s]

463	0.303	all 0.25	420	631	0.282	0.
-----	-------	----------	-----	-----	-------	----

ces	Epoch Size	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
-----	------------	---------	----------	----------	----------	--------

19	5/50 640: 100%	2.33G ██████████ 123/123 [00:25<00:00, 4.88it/s]	0.8196	1.792	0.9312	
----	----------------	--	--------	-------	--------	--

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00, 5.48it/s]

		all	420	631	0.226	0.
--	--	-----	-----	-----	-------	----

557 0.318 0.194

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

6/50 2.33G 0.7668 1.655 0.9115
22 640: 100%|██████████| 123/123 [00:24<00:00, 4.94it/
s]
R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
4.87it/s]

541 0.394 all 420 631 0.34 0.
0.261

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

7/50 2.33G 0.7496 1.609 0.9187
10 640: 100%|██████████| 123/123 [00:25<00:00, 4.88it/
s]
R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.73it/s]

502 0.395 all 420 631 0.358 0.
0.231

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

8/50 2.32G 0.7241 1.511 0.9099
15 640: 100%|██████████| 123/123 [00:24<00:00, 4.93it/
s]
R Class Images Instances Box(P
mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.39it/s]

532 all 420 631 0.319 0.
0.375 0.315

Epoch GPU_mem box_loss cls_loss dfl_loss Instan
ces Size

9/50 2.33G 0.6955 1.471 0.9059
16 640: 100%|██████████| 123/123 [00:24<00:00, 4.98it/
s]
R Class Images Instances Box(P
mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.02it/s]

568 all 420 631 0.328 0.
0.381 0.324

Epoch GPU_mem box_loss cls_loss dfl_loss Instan
ces Size

10/50 2.32G 0.695 1.442 0.9062
12 640: 100%|██████████| 123/123 [00:24<00:00, 4.97it/
s]
R Class Images Instances Box(P

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
4.87it/s]

		all	420	631	0.367	0.
593	0.43	0.379				

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

11/50 2.34G 0.6924 1.443 0.9131
19 640: 100%|██████████| 123/123 [00:24<00:00, 4.94it/
s]

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.13it/s]

	all	420	631	0.301	0.
548	0.378	0.231			

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

12/50 2.33G 0.6634 1.417 0.9
17 640: 100%|██████████| 123/123 [00:24<00:00, 4.95it/
s]

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.36it/s]

	all	420	631	0.322	
0.46	0.341	0.208			

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					
14	13/50	2.34G	0.6507	1.34	0.8892	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.92it/s]					
R	mAP50	Class mAP50–95): 100% ██████████ 14/14 [00:02<00:00, 5.59it/s]	Images	Instances	Box(P	
	0.56	0.464	all 0.403	420	631	0.437

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					
7	14/50	2.33G	0.6423	1.357	0.8943	
	640: 100% ██████████ 123/123 [00:25<00:00, 4.92it/s]					
R	mAP50	Class mAP50–95): 100% ██████████ 14/14 [00:02<00:00, 5.25it/s]	Images	Instances	Box(P	
	543	0.487	all 0.323	420	631	0.454

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					
16	15/50	2.32G	0.6272	1.316	0.8912	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.95it/s]					

s]

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.11it/s]

		all	420	631	0.446	0.
577	0.505	0.448				

ces	Epoch Size	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
-----	------------	---------	----------	----------	----------	--------

17 16/50 2.32G 0.6179 1.287 0.8834
640: 100%|██████████| 123/123 [00:25<00:00, 4.92it/
s]

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.66it/s]

545	0.473	all	420	631	0.437	0.
		0.421				

ces	Epoch Size	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
-----	------------	---------	----------	----------	----------	--------

21 17/50 2.33G 0.6017 1.312 0.8835
640: 100%|██████████| 123/123 [00:24<00:00, 4.99it/
s]

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.27it/s]

		all	420	631	0.393	0.
569	0.495	0.446				

ces	Epoch Size	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
15	18/50 640: 100% [██████████ 123/123 [00:24<00:00, 4.96it/s]	2.32G	0.5836	1.24	0.879	
R	mAP50 mAP50-95): 100% [██████████ 14/14 [00:02<00:00, 4.88it/s]	Class Images Instances		Box(P		

		all	420	631	0.45	0.
481	0.454	0.406				

ces	Epoch Size	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
11	19/50 640: 100% [██████████ 123/123 [00:25<00:00, 4.90it/s]	2.34G	0.5918	1.242	0.8738	
R	mAP50 mAP50-95): 100% [██████████ 14/14 [00:02<00:00, 5.74it/s]	Class Images Instances		Box(P		

		all	420	631	0.467	
0.55	0.503	0.452				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

14	20/50	2.32G	0.5861	1.224	0.8742	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.97it/s]					

R	mAP50	Class mAP50–95): 100% ██████████ 14/14 [00:02<00:00, 5.36it/s]	Images	Instances	Box(P	
---	-------	--	--------	-----------	-------	--

575		all	420	631	0.495	0.
	0.531	0.476				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

9	21/50	2.34G	0.5629	1.217	0.8697	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.99it/s]					
R	mAP50	Class mAP50–95): 100% ██████████ 14/14 [00:02<00:00, 4.99it/s]	Images	Instances	Box(P	

522		all	420	631	0.539	0.
	0.534	0.473				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

19	22/50	2.32G	0.5571	1.209	0.8649	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.96it/s]					

		Class	Images	Instances	Box(P)	
R	mAP50	mAP50–95): 100%	██████████	14/14 [00:02<00:00,		
		4.83it/s]				

564	0.504	all	420	631	0.477	0.
		0.458				

ces	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
	Size					

16	23/50	2.34G	0.5648	1.203	0.8705	
s]	640: 100%	██████████	123/123 [00:24<00:00, 4.96it/			

		Class	Images	Instances	Box(P)	
R	mAP50	mAP50–95): 100%	██████████	14/14 [00:02<00:00,		
		5.26it/s]				

0.6	0.548	all	420	631	0.487	
		0.492				

ces	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
	Size					

12	24/50	2.32G	0.5457	1.175	0.8733	
s]	640: 100%	██████████	123/123 [00:24<00:00, 4.99it/			

		Class	Images	Instances	Box(P)	
R	mAP50	mAP50–95): 100%	██████████	14/14 [00:02<00:00,		
		5.22it/s]				

		all	420	631	0.395	0.
--	--	-----	-----	-----	-------	----

695 0.525 0.326

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
ces	Size				

25/50	2.32G	0.5395	1.168	0.867	
16	640: 100% ██████████ 123/123 [00:25<00:00, 4.90it/s]				
R	mAP50	Class mAP50-95): 100% ██████████ 14/14 [00:02<00:00, 5.71it/s]	Images	Instances	Box(P)

546	0.548	all	420	631	0.525	0.
		0.497				

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
ces	Size				

26/50	2.32G	0.5468	1.171	0.8653	
9	640: 100% ██████████ 123/123 [00:25<00:00, 4.90it/s]				
R	mAP50	Class mAP50-95): 100% ██████████ 14/14 [00:02<00:00, 5.59it/s]	Images	Instances	Box(P)

0.62	0.566	all	420	631	0.512	
		0.515				

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
ces	Size				

27/50 2.32G 0.543 1.137 0.8666
15 640: 100%|██████████| 123/123 [00:24<00:00, 4.93it/
s]
R Class Images Instances Box(P
mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.15it/s]

598 all 420 631 0.464 0.
0.523 0.48

Epoch GPU_mem box_loss cls_loss dfl_loss Instan
ces Size

28/50 2.32G 0.5224 1.117 0.8553
14 640: 100%|██████████| 123/123 [00:24<00:00, 4.92it/
s]
R Class Images Instances Box(P
mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.46it/s]

561 all 420 631 0.542 0.
0.557 0.509

Epoch GPU_mem box_loss cls_loss dfl_loss Instan
ces Size

29/50 2.33G 0.5306 1.091 0.8626
9 640: 100%|██████████| 123/123 [00:24<00:00, 4.95it/s]
R Class Images Instances Box(P

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.25it/s]

643	0.575	all 0.523	420	631	0.484	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

30/50	2.34G	0.5199	1.089	0.8626	
11	640: 100% ██████████ 123/123 [00:24<00:00, 4.92it/ s]				

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
4.98it/s]

566	0.588	all 0.449	420	631	0.583	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

31/50	2.32G	0.5136	1.078	0.857	
10	640: 100% ██████████ 123/123 [00:24<00:00, 4.94it/ s]				

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.51it/s]

599	0.57	all 0.367	420	631	0.495	0.
-----	------	--------------	-----	-----	-------	----

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
	ces	Size				
10	32/50	2.33G	0.506	1.061	0.8494	
		640: 100% ██████████ 123/123 [00:24<00:00, 4.97it/s]				
			Class Images Instances		Box(P	
R	mAP50	mAP50–95): 100% ██████████ 14/14 [00:02<00:00,			14/14 [00:02<00:00,	
		4.82it/s]				
649	0.564	all	420	631	0.478	0.
		0.517				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
	ces	Size				
17	33/50	2.32G	0.4992	1.034	0.8494	
		640: 100% ██████████ 123/123 [00:24<00:00, 4.97it/s]				
			Class Images Instances		Box(P	
R	mAP50	mAP50–95): 100% ██████████ 14/14 [00:02<00:00,			14/14 [00:02<00:00,	
		5.25it/s]				
0.61	0.575	all	420	631	0.513	
		0.533				

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
	ces	Size				

34/50 2.32G 0.4966 1.063 0.8562
11 640: 100%|██████████| 123/123 [00:24<00:00, 4.97it/
s]
R Class Images Instances Box(P
mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.26it/s]

all 420 631 0.47 0.
607 0.56 0.513

Epoch GPU_mem box_loss cls_loss dfl_loss Instan
ces Size

35/50 2.32G 0.498 1.04 0.8527
17 640: 100%|██████████| 123/123 [00:24<00:00, 4.94it/
s]
R Class Images Instances Box(P
mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.32it/s]

all 420 631 0.531 0.
627 0.595 0.55

Epoch GPU_mem box_loss cls_loss dfl_loss Instan
ces Size

36/50 2.32G 0.4772 1.004 0.8513
8 640: 100%|██████████| 123/123 [00:25<00:00, 4.88it/s]
R Class Images Instances Box(P

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.98it/s]

663	0.594	all 0.545	420	631	0.477	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

37/50	2.32G	0.4766	1.009	0.8504	
12	640: 100% ██████████ 123/123 [00:24<00:00, 4.95it/ s]				

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.27it/s]

619	0.585	all 0.541	420	631	0.527	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

38/50	2.34G	0.4719	0.9959	0.8491	
13	640: 100% ██████████ 123/123 [00:24<00:00, 5.04it/ s]				

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
4.77it/s]

604	0.596	all 0.549	420	631	0.56	0.
-----	-------	--------------	-----	-----	------	----

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					
12	39/50	2.32G	0.461	0.9672	0.8422	
	640: 100% ██████████ 123/123 [00:24<00:00, 4.97it/s]					
		Class Images Instances Box(P				
R	mAP50 mAP50–95): 100% ██████████ 14/14 [00:02<00:00,					
	4.84it/s]					
602	0.612	all 0.566	420	631	0.578	0.

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					
11	40/50	2.32G	0.4658	0.9623	0.8468	
	640: 100% ██████████ 123/123 [00:24<00:00, 5.00it/s]					
		Class Images Instances Box(P				
R	mAP50 mAP50–95): 100% ██████████ 14/14 [00:02<00:00,					
	4.78it/s]					
614	0.625	all 0.581	420	631	0.594	0.

Closing dataloader mosaic

	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size					

41/50 2.32G 0.3931 0.9007 0.8133
10 640: 100%|██████████| 123/123 [00:24<00:00, 5.02it/
s]
R Class Images Instances Box(P

mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.39it/s]

581 all 420 631 0.547 0.
0.607 0.542

ces Epoch GPU_mem box_loss cls_loss dfl_loss Instan
Size

42/50 2.33G 0.3774 0.8568 0.8166
8 640: 100%|██████████| 123/123 [00:23<00:00, 5.27it/s]
R Class Images Instances Box(P
mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.75it/s]

633 all 420 631 0.546 0.
0.623 0.579

ces Epoch GPU_mem box_loss cls_loss dfl_loss Instan
Size

43/50 2.33G 0.3712 0.8205 0.813
13 640: 100%|██████████| 123/123 [00:23<00:00, 5.33it/
s]
R Class Images Instances Box(P

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.46it/s]

592	0.635	all 0.588	420	631	0.588	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

44/50	2.32G	0.3621	0.8038	0.8087	
10	640: 100% ██████████ 123/123 [00:22<00:00, 5.36it/ s]				

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.03it/s]

641	0.635	all 0.588	420	631	0.581	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

45/50	2.32G	0.3585	0.7847	0.8066	
10	640: 100% ██████████ 123/123 [00:23<00:00, 5.29it/ s]				

R mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.57it/s]

602	0.635	all 0.587	420	631	0.629	0.
-----	-------	--------------	-----	-----	-------	----

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				
11	46/50 640: 100% ██████████ 123/123 [00:22<00:00, 5.37it/s]	2.33G	0.3549	0.7733	0.8068
R	mAP50 5.10it/s]	Class mAP50	Images mAP50-95): 100% ██████████ 14/14 [00:02<00:00,	Instances	Box(P
598	0.646	all 0.603	420	631	0.622
					0.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				
10	47/50 640: 100% ██████████ 123/123 [00:23<00:00, 5.34it/s]	2.32G	0.3481	0.7633	0.8074
R	mAP50 5.13it/s]	Class mAP50	Images mAP50-95): 100% ██████████ 14/14 [00:02<00:00,	Instances	Box(P
627	0.639	all 0.597	420	631	0.574
					0.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instan
ces	Size				

48/50 2.31G 0.347 0.7271 0.8058
11 640: 100%|██████████| 123/123 [00:23<00:00, 5.30it/
s]
R Class Images Instances Box(P
mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.13it/s]

all 420 631 0.574 0.
674 0.648 0.604

Epoch GPU_mem box_loss cls_loss dfl_loss Instant
ces Size

49/50 2.33G 0.343 0.7159 0.7983
10 640: 100%|██████████| 123/123 [00:23<00:00, 5.33it/
s]
R Class Images Instances Box(P
mAP50 mAP50-95): 100%|██████████| 14/14 [00:02<00:00,
5.04it/s]

all 420 631 0.587
0.66 0.65 0.608

Epoch GPU_mem box_loss cls_loss dfl_loss Instant
ces Size

50/50 2.33G 0.3364 0.7049 0.7997
9 640: 100%|██████████| 123/123 [00:22<00:00, 5.40it/s]
R Class Images Instances Box(P

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
5.35it/s]

		all	420	631	0.608	0.
641	0.655	0.614				

50 epochs completed in 0.388 hours.

Optimizer stripped from runs/detect/train/weights/last.pt, 6.2M
B

Optimizer stripped from runs/detect/train/weights/best.pt, 6.2M
B

Validating runs/detect/train/weights/best.pt...

Ultralytics YOLOv8.1.45 🚀 Python-3.10.13 torch-2.0.0.post200 C
UDA:0 (Tesla T4, 14931MiB)

Model summary (fused): 168 layers, 3006428 parameters, 0 gradients, 8.1 GFLOPs

R mAP50 mAP50–95): 100%|██████████| 14/14 [00:02<00:00,
4.79it/s]

		Class	Images	Instances	Box(P	
641	0.655	all	420	631	0.608	0.
		0.613				
		FR-II	420	278	0.716	0.
788	0.794	0.739				
		FR-I	420	87	0.563	0.
563	0.597	0.561				
		FR-X	420	86	0.547	0.
547	0.524	0.49				
		R	420	180	0.607	0.
669	0.706	0.663				

Speed: 1.1ms preprocess, 2.0ms inference, 0.0ms loss, 0.7ms pos
tprocess per image

Results saved to **runs/detect/train**

Out[39]: ultralytics.utils.metrics.DetMetrics object with attributes:

```
ap_class_index: array([0, 1, 2, 3])
box: ultralytics.utils.metrics.Metric object
confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix ob
ject at 0x7f4524f3d3c0>
curves: ['Precision-Recall(B)', 'F1-Confidence(B)', 'Precision-
Confidence(B)', 'Recall-Confidence(B)']
curves_results: [[array([
    0,      0.001001,      0.002002,
0.003003,      0.004004,      0.005005,      0.006006,      0.007007,
0.008008,      0.009009,      0.01001,      0.011011,      0.012012,
0.013013,      0.014014,      0.015015,      0.016016,      0.017017,
0.018018,      0.019019,      0.02002,      0.021021,      0.022022,
0.023023,
    0.024024,      0.025025,      0.026026,      0.027027,
0.028028,      0.029029,      0.03003,      0.031031,      0.032032,
0.033033,      0.034034,      0.035035,      0.036036,      0.037037,
0.038038,      0.039039,      0.04004,      0.041041,      0.042042,
0.043043,      0.044044,      0.045045,      0.046046,      0.047047,
    0.048048,      0.049049,      0.05005,      0.051051,
0.052052,      0.053053,      0.054054,      0.055055,      0.056056,
0.057057,      0.058058,      0.059059,      0.06006,      0.061061,
0.062062,      0.063063,      0.064064,      0.065065,      0.066066,
0.067067,      0.068068,      0.069069,      0.07007,      0.071071,
    0.072072,      0.073073,      0.074074,      0.075075,
0.076076,      0.077077,      0.078078,      0.079079,      0.08008,
0.081081,      0.082082,      0.083083,      0.084084,      0.085085,
0.086086,      0.087087,      0.088088,      0.089089,      0.09009,
```

0.091091,	0.092092,	0.093093,	0.094094,	0.095095,
	0.096096,	0.097097,	0.098098,	0.099099,
0.1001,	0.1011,	0.1021,	0.1031,	0.1041,
0.10511,	0.10611,	0.10711,	0.10811,	0.10911,
0.11011,	0.11111,	0.11211,	0.11311,	0.11411,
0.11512,	0.11612,	0.11712,	0.11812,	0.11912,
	0.12012,	0.12112,	0.12212,	0.12312,
0.12412,	0.12513,	0.12613,	0.12713,	0.12813,
0.12913,	0.13013,	0.13113,	0.13213,	0.13313,
0.13413,	0.13514,	0.13614,	0.13714,	0.13814,
0.13914,	0.14014,	0.14114,	0.14214,	0.14314,
	0.14414,	0.14515,	0.14615,	0.14715,
0.14815,	0.14915,	0.15015,	0.15115,	0.15215,
0.15315,	0.15415,	0.15516,	0.15616,	0.15716,
0.15816,	0.15916,	0.16016,	0.16116,	0.16216,
0.16316,	0.16416,	0.16517,	0.16617,	0.16717,
	0.16817,	0.16917,	0.17017,	0.17117,
0.17217,	0.17317,	0.17417,	0.17518,	0.17618,
0.17718,	0.17818,	0.17918,	0.18018,	0.18118,
0.18218,	0.18318,	0.18418,	0.18519,	0.18619,
0.18719,	0.18819,	0.18919,	0.19019,	0.19119,
	0.19219,	0.19319,	0.19419,	0.1952,
0.1962,	0.1972,	0.1982,	0.1992,	0.2002,
0.2012,	0.2022,	0.2032,	0.2042,	0.20521,
0.20621,	0.20721,	0.20821,	0.20921,	0.21021,
0.21121,	0.21221,	0.21321,	0.21421,	0.21522,
	0.21622,	0.21722,	0.21822,	0.21922,
0.22022,	0.22122,	0.22222,	0.22322,	0.22422,
0.22523,	0.22623,	0.22723,	0.22823,	0.22923,
0.23023,	0.23123,	0.23223,	0.23323,	0.23423,
0.23524,	0.23624,	0.23724,	0.23824,	0.23924,

	0.24024,	0.24124,	0.24224,	0.24324,
0.24424,	0.24525,	0.24625,	0.24725,	0.24825,
0.24925,	0.25025,	0.25125,	0.25225,	0.25325,
0.25425,	0.25526,	0.25626,	0.25726,	0.25826,
0.25926,	0.26026,	0.26126,	0.26226,	0.26326,
	0.26426,	0.26527,	0.26627,	0.26727,
0.26827,	0.26927,	0.27027,	0.27127,	0.27227,
0.27327,	0.27427,	0.27528,	0.27628,	0.27728,
0.27828,	0.27928,	0.28028,	0.28128,	0.28228,
0.28328,	0.28428,	0.28529,	0.28629,	0.28729,
	0.28829,	0.28929,	0.29029,	0.29129,
0.29229,	0.29329,	0.29429,	0.2953,	0.2963,
0.2973,	0.2983,	0.2993,	0.3003,	0.3013,
0.3023,	0.3033,	0.3043,	0.30531,	0.30631,
0.30731,	0.30831,	0.30931,	0.31031,	0.31131,
	0.31231,	0.31331,	0.31431,	0.31532,
0.31632,	0.31732,	0.31832,	0.31932,	0.32032,
0.32132,	0.32232,	0.32332,	0.32432,	0.32533,
0.32633,	0.32733,	0.32833,	0.32933,	0.33033,
0.33133,	0.33233,	0.33333,	0.33433,	0.33534,
	0.33634,	0.33734,	0.33834,	0.33934,
0.34034,	0.34134,	0.34234,	0.34334,	0.34434,
0.34535,	0.34635,	0.34735,	0.34835,	0.34935,
0.35035,	0.35135,	0.35235,	0.35335,	0.35435,
0.35536,	0.35636,	0.35736,	0.35836,	0.35936,
	0.36036,	0.36136,	0.36236,	0.36336,
0.36436,	0.36537,	0.36637,	0.36737,	0.36837,
0.36937,	0.37037,	0.37137,	0.37237,	0.37337,
0.37437,	0.37538,	0.37638,	0.37738,	0.37838,
0.37938,	0.38038,	0.38138,	0.38238,	0.38338,
	0.38438,	0.38539,	0.38639,	0.38739,

0.38839,	0.38939,	0.39039,	0.39139,	0.39239,
0.39339,	0.39439,	0.3954,	0.3964,	0.3974,
0.3984,	0.3994,	0.4004,	0.4014,	0.4024,
0.4034,	0.4044,	0.40541,	0.40641,	0.40741,
	0.40841,	0.40941,	0.41041,	0.41141,
0.41241,	0.41341,	0.41441,	0.41542,	0.41642,
0.41742,	0.41842,	0.41942,	0.42042,	0.42142,
0.42242,	0.42342,	0.42442,	0.42543,	0.42643,
0.42743,	0.42843,	0.42943,	0.43043,	0.43143,
	0.43243,	0.43343,	0.43443,	0.43544,
0.43644,	0.43744,	0.43844,	0.43944,	0.44044,
0.44144,	0.44244,	0.44344,	0.44444,	0.44545,
0.44645,	0.44745,	0.44845,	0.44945,	0.45045,
0.45145,	0.45245,	0.45345,	0.45445,	0.45546,
	0.45646,	0.45746,	0.45846,	0.45946,
0.46046,	0.46146,	0.46246,	0.46346,	0.46446,
0.46547,	0.46647,	0.46747,	0.46847,	0.46947,
0.47047,	0.47147,	0.47247,	0.47347,	0.47447,
0.47548,	0.47648,	0.47748,	0.47848,	0.47948,
	0.48048,	0.48148,	0.48248,	0.48348,
0.48448,	0.48549,	0.48649,	0.48749,	0.48849,
0.48949,	0.49049,	0.49149,	0.49249,	0.49349,
0.49449,	0.4955,	0.4965,	0.4975,	0.4985,
0.4995,	0.5005,	0.5015,	0.5025,	0.5035,
	0.5045,	0.50551,	0.50651,	0.50751,
0.50851,	0.50951,	0.51051,	0.51151,	0.51251,
0.51351,	0.51451,	0.51552,	0.51652,	0.51752,
0.51852,	0.51952,	0.52052,	0.52152,	0.52252,
0.52352,	0.52452,	0.52553,	0.52653,	0.52753,
	0.52853,	0.52953,	0.53053,	0.53153,
0.53253,	0.53353,	0.53453,	0.53554,	0.53654,

0.53754,	0.53854,	0.53954,	0.54054,	0.54154,
0.54254,	0.54354,	0.54454,	0.54555,	0.54655,
0.54755,	0.54855,	0.54955,	0.55055,	0.55155,
	0.55255,	0.55355,	0.55455,	0.55556,
0.55656,	0.55756,	0.55856,	0.55956,	0.56056,
0.56156,	0.56256,	0.56356,	0.56456,	0.56557,
0.56657,	0.56757,	0.56857,	0.56957,	0.57057,
0.57157,	0.57257,	0.57357,	0.57457,	0.57558,
	0.57658,	0.57758,	0.57858,	0.57958,
0.58058,	0.58158,	0.58258,	0.58358,	0.58458,
0.58559,	0.58659,	0.58759,	0.58859,	0.58959,
0.59059,	0.59159,	0.59259,	0.59359,	0.59459,
0.5956,	0.5966,	0.5976,	0.5986,	0.5996,
	0.6006,	0.6016,	0.6026,	0.6036,
0.6046,	0.60561,	0.60661,	0.60761,	0.60861,
0.60961,	0.61061,	0.61161,	0.61261,	0.61361,
0.61461,	0.61562,	0.61662,	0.61762,	0.61862,
0.61962,	0.62062,	0.62162,	0.62262,	0.62362,
	0.62462,	0.62563,	0.62663,	0.62763,
0.62863,	0.62963,	0.63063,	0.63163,	0.63263,
0.63363,	0.63463,	0.63564,	0.63664,	0.63764,
0.63864,	0.63964,	0.64064,	0.64164,	0.64264,
0.64364,	0.64464,	0.64565,	0.64665,	0.64765,
	0.64865,	0.64965,	0.65065,	0.65165,
0.65265,	0.65365,	0.65465,	0.65566,	0.65666,
0.65766,	0.65866,	0.65966,	0.66066,	0.66166,
0.66266,	0.66366,	0.66466,	0.66567,	0.66667,
0.66767,	0.66867,	0.66967,	0.67067,	0.67167,
	0.67267,	0.67367,	0.67467,	0.67568,
0.67668,	0.67768,	0.67868,	0.67968,	0.68068,
0.68168,	0.68268,	0.68368,	0.68468,	0.68569,

0.68669,	0.68769,	0.68869,	0.68969,	0.69069,
0.69169,	0.69269,	0.69369,	0.69469,	0.6957,
	0.6967,	0.6977,	0.6987,	0.6997,
0.7007,	0.7017,	0.7027,	0.7037,	0.7047,
0.70571,	0.70671,	0.70771,	0.70871,	0.70971,
0.71071,	0.71171,	0.71271,	0.71371,	0.71471,
0.71572,	0.71672,	0.71772,	0.71872,	0.71972,
	0.72072,	0.72172,	0.72272,	0.72372,
0.72472,	0.72573,	0.72673,	0.72773,	0.72873,
0.72973,	0.73073,	0.73173,	0.73273,	0.73373,
0.73473,	0.73574,	0.73674,	0.73774,	0.73874,
0.73974,	0.74074,	0.74174,	0.74274,	0.74374,
	0.74474,	0.74575,	0.74675,	0.74775,
0.74875,	0.74975,	0.75075,	0.75175,	0.75275,
0.75375,	0.75475,	0.75576,	0.75676,	0.75776,
0.75876,	0.75976,	0.76076,	0.76176,	0.76276,
0.76376,	0.76476,	0.76577,	0.76677,	0.76777,
	0.76877,	0.76977,	0.77077,	0.77177,
0.77277,	0.77377,	0.77477,	0.77578,	0.77678,
0.77778,	0.77878,	0.77978,	0.78078,	0.78178,
0.78278,	0.78378,	0.78478,	0.78579,	0.78679,
0.78779,	0.78879,	0.78979,	0.79079,	0.79179,
	0.79279,	0.79379,	0.79479,	0.7958,
0.7968,	0.7978,	0.7988,	0.7998,	0.8008,
0.8018,	0.8028,	0.8038,	0.8048,	0.80581,
0.80681,	0.80781,	0.80881,	0.80981,	0.81081,
0.81181,	0.81281,	0.81381,	0.81481,	0.81582,
	0.81682,	0.81782,	0.81882,	0.81982,
0.82082,	0.82182,	0.82282,	0.82382,	0.82482,
0.82583,	0.82683,	0.82783,	0.82883,	0.82983,
0.83083,	0.83183,	0.83283,	0.83383,	0.83483,

0.83584,	0.83684,	0.83784,	0.83884,	0.83984,
	0.84084,	0.84184,	0.84284,	0.84384,
0.84484,	0.84585,	0.84685,	0.84785,	0.84885,
0.84985,	0.85085,	0.85185,	0.85285,	0.85385,
0.85485,	0.85586,	0.85686,	0.85786,	0.85886,
0.85986,	0.86086,	0.86186,	0.86286,	0.86386,
	0.86486,	0.86587,	0.86687,	0.86787,
0.86887,	0.86987,	0.87087,	0.87187,	0.87287,
0.87387,	0.87487,	0.87588,	0.87688,	0.87788,
0.87888,	0.87988,	0.88088,	0.88188,	0.88288,
0.88388,	0.88488,	0.88589,	0.88689,	0.88789,
	0.88889,	0.88989,	0.89089,	0.89189,
0.89289,	0.89389,	0.89489,	0.8959,	0.8969,
0.8979,	0.8989,	0.8999,	0.9009,	0.9019,
0.9029,	0.9039,	0.9049,	0.90591,	0.90691,
0.90791,	0.90891,	0.90991,	0.91091,	0.91191,
	0.91291,	0.91391,	0.91491,	0.91592,
0.91692,	0.91792,	0.91892,	0.91992,	0.92092,
0.92192,	0.92292,	0.92392,	0.92492,	0.92593,
0.92693,	0.92793,	0.92893,	0.92993,	0.93093,
0.93193,	0.93293,	0.93393,	0.93493,	0.93594,
	0.93694,	0.93794,	0.93894,	0.93994,
0.94094,	0.94194,	0.94294,	0.94394,	0.94494,
0.94595,	0.94695,	0.94795,	0.94895,	0.94995,
0.95095,	0.95195,	0.95295,	0.95395,	0.95495,
0.95596,	0.95696,	0.95796,	0.95896,	0.95996,
	0.96096,	0.96196,	0.96296,	0.96396,
0.96496,	0.96597,	0.96697,	0.96797,	0.96897,
0.96997,	0.97097,	0.97197,	0.97297,	0.97397,
0.97497,	0.97598,	0.97698,	0.97798,	0.97898,
0.97998,	0.98098,	0.98198,	0.98298,	0.98398,

0.98498, 0.98599, 0.98699, 0.98799,
0.98899, 0.98999, 0.99099, 0.99199, 0.99299,
0.99399, 0.99499, 0.996, 0.997, 0.998,
0.999, 1]), array([[0.95833, 0.95833, 0.9
5833, ..., 0.021748, 0.010874,
[1, 1, 1, 1, ..., 0.1201
7, 0.12017,
[1, 1, 1, 1, ..., 0.00704
8, 0.003524,
[1, 1, 1, 1, ..., 0.009426
5, 0.0047132,
[0, 0.001001, 0.002002, 0.003003, 0.0040
04, 0.005005, 0.006006, 0.007007, 0.008008, 0.00
9009, 0.01001, 0.011011, 0.012012, 0.013013, 0.
014014, 0.015015, 0.016016, 0.017017, 0.018018,
0.019019, 0.02002, 0.021021, 0.022022, 0.023023,
0.024024, 0.025025, 0.026026, 0.027027,
0.028028, 0.029029, 0.03003, 0.031031, 0.032032,
0.033033, 0.034034, 0.035035, 0.036036, 0.037037,
0.038038, 0.039039, 0.04004, 0.041041, 0.042042,
0.043043, 0.044044, 0.045045, 0.046046, 0.047047,
0.048048, 0.049049, 0.05005, 0.051051,
0.052052, 0.053053, 0.054054, 0.055055, 0.056056,
0.057057, 0.058058, 0.059059, 0.06006, 0.061061,
0.062062, 0.063063, 0.064064, 0.065065, 0.066066,
0.067067, 0.068068, 0.069069, 0.07007, 0.071071,
0.072072, 0.073073, 0.074074, 0.075075,
0.076076, 0.077077, 0.078078, 0.079079, 0.08008,
0.081081, 0.082082, 0.083083, 0.084084, 0.085085,
0.086086, 0.087087, 0.088088, 0.089089, 0.09009,
0.091091, 0.092092, 0.093093, 0.094094, 0.095095,

	0.096096,	0.097097,	0.098098,	0.099099,
0.1001,	0.1011,	0.1021,	0.1031,	0.1041,
0.10511,	0.10611,	0.10711,	0.10811,	0.10911,
0.11011,	0.11111,	0.11211,	0.11311,	0.11411,
0.11512,	0.11612,	0.11712,	0.11812,	0.11912,
	0.12012,	0.12112,	0.12212,	0.12312,
0.12412,	0.12513,	0.12613,	0.12713,	0.12813,
0.12913,	0.13013,	0.13113,	0.13213,	0.13313,
0.13413,	0.13514,	0.13614,	0.13714,	0.13814,
0.13914,	0.14014,	0.14114,	0.14214,	0.14314,
	0.14414,	0.14515,	0.14615,	0.14715,
0.14815,	0.14915,	0.15015,	0.15115,	0.15215,
0.15315,	0.15415,	0.15516,	0.15616,	0.15716,
0.15816,	0.15916,	0.16016,	0.16116,	0.16216,
0.16316,	0.16416,	0.16517,	0.16617,	0.16717,
	0.16817,	0.16917,	0.17017,	0.17117,
0.17217,	0.17317,	0.17417,	0.17518,	0.17618,
0.17718,	0.17818,	0.17918,	0.18018,	0.18118,
0.18218,	0.18318,	0.18418,	0.18519,	0.18619,
0.18719,	0.18819,	0.18919,	0.19019,	0.19119,
	0.19219,	0.19319,	0.19419,	0.1952,
0.1962,	0.1972,	0.1982,	0.1992,	0.2002,
0.2012,	0.2022,	0.2032,	0.2042,	0.20521,
0.20621,	0.20721,	0.20821,	0.20921,	0.21021,
0.21121,	0.21221,	0.21321,	0.21421,	0.21522,
	0.21622,	0.21722,	0.21822,	0.21922,
0.22022,	0.22122,	0.22222,	0.22322,	0.22422,
0.22523,	0.22623,	0.22723,	0.22823,	0.22923,
0.23023,	0.23123,	0.23223,	0.23323,	0.23423,
0.23524,	0.23624,	0.23724,	0.23824,	0.23924,
	0.24024,	0.24124,	0.24224,	0.24324,

0.24424,	0.24525,	0.24625,	0.24725,	0.24825,
0.24925,	0.25025,	0.25125,	0.25225,	0.25325,
0.25425,	0.25526,	0.25626,	0.25726,	0.25826,
0.25926,	0.26026,	0.26126,	0.26226,	0.26326,
	0.26426,	0.26527,	0.26627,	0.26727,
0.26827,	0.26927,	0.27027,	0.27127,	0.27227,
0.27327,	0.27427,	0.27528,	0.27628,	0.27728,
0.27828,	0.27928,	0.28028,	0.28128,	0.28228,
0.28328,	0.28428,	0.28529,	0.28629,	0.28729,
	0.28829,	0.28929,	0.29029,	0.29129,
0.29229,	0.29329,	0.29429,	0.2953,	0.2963,
0.2973,	0.2983,	0.2993,	0.3003,	0.3013,
0.3023,	0.3033,	0.3043,	0.30531,	0.30631,
0.30731,	0.30831,	0.30931,	0.31031,	0.31131,
	0.31231,	0.31331,	0.31431,	0.31532,
0.31632,	0.31732,	0.31832,	0.31932,	0.32032,
0.32132,	0.32232,	0.32332,	0.32432,	0.32533,
0.32633,	0.32733,	0.32833,	0.32933,	0.33033,
0.33133,	0.33233,	0.33333,	0.33433,	0.33534,
	0.33634,	0.33734,	0.33834,	0.33934,
0.34034,	0.34134,	0.34234,	0.34334,	0.34434,
0.34535,	0.34635,	0.34735,	0.34835,	0.34935,
0.35035,	0.35135,	0.35235,	0.35335,	0.35435,
0.35536,	0.35636,	0.35736,	0.35836,	0.35936,
	0.36036,	0.36136,	0.36236,	0.36336,
0.36436,	0.36537,	0.36637,	0.36737,	0.36837,
0.36937,	0.37037,	0.37137,	0.37237,	0.37337,
0.37437,	0.37538,	0.37638,	0.37738,	0.37838,
0.37938,	0.38038,	0.38138,	0.38238,	0.38338,
	0.38438,	0.38539,	0.38639,	0.38739,
0.38839,	0.38939,	0.39039,	0.39139,	0.39239,

0.39339,	0.39439,	0.3954,	0.3964,	0.3974,
0.3984,	0.3994,	0.4004,	0.4014,	0.4024,
0.4034,	0.4044,	0.40541,	0.40641,	0.40741,
	0.40841,	0.40941,	0.41041,	0.41141,
0.41241,	0.41341,	0.41441,	0.41542,	0.41642,
0.41742,	0.41842,	0.41942,	0.42042,	0.42142,
0.42242,	0.42342,	0.42442,	0.42543,	0.42643,
0.42743,	0.42843,	0.42943,	0.43043,	0.43143,
	0.43243,	0.43343,	0.43443,	0.43544,
0.43644,	0.43744,	0.43844,	0.43944,	0.44044,
0.44144,	0.44244,	0.44344,	0.44444,	0.44545,
0.44645,	0.44745,	0.44845,	0.44945,	0.45045,
0.45145,	0.45245,	0.45345,	0.45445,	0.45546,
	0.45646,	0.45746,	0.45846,	0.45946,
0.46046,	0.46146,	0.46246,	0.46346,	0.46446,
0.46547,	0.46647,	0.46747,	0.46847,	0.46947,
0.47047,	0.47147,	0.47247,	0.47347,	0.47447,
0.47548,	0.47648,	0.47748,	0.47848,	0.47948,
	0.48048,	0.48148,	0.48248,	0.48348,
0.48448,	0.48549,	0.48649,	0.48749,	0.48849,
0.48949,	0.49049,	0.49149,	0.49249,	0.49349,
0.49449,	0.4955,	0.4965,	0.4975,	0.4985,
0.4995,	0.5005,	0.5015,	0.5025,	0.5035,
	0.5045,	0.50551,	0.50651,	0.50751,
0.50851,	0.50951,	0.51051,	0.51151,	0.51251,
0.51351,	0.51451,	0.51552,	0.51652,	0.51752,
0.51852,	0.51952,	0.52052,	0.52152,	0.52252,
0.52352,	0.52452,	0.52553,	0.52653,	0.52753,
	0.52853,	0.52953,	0.53053,	0.53153,
0.53253,	0.53353,	0.53453,	0.53554,	0.53654,
0.53754,	0.53854,	0.53954,	0.54054,	0.54154,

0.54254,	0.54354,	0.54454,	0.54555,	0.54655,
0.54755,	0.54855,	0.54955,	0.55055,	0.55155,
	0.55255,	0.55355,	0.55455,	0.55556,
0.55656,	0.55756,	0.55856,	0.55956,	0.56056,
0.56156,	0.56256,	0.56356,	0.56456,	0.56557,
0.56657,	0.56757,	0.56857,	0.56957,	0.57057,
0.57157,	0.57257,	0.57357,	0.57457,	0.57558,
	0.57658,	0.57758,	0.57858,	0.57958,
0.58058,	0.58158,	0.58258,	0.58358,	0.58458,
0.58559,	0.58659,	0.58759,	0.58859,	0.58959,
0.59059,	0.59159,	0.59259,	0.59359,	0.59459,
0.5956,	0.5966,	0.5976,	0.5986,	0.5996,
	0.6006,	0.6016,	0.6026,	0.6036,
0.6046,	0.60561,	0.60661,	0.60761,	0.60861,
0.60961,	0.61061,	0.61161,	0.61261,	0.61361,
0.61461,	0.61562,	0.61662,	0.61762,	0.61862,
0.61962,	0.62062,	0.62162,	0.62262,	0.62362,
	0.62462,	0.62563,	0.62663,	0.62763,
0.62863,	0.62963,	0.63063,	0.63163,	0.63263,
0.63363,	0.63463,	0.63564,	0.63664,	0.63764,
0.63864,	0.63964,	0.64064,	0.64164,	0.64264,
0.64364,	0.64464,	0.64565,	0.64665,	0.64765,
	0.64865,	0.64965,	0.65065,	0.65165,
0.65265,	0.65365,	0.65465,	0.65566,	0.65666,
0.65766,	0.65866,	0.65966,	0.66066,	0.66166,
0.66266,	0.66366,	0.66466,	0.66567,	0.66667,
0.66767,	0.66867,	0.66967,	0.67067,	0.67167,
	0.67267,	0.67367,	0.67467,	0.67568,
0.67668,	0.67768,	0.67868,	0.67968,	0.68068,
0.68168,	0.68268,	0.68368,	0.68468,	0.68569,
0.68669,	0.68769,	0.68869,	0.68969,	0.69069,

0.69169,	0.69269,	0.69369,	0.69469,	0.6957,
	0.6967,	0.6977,	0.6987,	0.6997,
0.7007,	0.7017,	0.7027,	0.7037,	0.7047,
0.70571,	0.70671,	0.70771,	0.70871,	0.70971,
0.71071,	0.71171,	0.71271,	0.71371,	0.71471,
0.71572,	0.71672,	0.71772,	0.71872,	0.71972,
	0.72072,	0.72172,	0.72272,	0.72372,
0.72472,	0.72573,	0.72673,	0.72773,	0.72873,
0.72973,	0.73073,	0.73173,	0.73273,	0.73373,
0.73473,	0.73574,	0.73674,	0.73774,	0.73874,
0.73974,	0.74074,	0.74174,	0.74274,	0.74374,
	0.74474,	0.74575,	0.74675,	0.74775,
0.74875,	0.74975,	0.75075,	0.75175,	0.75275,
0.75375,	0.75475,	0.75576,	0.75676,	0.75776,
0.75876,	0.75976,	0.76076,	0.76176,	0.76276,
0.76376,	0.76476,	0.76577,	0.76677,	0.76777,
	0.76877,	0.76977,	0.77077,	0.77177,
0.77277,	0.77377,	0.77477,	0.77578,	0.77678,
0.77778,	0.77878,	0.77978,	0.78078,	0.78178,
0.78278,	0.78378,	0.78478,	0.78579,	0.78679,
0.78779,	0.78879,	0.78979,	0.79079,	0.79179,
	0.79279,	0.79379,	0.79479,	0.7958,
0.7968,	0.7978,	0.7988,	0.7998,	0.8008,
0.8018,	0.8028,	0.8038,	0.8048,	0.80581,
0.80681,	0.80781,	0.80881,	0.80981,	0.81081,
0.81181,	0.81281,	0.81381,	0.81481,	0.81582,
	0.81682,	0.81782,	0.81882,	0.81982,
0.82082,	0.82182,	0.82282,	0.82382,	0.82482,
0.82583,	0.82683,	0.82783,	0.82883,	0.82983,
0.83083,	0.83183,	0.83283,	0.83383,	0.83483,
0.83584,	0.83684,	0.83784,	0.83884,	0.83984,

	0.84084,	0.84184,	0.84284,	0.84384,
0.84484,	0.84585,	0.84685,	0.84785,	0.84885,
0.84985,	0.85085,	0.85185,	0.85285,	0.85385,
0.85485,	0.85586,	0.85686,	0.85786,	0.85886,
0.85986,	0.86086,	0.86186,	0.86286,	0.86386,
	0.86486,	0.86587,	0.86687,	0.86787,
0.86887,	0.86987,	0.87087,	0.87187,	0.87287,
0.87387,	0.87487,	0.87588,	0.87688,	0.87788,
0.87888,	0.87988,	0.88088,	0.88188,	0.88288,
0.88388,	0.88488,	0.88589,	0.88689,	0.88789,
	0.88889,	0.88989,	0.89089,	0.89189,
0.89289,	0.89389,	0.89489,	0.8959,	0.8969,
0.8979,	0.8989,	0.8999,	0.9009,	0.9019,
0.9029,	0.9039,	0.9049,	0.90591,	0.90691,
0.90791,	0.90891,	0.90991,	0.91091,	0.91191,
	0.91291,	0.91391,	0.91491,	0.91592,
0.91692,	0.91792,	0.91892,	0.91992,	0.92092,
0.92192,	0.92292,	0.92392,	0.92492,	0.92593,
0.92693,	0.92793,	0.92893,	0.92993,	0.93093,
0.93193,	0.93293,	0.93393,	0.93493,	0.93594,
	0.93694,	0.93794,	0.93894,	0.93994,
0.94094,	0.94194,	0.94294,	0.94394,	0.94494,
0.94595,	0.94695,	0.94795,	0.94895,	0.94995,
0.95095,	0.95195,	0.95295,	0.95395,	0.95495,
0.95596,	0.95696,	0.95796,	0.95896,	0.95996,
	0.96096,	0.96196,	0.96296,	0.96396,
0.96496,	0.96597,	0.96697,	0.96797,	0.96897,
0.96997,	0.97097,	0.97197,	0.97297,	0.97397,
0.97497,	0.97598,	0.97698,	0.97798,	0.97898,
0.97998,	0.98098,	0.98198,	0.98298,	0.98398,
	0.98498,	0.98599,	0.98699,	0.98799,

```
0.98899,      0.98999,      0.99099,      0.99199,      0.99299,
0.99399,      0.99499,      0.996,        0.997,        0.998,
0.999,         1]), array([[ 0.26982,      0.26982,      0.3
3097, ...,
          0,          0,          0],
          [ 0.16667,      0.16667,      0.188, ...,
          0,          0],
          [ 0.15108,      0.15108,      0.17655, ...,
          0,          0],
          [ 0.23057,      0.23057,      0.27164, ...,
          0,          0],
          [array([ 0.001001,      0.002002,      0.003003,      0.004004,      0.005
005,      0.006006,      0.007007,      0.008008,      0.009009,      0.
01001,      0.011011,      0.012012,      0.013013,      0.014014,
0.015015,      0.016016,      0.017017,      0.018018,      0.019019,
0.02002,      0.021021,      0.022022,      0.023023,
          0.024024,      0.025025,      0.026026,      0.027027,
          0.028028,      0.029029,      0.03003,       0.031031,      0.032032,
          0.033033,      0.034034,      0.035035,      0.036036,      0.037037,
          0.038038,      0.039039,      0.04004,       0.041041,      0.042042,
          0.043043,      0.044044,      0.045045,      0.046046,      0.047047,
          0.048048,      0.049049,      0.05005,       0.051051,
          0.052052,      0.053053,      0.054054,      0.055055,      0.056056,
          0.057057,      0.058058,      0.059059,      0.06006,       0.061061,
          0.062062,      0.063063,      0.064064,      0.065065,      0.066066,
          0.067067,      0.068068,      0.069069,      0.07007,       0.071071,
          0.072072,      0.073073,      0.074074,      0.075075,
          0.076076,      0.077077,      0.078078,      0.079079,      0.08008,
          0.081081,      0.082082,      0.083083,      0.084084,      0.085085,
          0.086086,      0.087087,      0.088088,      0.089089,      0.09009,
          0.091091,      0.092092,      0.093093,      0.094094,      0.095095,
          0.096096,      0.097097,      0.098098,      0.099099,
```

0.1001,	0.1011,	0.1021,	0.1031,	0.1041,
0.10511,	0.10611,	0.10711,	0.10811,	0.10911,
0.11011,	0.11111,	0.11211,	0.11311,	0.11411,
0.11512,	0.11612,	0.11712,	0.11812,	0.11912,
	0.12012,	0.12112,	0.12212,	0.12312,
0.12412,	0.12513,	0.12613,	0.12713,	0.12813,
0.12913,	0.13013,	0.13113,	0.13213,	0.13313,
0.13413,	0.13514,	0.13614,	0.13714,	0.13814,
0.13914,	0.14014,	0.14114,	0.14214,	0.14314,
	0.14414,	0.14515,	0.14615,	0.14715,
0.14815,	0.14915,	0.15015,	0.15115,	0.15215,
0.15315,	0.15415,	0.15516,	0.15616,	0.15716,
0.15816,	0.15916,	0.16016,	0.16116,	0.16216,
0.16316,	0.16416,	0.16517,	0.16617,	0.16717,
	0.16817,	0.16917,	0.17017,	0.17117,
0.17217,	0.17317,	0.17417,	0.17518,	0.17618,
0.17718,	0.17818,	0.17918,	0.18018,	0.18118,
0.18218,	0.18318,	0.18418,	0.18519,	0.18619,
0.18719,	0.18819,	0.18919,	0.19019,	0.19119,
	0.19219,	0.19319,	0.19419,	0.1952,
0.1962,	0.1972,	0.1982,	0.1992,	0.2002,
0.2012,	0.2022,	0.2032,	0.2042,	0.20521,
0.20621,	0.20721,	0.20821,	0.20921,	0.21021,
0.21121,	0.21221,	0.21321,	0.21421,	0.21522,
	0.21622,	0.21722,	0.21822,	0.21922,
0.22022,	0.22122,	0.22222,	0.22322,	0.22422,
0.22523,	0.22623,	0.22723,	0.22823,	0.22923,
0.23023,	0.23123,	0.23223,	0.23323,	0.23423,
0.23524,	0.23624,	0.23724,	0.23824,	0.23924,
	0.24024,	0.24124,	0.24224,	0.24324,
0.24424,	0.24525,	0.24625,	0.24725,	0.24825,

0.24925,	0.25025,	0.25125,	0.25225,	0.25325,
0.25425,	0.25526,	0.25626,	0.25726,	0.25826,
0.25926,	0.26026,	0.26126,	0.26226,	0.26326,
	0.26426,	0.26527,	0.26627,	0.26727,
0.26827,	0.26927,	0.27027,	0.27127,	0.27227,
0.27327,	0.27427,	0.27528,	0.27628,	0.27728,
0.27828,	0.27928,	0.28028,	0.28128,	0.28228,
0.28328,	0.28428,	0.28529,	0.28629,	0.28729,
	0.28829,	0.28929,	0.29029,	0.29129,
0.29229,	0.29329,	0.29429,	0.2953,	0.2963,
0.2973,	0.2983,	0.2993,	0.3003,	0.3013,
0.3023,	0.3033,	0.3043,	0.30531,	0.30631,
0.30731,	0.30831,	0.30931,	0.31031,	0.31131,
	0.31231,	0.31331,	0.31431,	0.31532,
0.31632,	0.31732,	0.31832,	0.31932,	0.32032,
0.32132,	0.32232,	0.32332,	0.32432,	0.32533,
0.32633,	0.32733,	0.32833,	0.32933,	0.33033,
0.33133,	0.33233,	0.33333,	0.33433,	0.33534,
	0.33634,	0.33734,	0.33834,	0.33934,
0.34034,	0.34134,	0.34234,	0.34334,	0.34434,
0.34535,	0.34635,	0.34735,	0.34835,	0.34935,
0.35035,	0.35135,	0.35235,	0.35335,	0.35435,
0.35536,	0.35636,	0.35736,	0.35836,	0.35936,
	0.36036,	0.36136,	0.36236,	0.36336,
0.36436,	0.36537,	0.36637,	0.36737,	0.36837,
0.36937,	0.37037,	0.37137,	0.37237,	0.37337,
0.37437,	0.37538,	0.37638,	0.37738,	0.37838,
0.37938,	0.38038,	0.38138,	0.38238,	0.38338,
	0.38438,	0.38539,	0.38639,	0.38739,
0.38839,	0.38939,	0.39039,	0.39139,	0.39239,
0.39339,	0.39439,	0.3954,	0.3964,	0.3974,

0.3984,	0.3994,	0.4004,	0.4014,	0.4024,
0.4034,	0.4044,	0.40541,	0.40641,	0.40741,
	0.40841,	0.40941,	0.41041,	0.41141,
0.41241,	0.41341,	0.41441,	0.41542,	0.41642,
0.41742,	0.41842,	0.41942,	0.42042,	0.42142,
0.42242,	0.42342,	0.42442,	0.42543,	0.42643,
0.42743,	0.42843,	0.42943,	0.43043,	0.43143,
	0.43243,	0.43343,	0.43443,	0.43544,
0.43644,	0.43744,	0.43844,	0.43944,	0.44044,
0.44144,	0.44244,	0.44344,	0.44444,	0.44545,
0.44645,	0.44745,	0.44845,	0.44945,	0.45045,
0.45145,	0.45245,	0.45345,	0.45445,	0.45546,
	0.45646,	0.45746,	0.45846,	0.45946,
0.46046,	0.46146,	0.46246,	0.46346,	0.46446,
0.46547,	0.46647,	0.46747,	0.46847,	0.46947,
0.47047,	0.47147,	0.47247,	0.47347,	0.47447,
0.47548,	0.47648,	0.47748,	0.47848,	0.47948,
	0.48048,	0.48148,	0.48248,	0.48348,
0.48448,	0.48549,	0.48649,	0.48749,	0.48849,
0.48949,	0.49049,	0.49149,	0.49249,	0.49349,
0.49449,	0.4955,	0.4965,	0.4975,	0.4985,
0.4995,	0.5005,	0.5015,	0.5025,	0.5035,
	0.5045,	0.50551,	0.50651,	0.50751,
0.50851,	0.50951,	0.51051,	0.51151,	0.51251,
0.51351,	0.51451,	0.51552,	0.51652,	0.51752,
0.51852,	0.51952,	0.52052,	0.52152,	0.52252,
0.52352,	0.52452,	0.52553,	0.52653,	0.52753,
	0.52853,	0.52953,	0.53053,	0.53153,
0.53253,	0.53353,	0.53453,	0.53554,	0.53654,
0.53754,	0.53854,	0.53954,	0.54054,	0.54154,
0.54254,	0.54354,	0.54454,	0.54555,	0.54655,

0.54755,	0.54855,	0.54955,	0.55055,	0.55155,
	0.55255,	0.55355,	0.55455,	0.55556,
0.55656,	0.55756,	0.55856,	0.55956,	0.56056,
0.56156,	0.56256,	0.56356,	0.56456,	0.56557,
0.56657,	0.56757,	0.56857,	0.56957,	0.57057,
0.57157,	0.57257,	0.57357,	0.57457,	0.57558,
	0.57658,	0.57758,	0.57858,	0.57958,
0.58058,	0.58158,	0.58258,	0.58358,	0.58458,
0.58559,	0.58659,	0.58759,	0.58859,	0.58959,
0.59059,	0.59159,	0.59259,	0.59359,	0.59459,
0.5956,	0.5966,	0.5976,	0.5986,	0.5996,
	0.6006,	0.6016,	0.6026,	0.6036,
0.6046,	0.60561,	0.60661,	0.60761,	0.60861,
0.60961,	0.61061,	0.61161,	0.61261,	0.61361,
0.61461,	0.61562,	0.61662,	0.61762,	0.61862,
0.61962,	0.62062,	0.62162,	0.62262,	0.62362,
	0.62462,	0.62563,	0.62663,	0.62763,
0.62863,	0.62963,	0.63063,	0.63163,	0.63263,
0.63363,	0.63463,	0.63564,	0.63664,	0.63764,
0.63864,	0.63964,	0.64064,	0.64164,	0.64264,
0.64364,	0.64464,	0.64565,	0.64665,	0.64765,
	0.64865,	0.64965,	0.65065,	0.65165,
0.65265,	0.65365,	0.65465,	0.65566,	0.65666,
0.65766,	0.65866,	0.65966,	0.66066,	0.66166,
0.66266,	0.66366,	0.66466,	0.66567,	0.66667,
0.66767,	0.66867,	0.66967,	0.67067,	0.67167,
	0.67267,	0.67367,	0.67467,	0.67568,
0.67668,	0.67768,	0.67868,	0.67968,	0.68068,
0.68168,	0.68268,	0.68368,	0.68468,	0.68569,
0.68669,	0.68769,	0.68869,	0.68969,	0.69069,
0.69169,	0.69269,	0.69369,	0.69469,	0.6957,

	0.6967,	0.6977,	0.6987,	0.6997,
0.7007,	0.7017,	0.7027,	0.7037,	0.7047,
0.70571,	0.70671,	0.70771,	0.70871,	0.70971,
0.71071,	0.71171,	0.71271,	0.71371,	0.71471,
0.71572,	0.71672,	0.71772,	0.71872,	0.71972,
	0.72072,	0.72172,	0.72272,	0.72372,
0.72472,	0.72573,	0.72673,	0.72773,	0.72873,
0.72973,	0.73073,	0.73173,	0.73273,	0.73373,
0.73473,	0.73574,	0.73674,	0.73774,	0.73874,
0.73974,	0.74074,	0.74174,	0.74274,	0.74374,
	0.74474,	0.74575,	0.74675,	0.74775,
0.74875,	0.74975,	0.75075,	0.75175,	0.75275,
0.75375,	0.75475,	0.75576,	0.75676,	0.75776,
0.75876,	0.75976,	0.76076,	0.76176,	0.76276,
0.76376,	0.76476,	0.76577,	0.76677,	0.76777,
	0.76877,	0.76977,	0.77077,	0.77177,
0.77277,	0.77377,	0.77477,	0.77578,	0.77678,
0.77778,	0.77878,	0.77978,	0.78078,	0.78178,
0.78278,	0.78378,	0.78478,	0.78579,	0.78679,
0.78779,	0.78879,	0.78979,	0.79079,	0.79179,
	0.79279,	0.79379,	0.79479,	0.7958,
0.7968,	0.7978,	0.7988,	0.7998,	0.8008,
0.8018,	0.8028,	0.8038,	0.8048,	0.80581,
0.80681,	0.80781,	0.80881,	0.80981,	0.81081,
0.81181,	0.81281,	0.81381,	0.81481,	0.81582,
	0.81682,	0.81782,	0.81882,	0.81982,
0.82082,	0.82182,	0.82282,	0.82382,	0.82482,
0.82583,	0.82683,	0.82783,	0.82883,	0.82983,
0.83083,	0.83183,	0.83283,	0.83383,	0.83483,
0.83584,	0.83684,	0.83784,	0.83884,	0.83984,
	0.84084,	0.84184,	0.84284,	0.84384,

0.84484,	0.84585,	0.84685,	0.84785,	0.84885,
0.84985,	0.85085,	0.85185,	0.85285,	0.85385,
0.85485,	0.85586,	0.85686,	0.85786,	0.85886,
0.85986,	0.86086,	0.86186,	0.86286,	0.86386,
	0.86486,	0.86587,	0.86687,	0.86787,
0.86887,	0.86987,	0.87087,	0.87187,	0.87287,
0.87387,	0.87487,	0.87588,	0.87688,	0.87788,
0.87888,	0.87988,	0.88088,	0.88188,	0.88288,
0.88388,	0.88488,	0.88589,	0.88689,	0.88789,
	0.88889,	0.88989,	0.89089,	0.89189,
0.89289,	0.89389,	0.89489,	0.8959,	0.8969,
0.8979,	0.8989,	0.8999,	0.9009,	0.9019,
0.9029,	0.9039,	0.9049,	0.90591,	0.90691,
0.90791,	0.90891,	0.90991,	0.91091,	0.91191,
	0.91291,	0.91391,	0.91491,	0.91592,
0.91692,	0.91792,	0.91892,	0.91992,	0.92092,
0.92192,	0.92292,	0.92392,	0.92492,	0.92593,
0.92693,	0.92793,	0.92893,	0.92993,	0.93093,
0.93193,	0.93293,	0.93393,	0.93493,	0.93594,
	0.93694,	0.93794,	0.93894,	0.93994,
0.94094,	0.94194,	0.94294,	0.94394,	0.94494,
0.94595,	0.94695,	0.94795,	0.94895,	0.94995,
0.95095,	0.95195,	0.95295,	0.95395,	0.95495,
0.95596,	0.95696,	0.95796,	0.95896,	0.95996,
	0.96096,	0.96196,	0.96296,	0.96396,
0.96496,	0.96597,	0.96697,	0.96797,	0.96897,
0.96997,	0.97097,	0.97197,	0.97297,	0.97397,
0.97497,	0.97598,	0.97698,	0.97798,	0.97898,
0.97998,	0.98098,	0.98198,	0.98298,	0.98398,
	0.98498,	0.98599,	0.98699,	0.98799,
0.98899,	0.98999,	0.99099,	0.99199,	0.99299,

```
0.99399,      0.99499,      0.996,      0.997,      0.998,
0.999,           1]), array([[ 0.1563,      0.1563,      0.1
9902, ...,      1,      1,      1],
   [ 0.090909,  0.090909,  0.10375, ...,
  1,      1,      1],
   [ 0.081871,  0.081871,  0.097049, ...,
  1,      1,      1],
   [ 0.13079,   0.13079,   0.15802, ...,
  1,      1,      1]]), 'Confidence', 'Precision'], [ar
ray([
  0,  0.001001,  0.002002,  0.003003,  0.0
04004,  0.005005,  0.006006,  0.007007,  0.008008,
  0.009009,  0.01001,   0.011011,  0.012012,  0.013013,
  0.014014,  0.015015,  0.016016,  0.017017,  0.018018,
  0.019019,  0.02002,   0.021021,  0.022022,  0.023023,
  0.024024,  0.025025,  0.026026,  0.027027,
  0.028028,  0.029029,  0.03003,   0.031031,  0.032032,
  0.033033,  0.034034,  0.035035,  0.036036,  0.037037,
  0.038038,  0.039039,  0.04004,   0.041041,  0.042042,
  0.043043,  0.044044,  0.045045,  0.046046,  0.047047,
  0.048048,  0.049049,  0.05005,   0.051051,
  0.052052,  0.053053,  0.054054,  0.055055,  0.056056,
  0.057057,  0.058058,  0.059059,  0.06006,   0.061061,
  0.062062,  0.063063,  0.064064,  0.065065,  0.066066,
  0.067067,  0.068068,  0.069069,  0.07007,   0.071071,
  0.072072,  0.073073,  0.074074,  0.075075,
  0.076076,  0.077077,  0.078078,  0.079079,  0.08008,
  0.081081,  0.082082,  0.083083,  0.084084,  0.085085,
  0.086086,  0.087087,  0.088088,  0.089089,  0.09009,
  0.091091,  0.092092,  0.093093,  0.094094,  0.095095,
  0.096096,  0.097097,  0.098098,  0.099099,
  0.1001,    0.1011,    0.1021,    0.1031,    0.1041,
```

0.10511,	0.10611,	0.10711,	0.10811,	0.10911,
0.11011,	0.11111,	0.11211,	0.11311,	0.11411,
0.11512,	0.11612,	0.11712,	0.11812,	0.11912,
	0.12012,	0.12112,	0.12212,	0.12312,
0.12412,	0.12513,	0.12613,	0.12713,	0.12813,
0.12913,	0.13013,	0.13113,	0.13213,	0.13313,
0.13413,	0.13514,	0.13614,	0.13714,	0.13814,
0.13914,	0.14014,	0.14114,	0.14214,	0.14314,
	0.14414,	0.14515,	0.14615,	0.14715,
0.14815,	0.14915,	0.15015,	0.15115,	0.15215,
0.15315,	0.15415,	0.15516,	0.15616,	0.15716,
0.15816,	0.15916,	0.16016,	0.16116,	0.16216,
0.16316,	0.16416,	0.16517,	0.16617,	0.16717,
	0.16817,	0.16917,	0.17017,	0.17117,
0.17217,	0.17317,	0.17417,	0.17518,	0.17618,
0.17718,	0.17818,	0.17918,	0.18018,	0.18118,
0.18218,	0.18318,	0.18418,	0.18519,	0.18619,
0.18719,	0.18819,	0.18919,	0.19019,	0.19119,
	0.19219,	0.19319,	0.19419,	0.1952,
0.1962,	0.1972,	0.1982,	0.1992,	0.2002,
0.2012,	0.2022,	0.2032,	0.2042,	0.20521,
0.20621,	0.20721,	0.20821,	0.20921,	0.21021,
0.21121,	0.21221,	0.21321,	0.21421,	0.21522,
	0.21622,	0.21722,	0.21822,	0.21922,
0.22022,	0.22122,	0.22222,	0.22322,	0.22422,
0.22523,	0.22623,	0.22723,	0.22823,	0.22923,
0.23023,	0.23123,	0.23223,	0.23323,	0.23423,
0.23524,	0.23624,	0.23724,	0.23824,	0.23924,
	0.24024,	0.24124,	0.24224,	0.24324,
0.24424,	0.24525,	0.24625,	0.24725,	0.24825,
0.24925,	0.25025,	0.25125,	0.25225,	0.25325,

0.25425,	0.25526,	0.25626,	0.25726,	0.25826,
0.25926,	0.26026,	0.26126,	0.26226,	0.26326,
	0.26426,	0.26527,	0.26627,	0.26727,
0.26827,	0.26927,	0.27027,	0.27127,	0.27227,
0.27327,	0.27427,	0.27528,	0.27628,	0.27728,
0.27828,	0.27928,	0.28028,	0.28128,	0.28228,
0.28328,	0.28428,	0.28529,	0.28629,	0.28729,
	0.28829,	0.28929,	0.29029,	0.29129,
0.29229,	0.29329,	0.29429,	0.2953,	0.2963,
0.2973,	0.2983,	0.2993,	0.3003,	0.3013,
0.3023,	0.3033,	0.3043,	0.30531,	0.30631,
0.30731,	0.30831,	0.30931,	0.31031,	0.31131,
	0.31231,	0.31331,	0.31431,	0.31532,
0.31632,	0.31732,	0.31832,	0.31932,	0.32032,
0.32132,	0.32232,	0.32332,	0.32432,	0.32533,
0.32633,	0.32733,	0.32833,	0.32933,	0.33033,
0.33133,	0.33233,	0.33333,	0.33433,	0.33534,
	0.33634,	0.33734,	0.33834,	0.33934,
0.34034,	0.34134,	0.34234,	0.34334,	0.34434,
0.34535,	0.34635,	0.34735,	0.34835,	0.34935,
0.35035,	0.35135,	0.35235,	0.35335,	0.35435,
0.35536,	0.35636,	0.35736,	0.35836,	0.35936,
	0.36036,	0.36136,	0.36236,	0.36336,
0.36436,	0.36537,	0.36637,	0.36737,	0.36837,
0.36937,	0.37037,	0.37137,	0.37237,	0.37337,
0.37437,	0.37538,	0.37638,	0.37738,	0.37838,
0.37938,	0.38038,	0.38138,	0.38238,	0.38338,
	0.38438,	0.38539,	0.38639,	0.38739,
0.38839,	0.38939,	0.39039,	0.39139,	0.39239,
0.39339.	0.39439.	0.3954.	0.3964.	0.3974.

0.4034, 0.4044, 0.40541, 0.40641, 0.40741,
0.40841, 0.40941, 0.41041, 0.41141,
0.41241, 0.41341, 0.41441, 0.41542, 0.41642,
0.41742, 0.41842, 0.41942, 0.42042, 0.42142,
0.42242, 0.42342, 0.42442, 0.42543, 0.42643,
0.42743, 0.42843, 0.42943, 0.43043, 0.43143
and the model training is resumed. We can check the results in the folder "runs", we can
find some samples and metrics plots generated during the training phase.
0.43243, 0.43343, 0.43443, 0.43544,
0.43644, 0.43744, 0.43844, 0.43944, 0.44044,

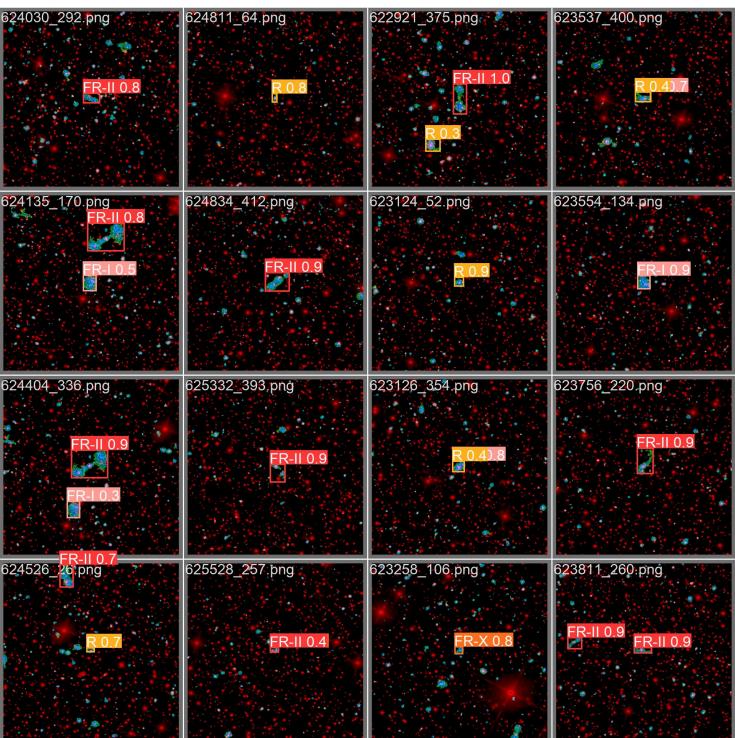
Results

In [42]: `list_files("runs/")`

0.45145, 0.45245, 0.45345, 0.45445, 0.45546,
/ 0.45646, 0.45746, 0.45846, 0.45946,
detect/ 0.46046, 0.46146, 0.46246, 0.46346, 0.46446,
train/ 0.46547, 0.46647, 0.46747, 0.46847, 0.46947,
args.yaml 0.47047, 0.47147, 0.47247, 0.47347, 0.47447,
events 0.47548, 0.47648, 0.47748, 0.47848, 0.47948,
labels 0.48048, 0.48148, 0.48248, 0.48348,
labels 0.48448, 0.48549, 0.48649, 0.48749, 0.48849,
train_batch0.jpg 0.48949, 0.49049, 0.49149, 0.49249, 0.49349,
train_batch1.jpg 0.49449, 0.49549, 0.49649, 0.49749, 0.49849,
train_batch2.jpg 0.4995, 0.5005, 0.5015, 0.5025, 0.5035,
results 0.5045, 0.50551, 0.50651, 0.50751,
train_batch4920.jpg 0.50851, 0.50951, 0.51051, 0.51151, 0.51251,
train_batch4921.jpg 0.51351, 0.51451, 0.51552, 0.51652, 0.51752,
train_batch4922.jpg 0.51852, 0.51952, 0.52052, 0.52152, 0.52252,
val_batch0_labels.jpg 0.52352, 0.52452, 0.52553, 0.52653, 0.52753,
val_batch0_pred.jpg 0.52853, 0.52953, 0.53053, 0.53153,
val_batch1_labels.jpg 0.53253, 0.53353, 0.53453, 0.53554, 0.53654,
val_batch1_pred.jpg 0.53754, 0.53854, 0.53954, 0.54054, 0.54154,
val_batch2_labels.jpg 0.54254, 0.54354, 0.54454, 0.54555, 0.54655,
val_batch2_pred.jpg 0.54755, 0.54855, 0.54955, 0.55055, 0.55155,
PR_curve.png

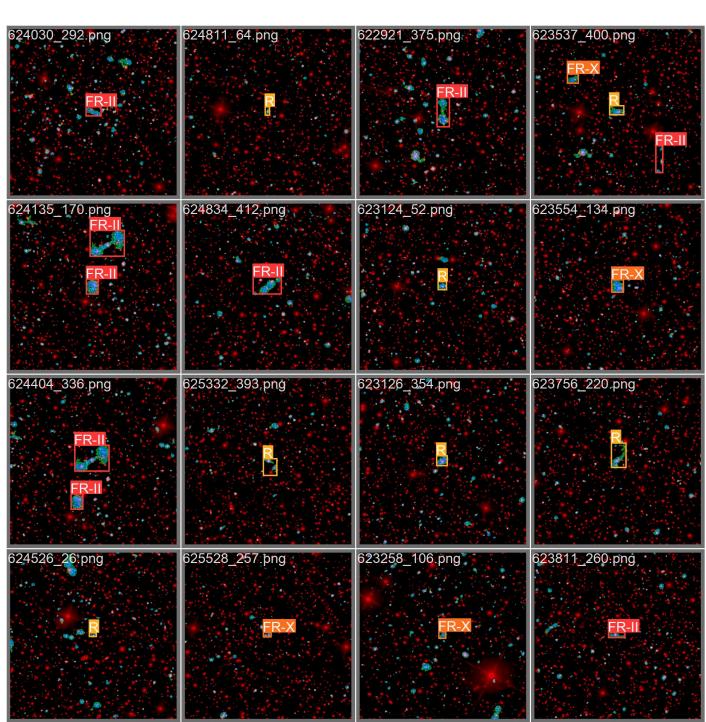
F1_0u5525þng	0.55355,	0.55455,	0.55556,
0.55656,P_cur0e5þü6,	0.55856,	0.55956,	0.56056,
0.56156,R_cur0e5þü6,	0.56356,	0.56456,	0.56557,
0.56657,confu0i567þüatrix_n0rþü85þed.png	0.56957,	0.57057,	
0.57157,confu0i567þüatrix.p0g57357,	0.57457,	0.57558,	
res015þ6þ8g	0.57758,	0.57858,	0.57958,
0.58058,weigh0sþ8158,	0.58258,	0.58358,	0.58458,
0.58559,l0sþ86þ9,	0.58759,	0.58859,	0.58959,
0.59059,b0sþ9þþ9,	0.59259,	0.59359,	0.59459,
0.5956,0.5966,	0.5976,	0.5986,	0.5996,
	0.6006,	0.6016,	0.6026,
0.6046,0.60561,	0.60661,	0.60761,	0.60861,
0.60961,0.61061,	0.61161,	0.61261,	0.61361,
0.61461,0.61562,	0.61662,	0.61762,	0.61862,
0.61962,0.62062,	0.62162,	0.62262,	0.62362,
	0.62462,	0.62563,	0.62663,
0.62863,0.62963,	0.63063,	0.63163,	0.63263,
0.63363,0.63463,	0.63564,	0.63664,	0.63764,
0.63864,0.63964,	0.64064,	0.64164,	0.64264,
0.64364,0.64464,	0.64565,	0.64665,	0.64765,
	0.64865,	0.64965,	0.65065,
0.65265,0.65365,	0.65465,	0.65566,	0.65666,
0.65766,0.65866,	0.65966,	0.66066,	0.66166,
0.66266,0.66366,	0.66466,	0.66567,	0.66667,
0.66767,0.66867,	0.66967,	0.67067,	0.67167,
	0.67267,	0.67367,	0.67467,
0.67668,0.67768,	0.67868,	0.67968,	0.68068,
0.68168,0.68268,	0.68368,	0.68468,	0.68569,
0.68669,0.68769,	0.68869,	0.68969,	0.69069,
0.69169,0.69269,	0.69369,	0.69469,	0.6957,
	0.6967,	0.6977,	0.6987,
			0.6997,

0.7007,	0.7017,	0.7027,	0.7037,	0.7047,
0.70571,	0.70671,	0.70771,	0.70871,	0.70971,
0.71071,	0.71171,	0.71271,	0.71371,	0.71471,
0.71572,	0.71672,	0.71772,	0.71872,	0.71972,
	0.72072,	0.72172,	0.72272,	0.72372,
0 72472	0 72573	0 72673	0 72773	0 72873



Predictions

0.82082,	0.82182,	0.82282,	0.82382,	0.82482,
0.82583,	0.82683,	0.82783,	0.82883,	0.82983,
0.83083,	0.83183,	0.83283,	0.83383,	0.83483,
0.83584,	0.83684,	0.83784,	0.83884,	0.83984,
	0.84084,	0.84184,	0.84284,	0.84384,
0.84484,	0.84585,	0.84685,	0.84785,	0.84885,



Labels

0.82382,	0.82482,
0.82883,	0.82983,
0.83383,	0.83483,
0.83884,	0.83984,
0.84284,	0.84384,
0.84785,	0.84885,

0.84985, 0.85085, 0.85185, 0.85285, 0.85385,
0.85485, 0.85586, 0.85686, 0.85786, 0.85886,
0.85986, 0.86086, 0.86186, 0.86286, 0.86386,
0.86486, 0.86586, 0.86686, 0.86786, 0.86886,
0.86887, 0.86987, 0.87087, 0.87187, 0.87287,
0.87387, 0.87487, 0.87588, 0.87688, 0.87788
how about set some zones of interest and detect how many objects are in such zones?
0.87888, 0.87988, 0.88088, 0.88188, 0.88288,

In [43]: `#!pip install supervision==0.2.0`

```
Collecting supervision==0.2.0
  Downloading supervision-0.2.0-py3-none-any.whl.metadata (6.0
KB)
Requirement already satisfied: numpy in /opt/conda/lib/python3.
10/site-packages (from supervision==0.2.0) (1.26.4)
Requirement already satisfied: opencv-python in /opt/conda/lib/
python3.10/site-packages (from supervision==0.2.0) (4.5.3)
Requirement already satisfied: matplotlib in /opt/conda/lib/pyt
hon3.10/site-packages (from supervision==0.2.0) (3.8.3)
Requirement already satisfied: contourpy>1.0.1 in /opt/conda/l
ib/python3.10/site-packages (from supervision==0.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/p
ython3.10/site-packages (from matplotlib>supervision==0.2.
0,11.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/p
ython3.10/site-packages (from matplotlib>supervision==0.2.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/
lib/python3.10/site-packages (from matplotlib>supervision==0.
2.0,11.49.0)
Requirement already satisfied: kiwisolver>1.3.1 in /opt/conda/
lib/python3.10/site-packages (from matplotlib>supervision==0.
2.0,11.4.5)
Requirement already satisfied: packaging>=20.0 in /opt/conda/li
```

0/099hon3.10/site-packages (from matplotlib->superVision==0.0.9
020123.2), 0, 0, 0],
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.10/site-packages (from matplotlib->superVision==0.2.0) (9.5.0) [0.97674, 0.97674, 0.97674, ...]
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib->superVision==0.2.0) (3.1.2) [0, 0]], 'Confidence', 'Recall'])
Requestsemé0t6al4520986565485ed: python-dateutil>=2.7 in /opt/conda/lib/python3.10/site-packages ('metrcs/mAP50(B)'>superVision 050(B)' (2:Merics/mAP50-95(B))
Requirement already satisfied: os56058,5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.7->matplotlib->superVision 0.2.0:FR-I', 2: 'FR-X', 3: 'R')
Installing superVision-0.2.0-py3-none-any.whl (19 kB)
Testé à l'identique dans la session 006080336800813492, 'metrics/mAP50(B)': 0.65525
56672829998, 'metrics/mAP50-95(B)': 0.6132538132536094, 'fitness': 0.6174539986565485}
save_dir: PosixPath('runs/detect/train')
speed: {'preprocess': 1.1317849159240723, 'inference': 1.986695
471264067, 'loss': 0.00043653306506928944, 'postprocess': 0.677
5259971618652}
task: 'detect'

In [44]: `import supervision as sv`

lets use a new sample image

```
In [60]: sample_path = Path('datasets/galaxyradionet/data/RadioGalaxyNET/val/624
```

```
In [61]: image_sample = Image.open(sample_path)
```

```
In [62]: results = model(image_sample, imgsz=640)[0]
```

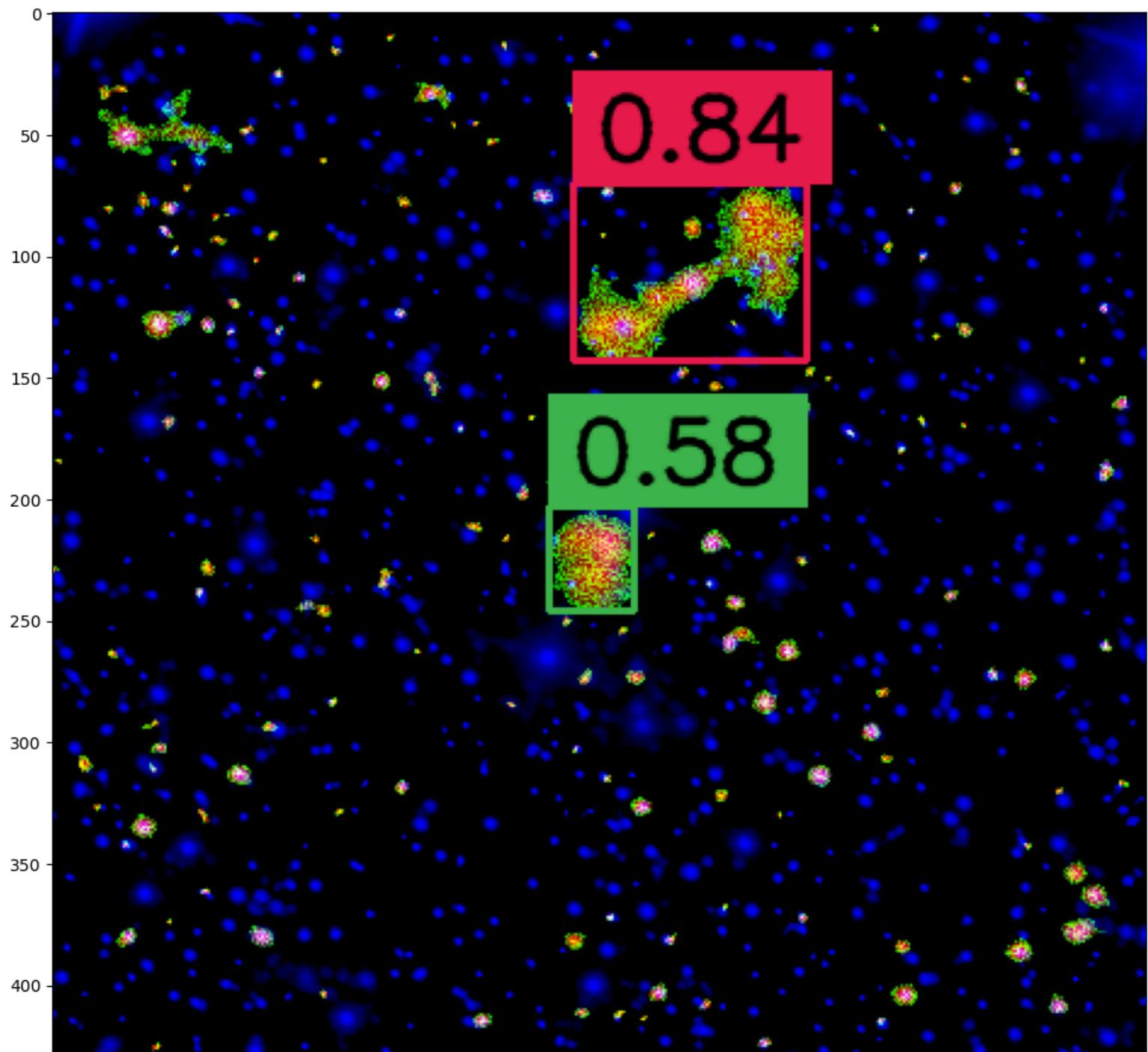
```
0: 640x640 1 FR-II, 1 FR-I, 9.1ms
Speed: 3.4ms preprocess, 9.1ms inference, 1.6ms postprocess per
image at shape (1, 3, 640, 640)
```

```
In [ ]:
```

```
In [63]: detections = sv.Detections.from_yolov8(results)
```

```
In [64]: # annotate
box_annotator = sv.BoxAnnotator(thickness=2, text_thickness=2, text_sca
image_sample_box = box_annotator.annotate(scene=np.array(image_sample),
```

```
In [65]: %matplotlib inline  
sv.show_frame_in_notebook(image_sample_box, (12, 12))
```



```
In [66]: # initiate polygon zone  
# x, y pos vertices  
polygon = np.array([  
    [150, 100],  
    [300, 100],  
    [300, 150],  
    [150, 150]  
])
```

```
In [67]: image_sample.size
```

```
Out[67]: (450, 450)
```

```
In [68]: zone = sv.PolygonZone(polygon=polygon, frame_resolution_wh=(450, 450))
```

```
In [69]: # initiate annotators  
box_annotator = sv.BoxAnnotator(thickness=2, text_thickness=2, text_sca  
zone_annotator = sv.PolygonZoneAnnotator(zone=zone, color=sv.Color.whit
```

```
In [70]: # detect  
results = model(image_sample, imgsz=640)[0]  
detections = sv.Detections.from_yolov8(results)  
#detections = detections[detections.class_id == 0]  
zone.trigger(detections=detections)
```

0: 640x640 1 FR-II, 1 FR-I, 8.5ms
Speed: 3.1ms preprocess, 8.5ms inference, 1.6ms postprocess per
image at shape (1, 3, 640, 640)

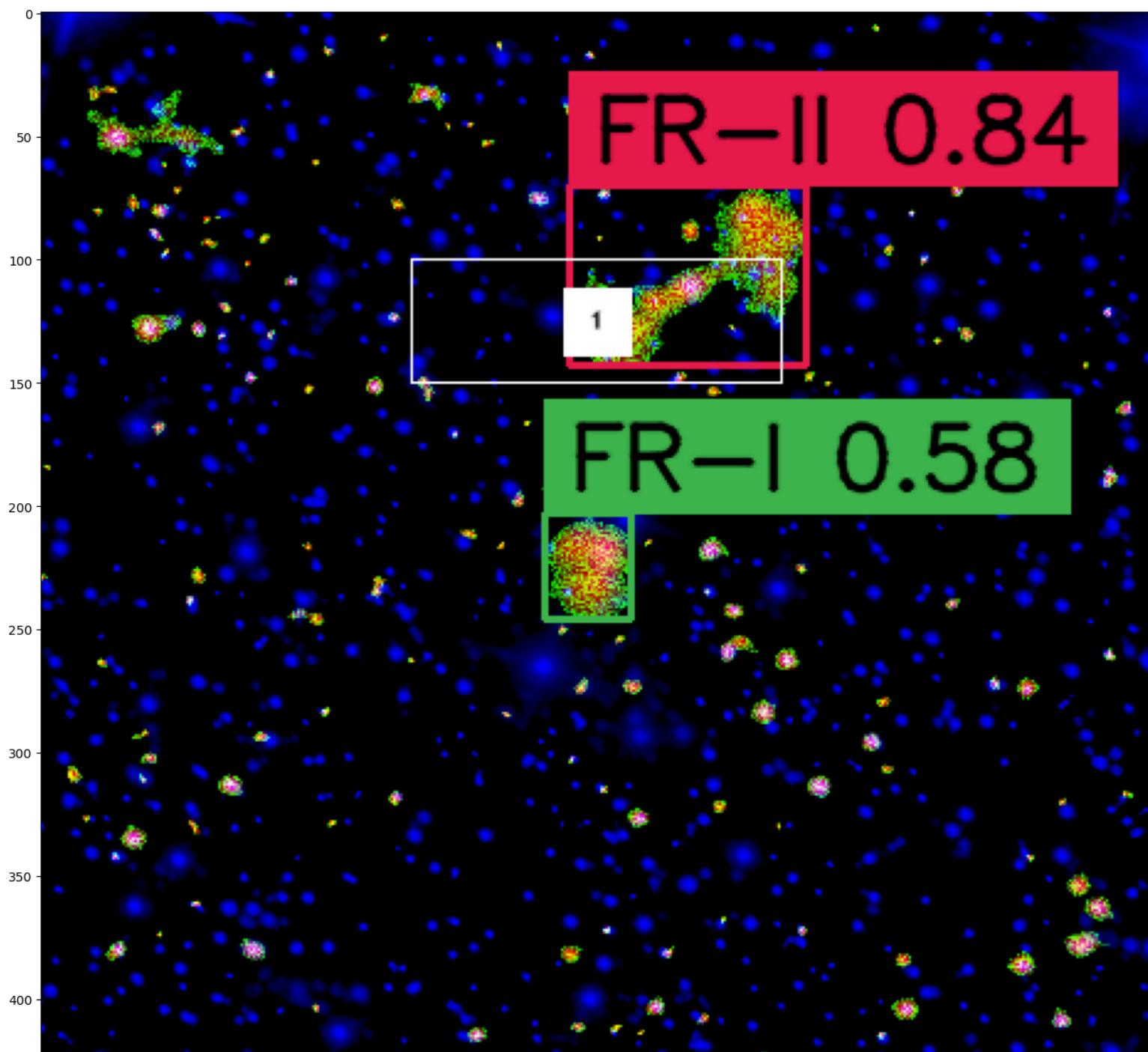
```
Out[70]: array([ True, False])
```

```
In [104]: detections.class_id == 0
```

```
Out[104]: array([ True])
```

```
In [71]: # annotate
box_annotator = sv.BoxAnnotator(thickness=2, text_thickness=2, text_sca
labels = [f"{model.names[class_id]} {confidence:0.2f}" for _, confidence in
image_sample_box = box_annotator.annotate(scene=np.array(image_sample),
image_sample_box = zone_annotator.annotate(scene=image_sample_box)
```

```
In [72]: sv.show_frame_in_notebook(image_sample_box, (16, 16))
```



```
In [73]: def calculate_bounding_boxes(image_shape, num_rows, num_cols):
    """
        Calculate bounding box coordinates for sectors in an image.

In [92]: Parameters:
Out[92]:     image_shape (tuple): Shape of the input image (height, width).
In [107]:     num_rows (int): Number of rows for sectors.
Out[107]:     num_cols (int): Number of columns for sectors.

Returns:
    list of numpy.ndarray: List containing bounding box coordinates
"""

height, width = image_shape
sector_height = height // num_rows
sector_width = width // num_cols

bounding_boxes = []

for row in range(num_rows):
    for col in range(num_cols):
        x1 = col * sector_width
        y1 = row * sector_height
        x2 = x1 + sector_width
        y2 = y1 + sector_height
        bounding_boxes.append(np.array([[x1, y1], [x2, y1], [x2, y2], [x1, y2]]))

return bounding_boxes
```

```
In [92]: image_shape = (450, 450)
        num_rows = 2
        num_cols = 2
        bounding_boxes = calculate_bounding_boxes(image_shape, num_rows, num_cols)
```

```
In [93]: len(bounding_boxes)
```

```
Out[93]: 4
```

```
In [94]: colors = sv.ColorPalette.default()
        polygons = calculate_bounding_boxes(image_shape, num_rows, num_cols)
```

```
In [95]: zones = [
    sv.PolygonZone(
        polygon=polygon,
        frame_resolution_wh=(450, 450)
    )
    for polygon
    in polygons
]
```

```
In [96]: len(zones)
```

```
Out[96]: 4
```

```
In [97]: zone_annotators = [
    sv.PolygonZoneAnnotator(
        zone=zone,
        color=colors.by_idx(index),
        thickness=1,
        text_thickness=2,
        text_scale=.3
    )
    for index, zone
    in enumerate(zones)
]
```

```
In [98]: box_annotators = [
    sv.BoxAnnotator(
        color=colors.by_idx(index),
        thickness=1,
        text_thickness=1,
        text_scale=.2
    )
    for index
    in range(len(polygons))
]
```

```
In [99]: # detect
```

```
results = model(image_sample, imgsz=640)[0]
detections = sv.Detections.from_yolov8(results)
```

```
0: 640x640 1 FR-II, 1 FR-I, 7.5ms
```

```
Speed: 2.6ms preprocess, 7.5ms inference, 1.2ms postprocess per
image at shape (1, 3, 640, 640)
```

```
In [103]: mask
```

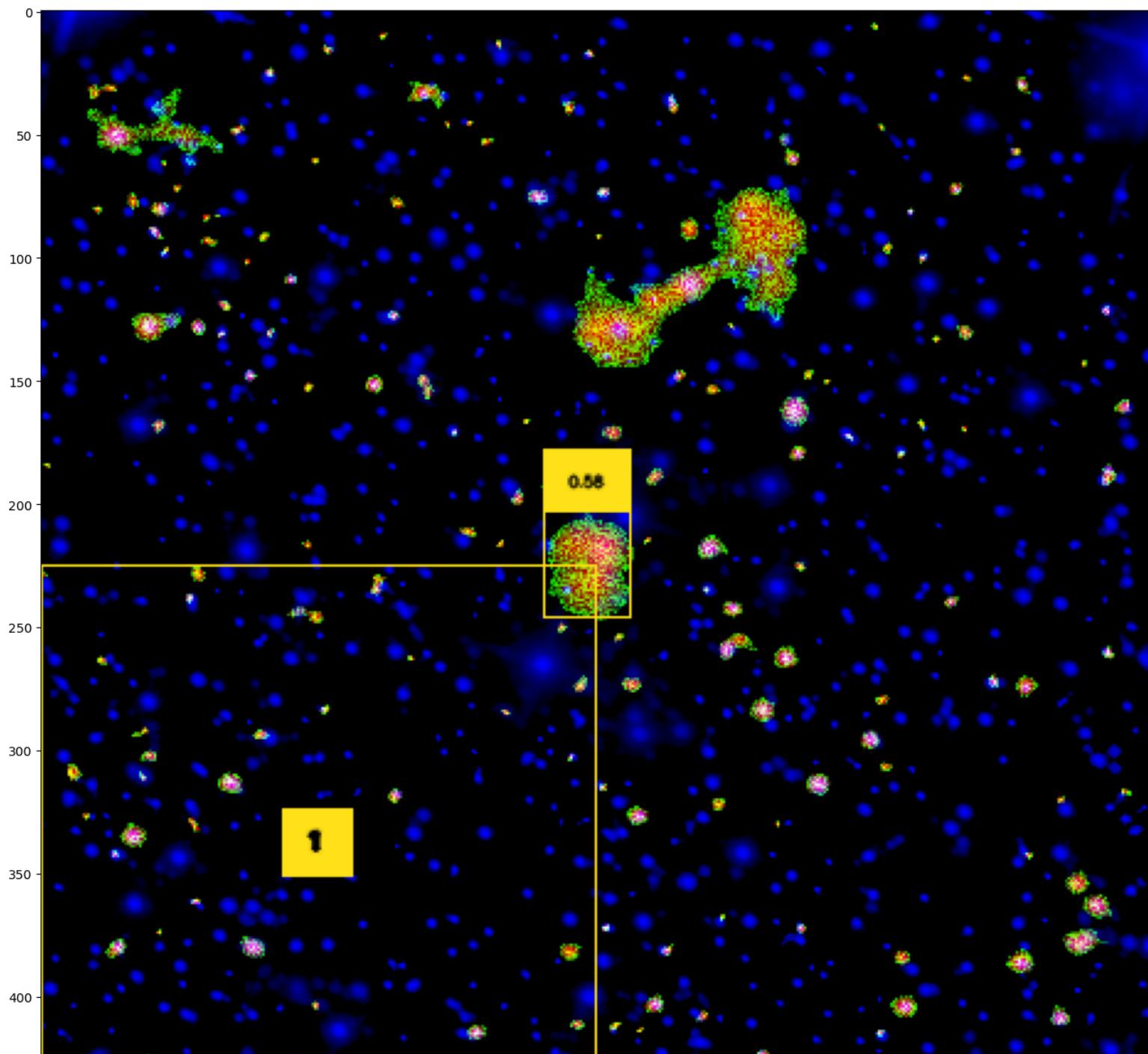
```
Out[103]: array([False, False])
```

```
In [81]: np.bool = np.bool_
```

In [104]:

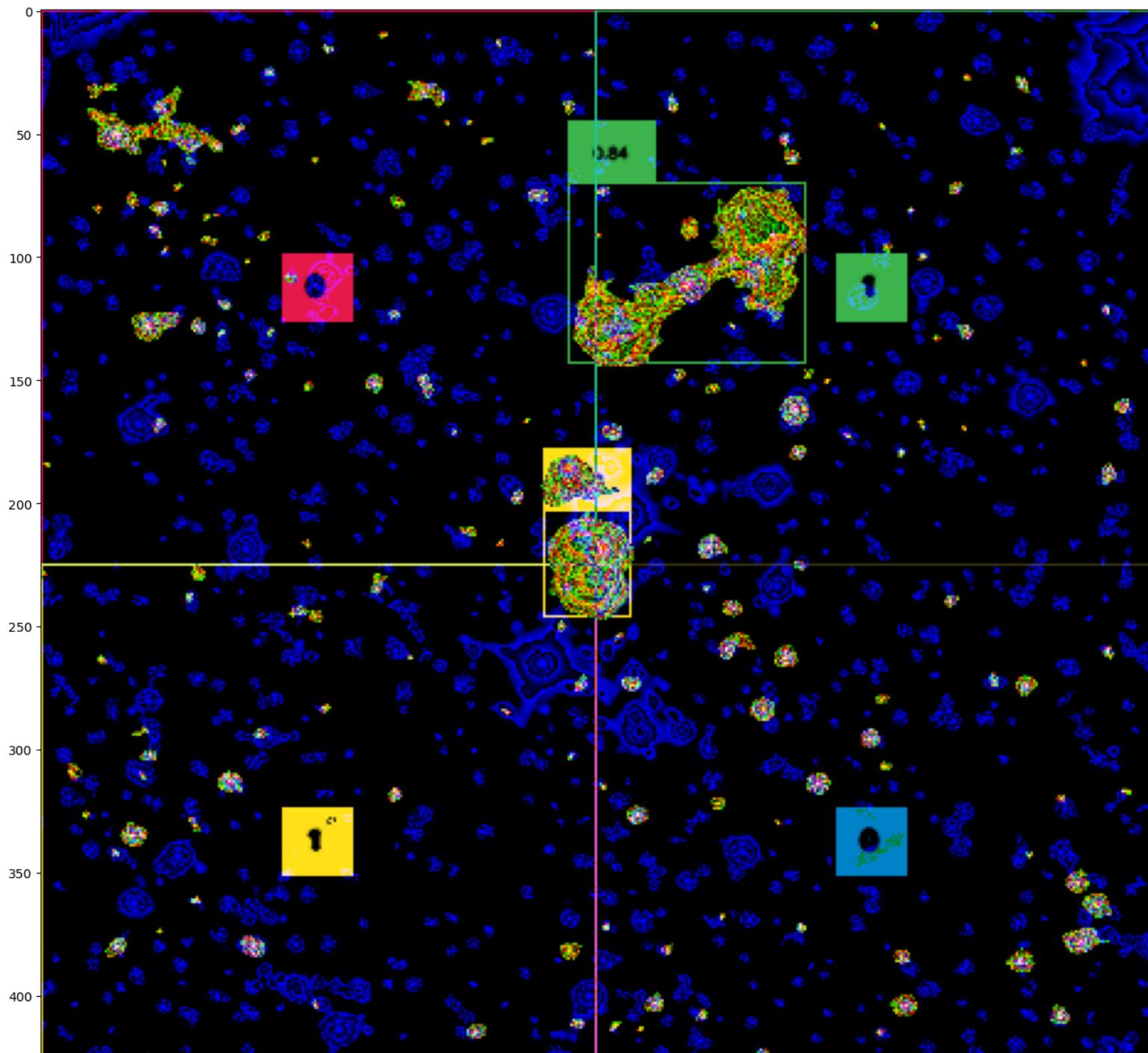
```
quadrants = []
for zone, zone_annotator, box_annotator in zip(zones, zone_annotators,
mask = zone.trigger(detections=detections)
detections_filtered = detections[mask]
image_sample_box = box_annotator.annotate(scene=np.array(image_samp
image_sample_box = zone_annotator.annotate(scene=image_sample_box)
quadrants.append(image_sample_box)
```

```
In [108]: sv.show_frame_in_notebook(quadrants[2], (16, 16))
```



```
In [132]: sv.show_frame_in_notebook(np.clip(sum(quadrants), 0, 255).astype(np.uint8))
```

```
In [ ]:
```



Industrial applications

Inspection & monitoring

- Inspections and monitoring are essential to ensure the safe operation of energy assets



Solar

- PV panels – cracking and corrosion



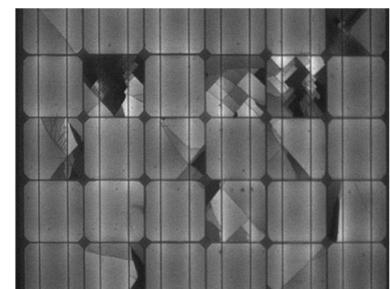
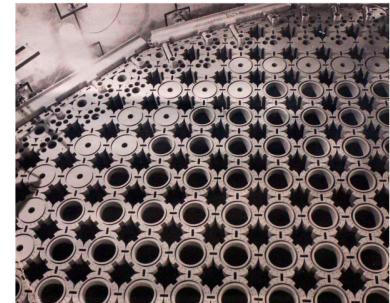
Nuclear

- Graphite – cracking and weight loss

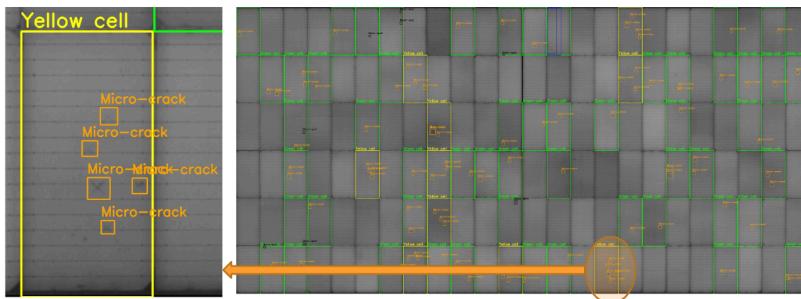


Wind

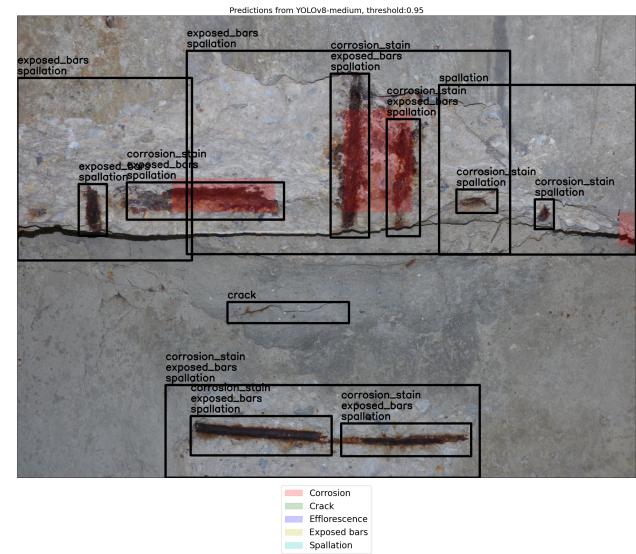
- Wind turbines – cracks, erosion, lightning, etc.



Industrial application



Zakaria Ghanim, Benoit Braisaz, Gustavo Medina Vazquez,
Nader Somrani, Anushiya Thevapalan



Felipe Ferreira de Freitas, Gustavo Medina
Vazquez, Nader Somrani

