

Document index

Benchmark API	3
Video intelligence API (Google)	3
Features	3
Prices	3
Rekognition (Amazon)	4
Features	4
Prices	4
TensorFlow Object detection API	5
Features	5
Natural language processing	5
Images	5
Audio	5
Video	5
Prices	6
Ximilar	6
Features	6
Prices	6
Valossa	6
Features	6
Prices	7
Watson Visual Recognition (IBM)	7
Image AI	7
Features	7
Prices	7
TorchVision	7
Features	7
Prices	8
Sensifai	8
Features	8
Prices	8
Conclusion	8
SOLUTION PROTOTYPING	9
Project's repository	9
TENSORFLOW: OBJECT DETECTION API	9
WORKFLOW STEPS	9
TESTING PROBLEMS	11
PERFORMANCE	11
YOLOv5	12
WORKFLOW STEPS	12
PERFORMANCE	12

SHALLOW LEARNING	15
Data collection	15
Feature engineering	15
ML Algorithm	15
Split in segments	15
CONCLUSION	16
TEXT RECOGNITION	17
TESSERACT OCR	17
WORKFLOW STEPS	17
IMPLEMENTATION	18
main.py	18
class Record	20
class FrameWithFaces	21
class Segment	21
Video processing	22
calculateFeatureAndTrain_module.py	27
svm_module.py	31
xgboost_module.py	33
testTFmodel.py	35
testYOLOv5-lavagne.py	36
HOW TO USE	37
OUTPUT FORMAT	38
Terminal output	38
JSON data structure format	39

Benchmark API

Video intelligence API (Google)

<https://cloud.google.com/video-intelligence>

Features

- Livestream mode
- Annotate videos saved locally or on cloud
- Explicit content detection
- Face detection
- Label detection
- Logo recognition
- Object tracking
- Person detection
- Shot change detection
- **Speech transcription**
- Text detection
- Costo: Free per 16,5 ore di video al mese se non vado oltre il limite

Prices

Prices and functions per offline videos

Funzionalità	Primi 1000 minuti	Oltre i 1000 minuti
Rilevamento etichette	Nessun costo	\$ 0,10 al minuto
Rilevamento inquadrature	Nessun costo	\$ 0,05 al minuto o nessun costo con rilevamento etichette
Rilevamento di contenuti esplicativi	Nessun costo	\$ 0,10 al minuto
Trascrizione del parlato	Nessun costo	\$ 0,048 al minuto (addebiti solo per trascrizioni en-US)
Monitoraggio oggetti	Nessun costo	\$ 0,15 al minuto
Rilevamento testo	Nessun costo	\$ 0,15 al minuto
Rilevamento loghi	Nessun costo	\$ 0,15 al minuto
Rilevamento facciale	Nessun costo	\$ 0,10 al minuto
Rilevamento persone	Nessun costo	\$ 0,10 al minuto
Riconoscimento di volti celebri	Nessun costo	\$ 0,10 al minuto

Prices and functions per live videos

Funzionalità	Primi 1000 minuti	Oltre i 1000 minuti
Rilevamento etichette	Nessun costo	\$ 0,12 al minuto
Rilevamento inquadrature	Nessun costo	\$ 0,07 al minuto
Rilevamento di contenuti espliciti	Nessun costo	\$ 0,12 al minuto
Monitoraggio oggetti	Nessun costo	\$ 0,17 al minuto

Rekognition (Amazon)

<https://aws.amazon.com/it/rekognition>

Features

- Scene objects and activities detection
- Explicit content detection
- Text detection
- Celebrity face detection
- Face analysis and detection
- Face recognition
- People movement detection and livestream video analysis.

CONs: no speech-recognition (?)

Prices

12 months free with 1000 minutes per month (16,5 hours)

Tabella dei prezzi

Analisi di video archiviate

Regione: Europa (Francoforte) ▾

Caratteristica	Prezzi
Rilevamento delle etichette	0,12 USD/min
Moderazione dei contenuti	0,12 USD/min
Rilevamento del testo	0,12 USD/min
Rilevamento facciale	0,12 USD/min
Riconoscimento di volti celebri	0,12 USD/min
Ricerca facciale	0,12 USD/min
Rilevamento dei movimenti delle persone	0,12 USD/min

Analisi multimediale

Caratteristica	Prezzi
Rilevamento dei fotogrammi	0,06 USD/min
Rilevamento di fotogrammi neri, titoli di coda, barre colore ("segnali d'azione tecnici")	0,06 USD/min

Analisi di video in live stream

Caratteristica	Prezzi
Ricerca facciale	0,144 USD/min

I costi di utilizzo applicati sono mensili e ripartiti proporzionalmente nei minuti di utilizzo parziale.

Storage di metadati facciali 0.00001 USD/metadati facciali al mese**

**I costi di storage applicati sono mensili e ripartiti proporzionalmente nei mesi di utilizzo parziale

TensorFlow Object detection API

https://github.com/tensorflow/models/tree/master/research/object_detection
<https://www.youtube.com/watch?v=yqkISICHH-U>

Features

Natural language processing

- Text classification
- Real time semantic search
- Multilingual question answering

Images

- Image classification
- Object recognition
- Human faces generation

Audio

- Pitch recognition
- Sound classification

Video

- Action recognition
- Video interpolation
- Text-to-video retrieval

CONs: ma solo Object detection senza Speech recognition / transcript.

Prices

Free (open source)

Ximilar

<https://www.ximilar.com/>

Features

- Image recognition
- Photo similarity
- Product similarity

Prices

Free	Business	Professional
€0 \$0 + €0.9 or \$0.95 per 1k credits	€55 \$59 or €149 \$175 monthly 100k or 300k API credits*	€450 \$499 monthly 1M API credits*
3,000 API credits free*	Fast parallel processing	Fast parallel processing
Sequential processing	Standard support + one consultation with ML expert	Priority support + regular consultations with ML expert
Limited email support	Free services plus: — Advanced Annotation System — Object Detection — Fashion Tagging <i>Contact us to test other services.</i>	Business services plus: — Team Collaboration — Visual Fashion Search
Available services: — Image Recognition — Generic Tagging — Photos Similarity — Product Similarity <i>Contact us to test other services.</i>	Image Recognition: 3 tasks	Image Recognition: 20 tasks, unlim. flows — ML loop: add processed images to training data
Image Recognition: 3 tasks	Image Recognition: 10 tasks	Object Detection: 20 different objects
Recognition Flows: 1 flow	Recognition Flows: 3 flows	Similarity: 3 collections, unlim. images — nightly sync of your DB — custom similarity model
Similarity: 1 collection, 3,000 images	Object Detection: 3 different objects	Fashion Tagging: map our taxonomy to yours
TRY IT NOW Quick setup No credit card required	UPGRADE IN APP Easy & quick setup Credit card or PayPal	UPGRADE IN APP Easy & quick setup

Valossa

<https://valossa.com/>

Features

- Video recognition
- Face recognition
- Emotion & sentiment detection
- Object detection

- **Audio sound tags**
- Explicit content detection
- Speech analysis
- Video categories and topics

Prices

Price to be agreed with the staff.

Watson Visual Recognition (IBM)

<https://cloud.ibm.com/docs/visual-recognition?topic=visual-recognition-index>

It will shut down on 31st December 2021, from 7th January 2021 it is no longer available.

Image AI

<http://imageai.org/>

Features

- Object detection
- Video Tracking
- Video analysis and tags
- Livestream mode
- Detection speed

Prices

Free (open source)

TorchVision

<https://pytorch.org/vision/0.9/io.html#video>

Features

- Image classification
- Object detection
- Instance segmentation
- Keypoint detection (simplified human skeleton)
- **Video classification**

- Semantic segmentation

Prices

Free (open source)

Sensifai

<https://sensifai.com/#services>

<https://demo.sensifai.com/>

Features

- Video recognition
- Video segmentation
- Explicit content detection
- Celebrity detection
- Object, scene, attribute recognition
- Action, sport recognition
- Logo recognition
- Landmark recognition
- Audio, sound, speech recognition

Prices

Prices to be agreed with the staff.

Conclusion

The ideal choices would be Valossa or Sensifai, but since they need to be agreed on the price, a good trade-off could be Google's Video Intelligence API.

If our goal is just video recognition, then we could use some open-source frameworks. But if we also need speech and audio recognition, then Google's Video Intelligence API is the safe bet.

SOLUTION PROTOTYPING

Our problem was to detect “BLACKBOARD” or in general to detect the scene whether it's BLACKBOARD, SLIDE, SLIDE-AND-TALK or TALK. We've tried to develop a machine learning model by prototyping the following three methods.

Project's repository

<https://github.com/Gabbosaur/SWT-AggregateLD>

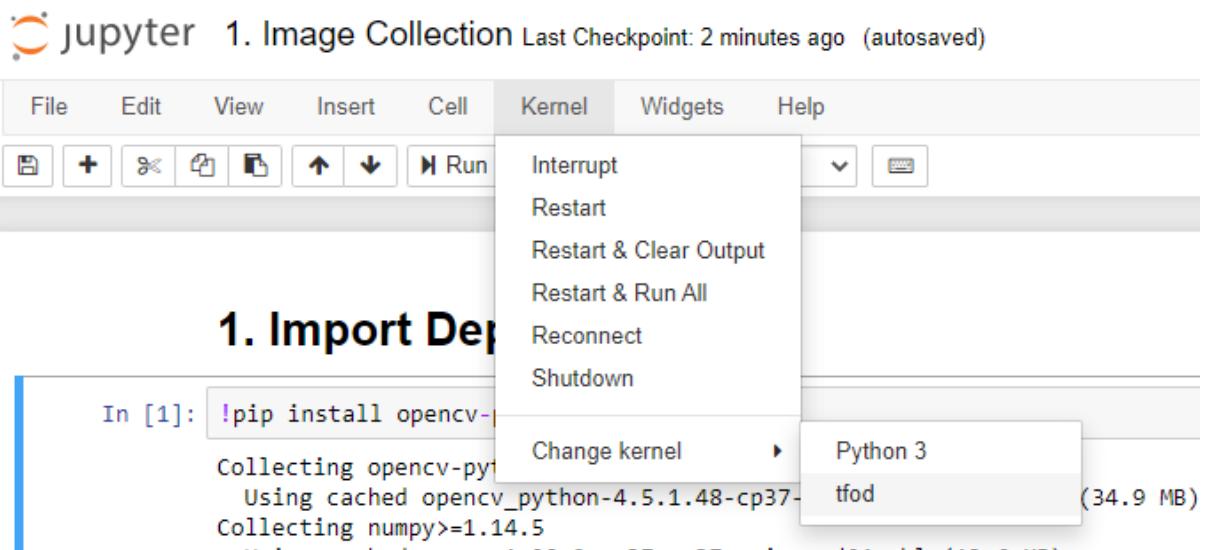
TENSORFLOW: OBJECT DETECTION API

We chose this framework because the stakeholder preferred open-source APIs and it contains the features we needed.

The following steps are aimed at obtaining **custom detections**. For instance, gesture recognition.

WORKFLOW STEPS

1. Clone this repository: <https://github.com/nicknochnack/TFODCourse>
2. Create a new virtual environment using `python -m venv tfod`
3. Activate your virtual environment
 - o `source tfod/bin/activate # Linux`
 - o `.\tfod\Scripts\activate # Windows`
4. Install dependencies and add virtual environment to the Python Kernel
 - o `python -m pip install --upgrade pip`
 - o `pip install ipykernel`
 - o `python -m ipykernel install --user --name=tfodj`
5. Collect images using the Notebook [1. Image Collection.ipynb](#) - ensure you change the kernel to the virtual environment as shown below



6. Manually divide collected images into two folders: train and test. So now all folders and annotations should be split between the following two folders.
 - o \TFODCourse\Tensorflow\workspace\images\train
 - o \TFODCourse\Tensorflow\workspace\images\test
7. Begin the training process by opening [2. Training and Detection.ipynb](#), this notebook will walk you through installing Tensorflow Object Detection, making detections, saving and exporting your model.
8. During this process the Notebook will install Tensorflow Object Detection. You should ideally receive a notification indicating that the API has installed successfully at Step 8 with the last line stating OK.

```

copying object_detection/protos/hyperparams_pb2.py -> build/lib/object_detection/protos
In [16]: VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}

[ RUN   ] ModelBuilderTF2Test.test_session
[ SKIPPED ] ModelBuilderTF2Test.test_session
[ RUN   ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0403 11:13:39.484760 4864 test_util.py:2096] time(_main_.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN   ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I0403 11:13:39.485760 4864 test_util.py:2096] time(_main_.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN   ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I0403 11:13:39.485760 4864 test_util.py:2096] time(_main_.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
-----
Ran 21 tests in 15.178s
OK (skipped=1)

```

If not, resolve installation errors by referring to the [Error Guide.md](#) in this folder.

9. Once you get to step 6. Train the model, inside of the notebook, you may choose to train the model from within the notebook. I have noticed however that training inside of a separate terminal on a Windows machine you're able to display live loss metrics.

```

WARNING:tensorflow:From D:\YouTube\OD\TFODCourse\tfod\lib\site-packages\tensorflow\python\util\deprecation.py:605: calling map_fn_v2 (from tensorflow.python.ops.map_fn) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Use fn_output_signature instead
W0403 11:31:26.529881 9112 deprecation.py:537] From D:\YouTube\OD\TFODCourse\tfod\lib\site-packages\tensorflow\python\util\deprecation.py:605: calling map_fn_v2 (from tensorflow.python.ops.map_fn) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Use fn_output_signature instead
INFO:tensorflow:Step 100 per-step time 0.141s loss=0.833
I0403 11:31:50.217591 1468 model.lib.v2.py:682] Step 100 per-step time 0.141s loss=0.833
INFO:tensorflow:Step 200 per-step time 0.144s loss=0.537
I0403 11:32:04.571963 1468 model.lib.v2.py:682] Step 200 per-step time 0.144s loss=0.537
INFO:tensorflow:Step 300 per-step time 0.115s loss=0.433
I0403 11:32:18.239534 1468 model.lib.v2.py:682] Step 300 per-step time 0.115s loss=0.433
INFO:tensorflow:Step 400 per-step time 0.116s loss=0.470
I0403 11:32:31.978699 1468 model.lib.v2.py:682] Step 400 per-step time 0.116s loss=0.470
INFO:tensorflow:Step 500 per-step time 0.136s loss=0.365
I0403 11:32:45.594426 1468 model.lib.v2.py:682] Step 500 per-step time 0.136s loss=0.365
INFO:tensorflow:Step 600 per-step time 0.171s loss=0.329
I0403 11:32:59.319796 1468 model.lib.v2.py:682] Step 600 per-step time 0.171s loss=0.329
INFO:tensorflow:Step 700 per-step time 0.115s loss=0.389
I0403 11:33:12.973443 1468 model.lib.v2.py:682] Step 700 per-step time 0.115s loss=0.389
INFO:tensorflow:Step 800 per-step time 0.111s loss=0.304
I0403 11:33:26.265393 1468 model.lib.v2.py:682] Step 800 per-step time 0.111s loss=0.304

```

(screenshot during training phase)

10. You can optionally evaluate your model inside of Tensorboard. Once the model has been trained and you have run the evaluation command under Step 7. Navigate to the evaluation folder for your trained model e.g.

- `cd Tensorlfow/workspace/models/my_ssd_mobnet/eval`

and open Tensorboard with the following command

```
tensorboard --logdir=.
```

Tensorboard will be accessible through your browser and you will be able to see metrics including mAP - mean Average Precision, and Recall.

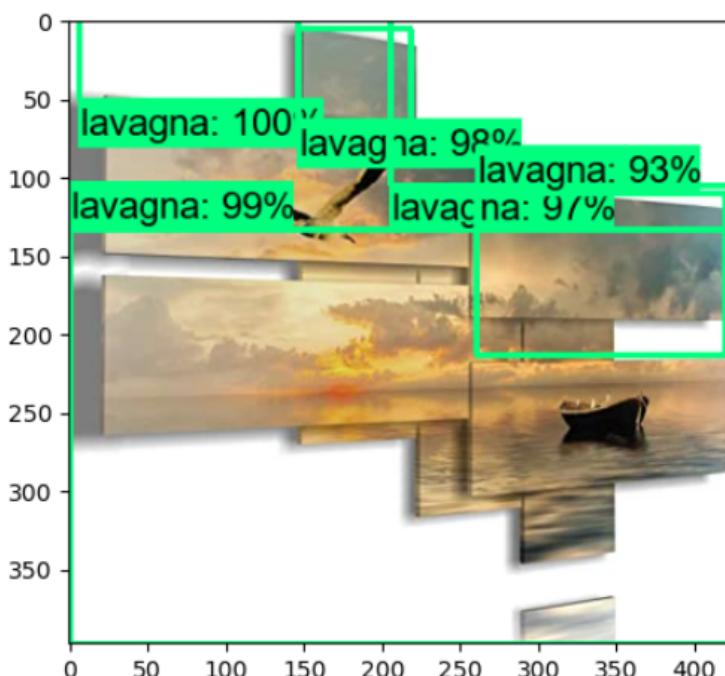
TESTING PROBLEMS

We tested with a few images taken with a webcam and using the model [SSD MobileNet V2 FPNLite 320x320](#), the performance was okay.

After that, we tried to use other images taken from the web (with different backgrounds and hand positions), but the performance got worse.

PERFORMANCE

Testing performance of TF model custom trained MobileNet, which as you can see performed poorly.



YOLOv5

<https://github.com/ultralytics/yolov5>

Pre-trained model used to detect the most common objects in the real world.

WORKFLOW STEPS

PERFORMANCE

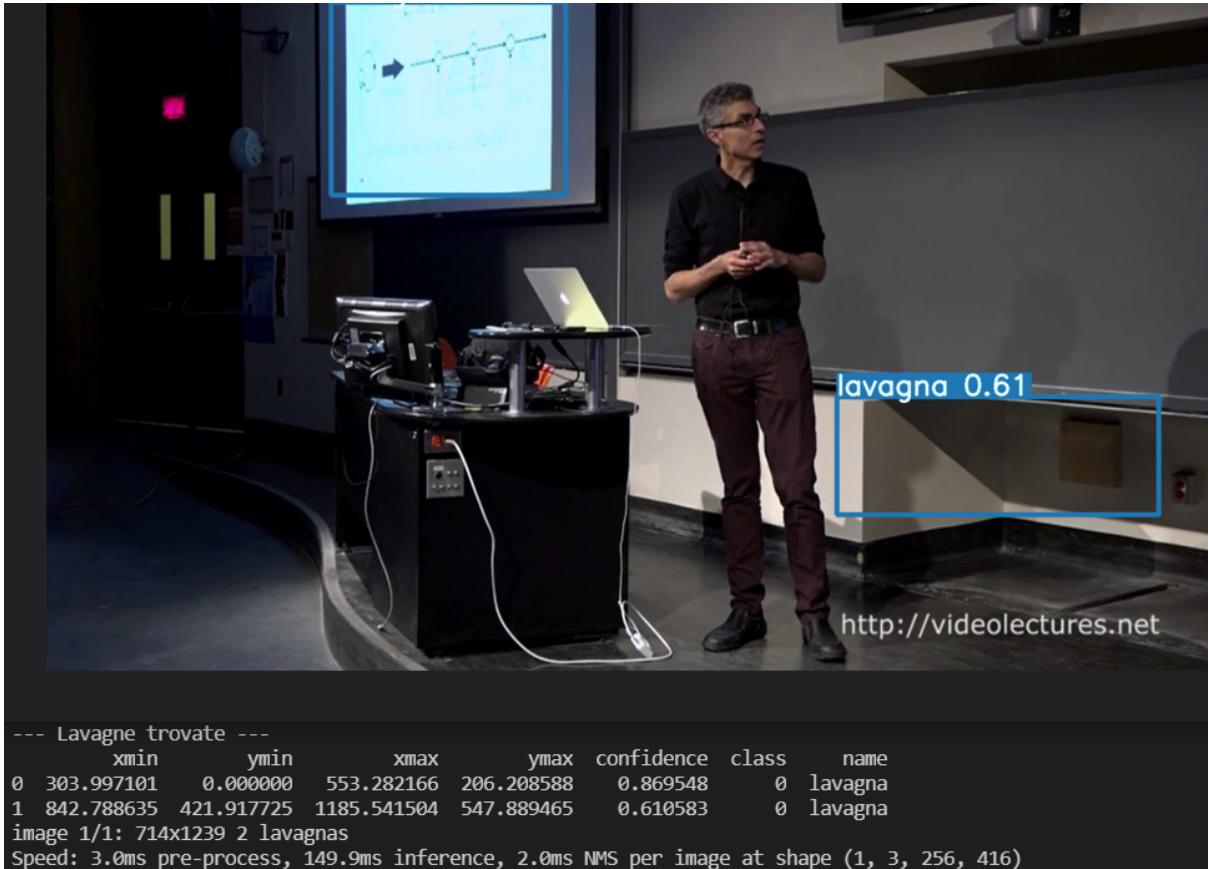
Sometimes it works



Sometimes it doesn't.

$$D = \frac{1}{C} \frac{1}{\ell} \frac{d\ell}{dt} = \frac{1}{C} \frac{1}{P} \frac{dP}{dt}$$
$$D^2 = \frac{1}{P^2} \frac{P_0 - P}{P} \sim \frac{1}{P^2} \quad (1a)$$
$$D^2 = \frac{K\varrho}{3} \frac{P_0 - P}{P} \sim -K\varrho \quad (2a)$$
$$D^2 \sim 10^{-53}$$
$$\varrho \sim 10^{-26}$$
$$P_0 \sim 10^8 \text{ g/cm}^3$$
$$t \sim 10^{10} (10^{11}) \text{ s}$$

```
--- Lavagne trovate ---
Empty DataFrame
Columns: [xmin, ymin, xmax, ymax, confidence, class, name]
Index: []
image 1/1: 2736x3648
Speed: 3.0ms pre-process, 161.9ms inference, 1.0ms NMS per image at shape (1, 3, 320, 416)
```



Sometimes the problem is that the value of confidence detected can be ambiguous, because sometimes when we have blackboards, it has low confidence and when we don't have blackboards, it has high confidence (for example the previous screenshot has detected the projected slide as blackboard with high confidence 86,9%).

SHALLOW LEARNING

In the end, we opted for this approach since we didn't have a big dataset to train our model. We've followed this paper and adapted to our problem:

https://scholar.google.com/citations?view_op=view_citation&hl=en&user=AvJA648AAAAJ&alert_preview_top_rm=2&citation_for_view=AvJA648AAAAJ:nb7KW1ujOQ8C

Data collection

We took screenshots from various videos from the web and split them into 4 categories.

- Blackboard
- Slide
- Slide-and-talk
- Talk

Feature engineering

We analyzed the data we gathered and extracted 19 normalized features with respect to the image resolution.

- Color histogram divided into 16 groups of colors
- People's face presence (0 = no people's face, 1 = one face, 2 = two or more faces)
- Position (xmedio) of the person's face
- Faces' size (area of the bounding box)

ML Algorithm

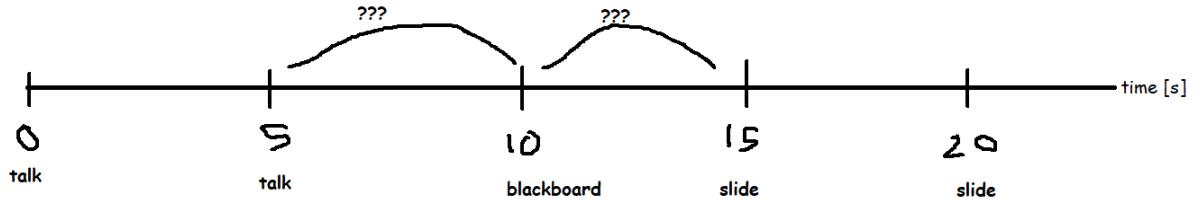
The paper used the SVM algorithm to train the model for the classification problem. We've tried it with Optuna (hyperparameter optimization framework) and it performed around 60-70% of accuracy.

So we've changed our approach and used the XGBoost algorithm along with Optuna:

- First time with 150 trials, we obtained 83,9% of accuracy with a lot of uncertainty in the classification
- Second time with 500 trials, we obtained 81,6% of accuracy but with less uncertainty

Split in segments

The stakeholder wanted to split the input video into segments of the same detected scenes. The split has a starting time and an ending time and between the segments we have an instance of time (secondsForFrameSkip variable) where it isn't classified.



In this example we have the first two frames classified as “TALK” but when we detected the third frame as “BLACKBOARD” we don’t know how the frames between 5 and 10 are classified. We can suppose many things like taking the average (from time 5 to 7,5 classified as TALK and time 7,5 to 10 as BLACKBOARD). It depends on how precise we want to achieve.

CONCLUSION

Since we **don't have much data**, the **optimal solution** is the **shallow learning** approach.

TEXT RECOGNITION

TESSERACT OCR

Since TensorFlow Object Detection API doesn't contain text detection, we opted to use Tesseract OCR as an extension module to detect texts in images.

WORKFLOW STEPS

1. Setup
 - a. pip install pytesseract
 - b. download tesseract.exe <https://github.com/UB-Mannheim/tesseract/wiki>
 - c. install it
2. In VScode

```
import cv2
import pytesseract

pytesseract.pytesseract.tesseract_cmd = "C:\\\\Program
Files\\\\Tesseract-OCR\\\\tesseract.exe"
img = cv2.imread('image.jpg')
# img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
print(pytesseract.image_to_string(img))
cv2.imshow("Result", img)
cv2.waitKey(0)
```

This code prints out the texts present in the given image. It works well with good contrasts and standard texts.

IMPLEMENTATION

The code is splitted into many Python files.

- main.py is the main script which connects all the modules used
- calculateFeatureAndTrain_module.py
- svm_module.py
- xgboost_module.py
- testTFmodel.py
- testYOLOv5-lavagne.py

main.py

```
4  # Controllo degli argomenti passati in linea di comando
5  if (len(sys.argv)<3):
6      print("ERRORE: inserisci il percorso del video da elaborare e ogni quanti secondi prendere un frame.")
7      sys.exit()
8
9 try:
10     secondsForFrameskip=int(sys.argv[2])
11 except:
12     print("ERRORE: inserire un numero intero come secondo parametro.")
13     sys.exit()
14
15 PATH_VIDEO = sys.argv[1]
16 if not os.path.exists(PATH_VIDEO):
17     print("ERRORE: video inesistente.")
18     sys.exit()
```

The first lines of code are used to check the arguments if they are valid or not.

```
33 matplotlib.use("TkAgg", force=True)
34 #%matplotlib inline
35 #plt.show()
36
37
38 CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
39 PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
40 PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
41 TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
42 LABEL_MAP_NAME = 'label_map.pbtxt'
43
44 > paths = {
45     'models': 'path_to_your_models',
46     'images': 'path_to_your_images',
47     'labels': 'path_to_your_labels',
48     'tfrecords': 'path_to_your_tfrecords'
49 }
50
51 > files = {
52     'tfrecord': 'path_to_your_tfrecord_file'
53 }
54
55 > def testConYoloV3(): ...
56
57
58
59
60
61
62
63
64
65
66
67
68
69 #####YOLO V3, non utilizzata perché rileva oggetti
70 > def testConYoloV3(): ...
```

These lines of code are no longer used; they were used to test Tensorflow's approach which gave poor performance.

The function “testConYoloV3()”

```

69 #####YOLO V3, non utilizzata perché rileva oggetti
70 def testConYoloV3():
71     # Load Yolo
72     print("LOADING YOLO")
73     net = cv2.dnn.readNet(files['YOLOV3_SPP_WEIGHTS'], files['YOLOV3_CFG'])
74     #save all the names in file o the list classes
75     classes = []
76     with open(files['COCO_NAMES'], "r") as f:      You, 2 months ago • trovato facce e salva le facce in cartella per
77         |   classes = [line.strip() for line in f.readlines()]
78
79     print(classes)
80
81     #get layers of the network
82     layer_names = net.getLayerNames()
83     #Determine the output layer names from the YOLO model
84     output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
85     print("YOLO LOADED")
86
87
88     # Capture frame-by-frame
89     IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'thumbsdown.b1f20c56-b4d4-11eb-ae88-240a64b78789.jpg')
90     #IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'thumbsup.fd7cdb14-b4d4-11eb-b662-240a64b78789.jpg')
91     img = cv2.imread(IMAGE_PATH)
92     #img=cv2.imread("test_img.jpg")
93     img = cv2.resize(img, None, fx=0.4, fy=0.4)
94     height, width, channels = img.shape
95
96     # Using blob function of opencv to preprocess image
97     #blob = cv2.dnn.blobFromImage(img, 1 / 255.0, (416, 416), swapRB=True, crop=False)
98
99     blob = cv2.dnn.blobFromImage(img, scaleFactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)
100
101     #for b in blob:
102     #    for n,img_blob in enumerate(b):
103     #        cv2.imshow(str(n),img_blob)
104
105     #Detecting objects
106     net.setInput(blob)

```

loads YoloV3's model and COCO's classes name, then we load a test image and apply some image filtering, this result is given to the model which we obtain the detection results to draw the bounding box.

The problem with YoloV3 is that if we have a person in the frame, the model detects the person multiple times. So we tried with the latest version YoloV5.

```

154 import yolov5
155 #import torch
156 # YOLOV5_model=os.path.join('tfod', 'Lib','site-packages', 'yolov5' , 'models', 'yolov5s.yaml'),
157 # print(os.getcwd()) # returns the currently working directory of a process
158 # model = yolov5.load(YOLOV5_model) # carico modello tramite formato yaml, però non funziona
159 # model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True) # scarico il modello
160 model = yolov5.load('yolov5s.pt') # se ho il modello in locale
161
162 def rilevaPersona(model, frame):
163     #####YOLO V5
164
165     isPerson=False
166     ## Immagine per test rilevazione persona
167     # IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'livelong.fed895dc-b264-11eb-bccc-086266b476b9.jpg')
168
169     # img = cv2.imread(frame)
170
171     # inference
172     #results = model(img)
173
174     results = model(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB), size=400)
175
176     labels, cord_thres = results.xyxy[0][:, :-1].numpy(), results.xyxy[0][:, :-1].numpy() # estraggo labels e coordinate dei
177     classes = model.names
178     label_class = []
179
180     # show results
181     for i in labels:
182         # print(classes[int(i)])
183         label_class.append(classes[int(i)])
184         if(classes[int(i)]=="person"):
185             isPerson=True
186
187     # show results
188
189

```

This function “rilevaPersona(model, frame)” is used in the project to detect people and takes as parameters the model which is “yolov5s.pt” and the image to be analyzed.

We've done the same thing as done for YoloV3 and if we detect a person we set our variable isPerson as True. This function returns isPerson variable and all the objects detected by the model.

class Record

```

263 @dataclass
264 class Record: # 1 frame
265     num_frame: int
266     time: float
267     list_words: list
268     isPersonDetected: bool
269     isSpeaker: bool
270     # path_faces: list      # ['faccia1', 'faccia2', 'faccia3']
271     idFaces: list          # [0,1,2] in questo caso ci sono 3 persone distinte e alla fine contare le occorrenze e scegliere il massimo
272     scene: any              # blackboard || slide || slide-and-talk || talk
273     objects_detected: any  # lista di oggetti rilevati tramite YoloV5
274
275     def to_dict(self):
276         return {"num_frame": self.num_frame, "time": self.time, "list_words": self.list_words, "isPersonDetected": self.isPersonDetected}
277

```

This class contains the information for each frame.

- num_frame: is the number of frame of the video that is going to be analyzed
- time: a float number that represents the time position of the frame in the video
- list_words: list of the words detected by the OCR
- isPersonDetected: boolean value that represents if there is a person in that frame
- isSpeaker: boolean value that represents if the main speaker is present in that frame
- idFaces: list of faces based on unique id
- scene: string value that represents the actual scene in the frame ('blackboard', 'slide', 'slide-and-talk', 'talk')
- objects_detected: list of objects detected by YoloV5

- function `to_dict(self)`: used to format the data in json format

class FrameWithFaces

```

282     @dataclass
283     class FrameWithFaces:
284         num_frame: int
285         num_faces: int
286         isProcessed: list
287         time: float

```

This class contains the information of a certain frame that contains faces. It is used to assign IDs to detected faces.

- `num_frame`: is the number of frame of the video that is going to be analyzed
- `num_faces`: is the number of faces in that frame
- `isProcessed`: list of boolean value that represents faces that are already processed, for example if the first face of the frame is already assigned to a specific ID, the first value of `isProcessed` list will be True
- `time`: a float number that represents the time position of the frame in the video

class Segment

```

289     @dataclass
290     class Segment:
291         id_label: any
292         start_time: float
293         end_time: float
294         list_of_same_scene: list      # lista di Records di quel tipo di scena
295
296         def to_dict(self):
297             return {"id_label": self.id_label, "start_time": self.start_time, "end_time": self.end_time, "list_of_same_scene": [ obj.to_dict

```

This class contains the information of specific segments of the video based on the scene detected.

- `id_label`: name of the scene ('blackboard', 'slide', 'slide-and-talk', 'talk')
- `start_time`: when the segment starts
- `end_time`: when the segment finishes
- `list_of_same_scene`: list of Record class that has the same 'scene' consecutively
- function `to_dict(self)`: used to format the data in json format

Video processing

```
324 # Carico il modello migliore per scene prediction
325 scene_model = pickle.load(open("xgboost500.sav", 'rb'))
326
327 while video.isOpened():
328     ret, frame = video.read()
329     if not ret:
330         break
331     totalFrame += 1
332     if i > frame_skip - 1: # In questo caso ogni 5 secondi
333         duration = frame_counter*temp
334         minutes = int(duration/60)
335         seconds = int(duration%60)
336         print(" ----- Frame at " + str(minutes) + " min and " + str(seconds) + " seconds ----- ")
337         n_frame_analyzed+=1
338         #cv2.imwrite('test_'+str(i)+'.jpg', frame)
339         #print(pytesseract.image_to_string(frame))
340         temp_list_words=[]
341         boxes=pytesseract.image_to_data(frame)
342         frame_counter+=frame_skip
343         #print("frame number: " + str(frame_counter))
344
345         # Riconoscimento persona con YoloV5
346         isPersonDetected, oggetti_rilevati = rilevaPersona(model, frame)
347         print("OGGETTI RILEVATI: ", oggetti_rilevati)
348         if isPersonDetected == True:
349             ...
350             import cv2
351             import mediapipe as mp
352             import numpy as np
353             mp_drawing = mp.solutions.drawing_utils
354             mp_drawing_styles = mp.solutions.drawing_styles
355             mp_pose = mp.solutions.pose
```

First of all we load the model we've previously trained to classify the scene.

Then we read the video given as input and analyze each frame. We use the pytesseract OCR to detect and extract the words present in the actual frame and store them in our data structure.

We use the function “rilevaPersona(model,frame)” as previously described and we check if there is any person; if yes, we launch mediapipe to extract their faces and save into a temporary image file, else we skip the extraction.

The commented code at line 349 can be used to extract the body landmarks points for further applications.

```
453 # Lista delle parole con OCR
454 for x,b in enumerate(boxes.splitlines()):
455     if x!=0:
456         b=b.split()
457         if len(b)==12: # gli indici prima del 12 sono altre informazioni del testo (colore, posizione, etc..)
458             #print(b[11])
459             #lista delle parole
460             temp_list_words.append(b[11])
```

OCR: detects words in the frame and stores them in our data structure.

```
462 # Inizializzo la lista delle facce per le facce univoche trovate con isProcessed
463 list_idFaces = []
464 for i in range(lunghezza_isProcessed):
465     list_idFaces.append(-1)
466
467 lunghezza_isProcessed = 1 # se non ci sono facce, la lista idFaces è -1 (anziché vuota)
```

All detected faces are initialized as value -1.

Then we have the scene prediction.

```
474     # SCENE PREDICTION
475     image_feature = calculateFeatureAndTrain_module.singleImageFeatureExtraction(img=frame)
476     image_feature = np.array([image_feature])
477     # Predict the response for test dataset
478     y_pred = scene_model.predict(image_feature)
479     y_pred_proba = scene_model.predict_proba(image_feature)
480
481     counter_scenes[int(y_pred)] += 1
482
483     print("Probabilità del tipo di scena:\t", y_pred_proba)
484
485     sorted_y_pred_proba = y_pred_proba
486     sorted_y_pred_proba.sort()
487     differenza_prob = sorted_y_pred_proba[0][-1]-sorted_y_pred_proba[0][-2]
488     print("Differenza tra le due probabilità più grandi:\t{:5f} ({:.2f}%)".format(differenza_prob, differenza_prob*100))
489     # print("Varianza della probabilità:\t", np.var(y_pred_proba))
490     print("Scene:\t", scenes[int(y_pred)])
```

By using the previously loaded model, we predict the actual scene by analyzing and passing the features of the actual frame.

We store all the information gathered to our value “rec” of type dataclass Record, then we check if we already have the same frame based on words detected and if there is a person.

```
494     #struct con numero del frame e la lista di parole
495     rec=Record(frame_counter,frame_counter*temp,temp_list_words, isPersonDetected, False, list_idFaces, scenes[int(y_pred)], oggetto)
496     #controlliamo se questo identico record è già contenuto nella lista
497     #potrebbe essere dispendioso, facciamo solo il compare con l'ultimo frame in listOfRecords?
498     flag=False
499
500     # conta se ci sono frames con la lista delle parole uguali
501     for y in listOfRecords:
502         if(y.list_words == rec.list_words):
503             flag=True
504             count_frame_doppi += 1
505             break
506
507     # listOfRecords.append(rec) # Inserisco ogni Record anche se doppione
508
509
510     if rec.isPersonDetected == True: #se c'è una persona
511         listOfRecords.append(rec)
512     else: # se NON c'è una persona
513         if flag == False: # ma le parole sono diverse
514             listOfRecords.append(rec)
515
516
517     # # SCOMMENTARE PER AVERE I RECORD UNIVOCI IN BASE ALLE PAROLE
518     # if(flag==False):
519     #     listOfRecords.append(rec) # avremo listOfRecords UNIVOCI
```

Comparing faces

```
529 | # CICLO CHE VA A COMPARARE TUTTE LE FACCE
530 | # Compara le due foto e definisce se le due persone trovate sono la stessa persona
531 | # result = DeepFace.verify(img1_path = "images/lec3.jpg", img2_path = IMAGE_FACES_PATH+"face_of_lec2_face0.jpg")
532 | # print(result)
533 | # print(frame_with_faces)
534 |
535 | from os import listdir
536 | from os.path import isfile, join
537 |
538 | mypath = 'images/faces/'
539 | onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
540 | # print(onlyfiles)
541 |
542 |
543 | FILENAME = os.path.splitext(NOME_VIDEO)[0]
544 | idFace = -1
545 | totalFace=[]
546 |
547 | # Loop che compara le facce trovate e conta il numero di occorrenze della relativa faccia
548 | for i in range(len(frame_with_faces)):
549 |     frame = frame_with_faces[i]
550 |     for n_face in range(frame.num_faces):
551 |         # La prima immagine di faccia
552 |         path_img = 'images/faces/face_of_' + FILENAME + '_frame_' + str(frame.num_frame) + '_face_' + str(n_face) + '.jpg'
553 |
554 |         if frame_with_faces[i].isProcessed[n_face] == False: # assegnamo una faccia se non è stata processata
555 |
556 |             try:
557 |                 obj = DeepFace.verify(img1_path=path_img, img2_path=path_img) # se trova una faccia continua, se no exception
558 |                 idFace+=1
559 |                 totalFace.append(1)
560 |                 frame_with_faces[i].isProcessed[n_face] = True
561 |                 for k in range(i+1, len(listOfRecords)):
562 |                     if(listOfRecords[k].time == frame_with_faces[i].time):
563 |                         listOfRecords[k].idFaces[n_face] = idFace
564 |                         break
565 |
566 |
567 |             for j in range(i+1, len(frame_with_faces)):
```

Once we've analyzed all the frames, we start to compare the faces to detect who is the host/speaker.

Our assumption is that the person who appears the most in the video is the host/speaker. At this point we will have a folder full of faces to be compared to each other. We use "[DeepFace](#)" to check if a face belongs to a certain person.

We start by picking the first image face and compare it to all the rest of the image faces. When we find that the comparison fits together, we assign the same ID to the face we are comparing. We keep track of the faces that are already processed to optimize the algorithm. After all the comparisons, all the faces have an ID assigned.

```
599 | idSpeaker=totalFace.index(max(totalFace))
600 | maxSpeakerApparence=max(totalFace) # numero delle apparizioni dello speaker
601 |
602 | counter=0
603 | # Setta isSpeaker a True per ogni record che contiene gli ID della faccia che è comparsa più volte
604 | for i in range(len(listOfRecords)):
605 |     for j in range(len(listOfRecords[i].idFaces)):
606 |         if listOfRecords[i].idFaces[j]==idSpeaker:
607 |             listOfRecords[i].isSpeaker=True
608 |             counter+=1
609 |             break
610 |         if counter==maxSpeakerApparence:
611 |             break
```

We extract the ID of the speaker by taking the index of the highest value in the "totalFace" array. For example if totalFace is [7,10,3] it means that we have 3 different faces, the first one appeared 7 times, the second 10 times and third 3 times. Then we take the highest value which is 10 in position 1 of the array. So the face with ID = 1 is the speaker's face.

Now we set in our list of Records, all the “isSpeaker” to True if the record has the ID face = 1.

Video segment

We create the Segment structure and group all the consecutive frames with the same scene. In the end, we print the video summary which includes the array of all the detected faces, the id of the speaker, the percentage of the scenes and the information related to the segments.

After all those face file images generated, we proceed to delete them because they're no longer used.

```
679 # DELETE FACE IMAGES
680 import os, shutil
681 for filename in os.listdir(IMAGE_FACES_PATH):
682     file_path = os.path.join(IMAGE_FACES_PATH, filename)
683     try:
684         if os.path.isfile(file_path) or os.path.islink(file_path):
685             os.unlink(file_path)
686         elif os.path.isdir(file_path):
687             shutil.rmtree(file_path)
688     except Exception as e:
689         print('Failed to delete %s. Reason: %s' % (file_path, e))
```

We had to save the images because DeepFace doesn't take as input the image itself but only the path to the image.

In conclusion, we write the information of the list of segments which contains all the information of the segments and the related frame's data to a text file in JSON format.

```
693 | # # Write video data in json output file
694 | import json
695 | results = [obj.to_dict() for obj in listofSegments]
696 | print(results)
697 | with open('data_'+FILENAME+'.txt', 'w') as outfile:
698 |     json.dump(results, outfile)
699 |
700 |
701 | # # Read video data from json file
702 | # with open('data_prova2persone.txt') as f:
703 | #     data = json.load(f)
704 |
705 | # print(type(data))
706 |
707 | # # Example of reading data structure from JSON file
708 | # for i in data:
709 | #     for j in i["list_of_same_scene"]:
710 | #         for k in j["list_words"]:
711 | #             if k=='\\':
712 | #                 print("!!!!!!")
713 | #     #print(i)
```

We've also included some lines of code to read the text file and to access the data structure.

calculateFeatureAndTrain_module.py

This module is used to calculate our features for the machine learning model.

```
20 | # Plot the color histogram divided in 16 groups
21 | def plotList(array):
22 |     objects = ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15')
23 |     y_pos = np.arange(len(objects))
24 |     performance = array
25 |
26 |     plt.bar(y_pos, performance, align='center', alpha=0.5)
27 |     plt.xticks(y_pos, objects)
28 |
29 |     plt.show()
```

This function plots the color histogram.

```
31 | # Print the confusion matrix
32 | def confusionMatrix(y_test, y_pred, actions):
33 |     print("\nCONFUSION MATRIX\n")
34 |
35 |     # print(y_pred)
36 |     # print(y_pred.argmax(axis=1))
37 |
38 |     # Model Accuracy, how often is the classifier correct?
39 |     # print("metrics.accuracy score 1D (non tiene conto dei null):\t", metrics.accuracy_score(y_test.argmax(axis=1), y_pred.argmax(axis=1)))
40 |     print("metrics.accuracy score normale (considera i null sbagliati) - Testing score:\t", metrics.accuracy_score(y_test, y_pred))
41 |
42 |     matrix = [[0 for x in range(len(actions))] for y in range(len(actions))]
43 |     errori = 0
44 |     predNull = 0
45 |     for i in range(0,len(y_test)):
46 |         if 1 in y_pred[i]:
47 |             if(np.argmax(y_pred[i]) != np.argmax(y_test[i])):
48 |                 matrix[np.argmax(y_test[i])][np.argmax(y_pred[i])] = matrix[np.argmax(y_test[i])][np.argmax(y_pred[i])] + 1
49 |                 errori+=1
50 |                 #print("valore predetto per campione "+ str(i)+ " : "+str(actions[np.argmax(y_pred[i])])) #prediction
51 |                 #print("valore effettivo per campione "+ str(i)+ " : "+str(actions[np.argmax(y_test[i])]))+"\n") #valore effettivo
52 |             else:
53 |                 matrix[np.argmax(y_test[i])][np.argmax(y_pred[i])] = matrix[np.argmax(y_test[i])][np.argmax(y_pred[i])] + 1
54 |         else:
55 |             errori+=1
56 |             predNull+=1
57 |
58 |     mat=pd.DataFrame(np.row_stack(matrix))
59 |     col=[]
60 |     for i in range(len(actions)):
61 |         col.append(actions[i])
62 |
63 |     mat.columns=col
64 |     mat.index=actions
65 |
66 |     print("numero campioni di test: "+str(len(y_pred))+" campioni erroneamente classificati: "+str(errori) + " campioni classificati")
67 |     print(mat)
68 |
69 |     # print("\nmetrics.confusion_matrix")
70 |     # print(confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1)))
```

This confusionMatrix function calculates and prints the confusion matrix.

```

73 | # Extract features from multiple images in a folder
74 | def featureExtraction(IMAGE_TRAIN_PATH):
75 |     PROJECT_PATH=pathlib.Path(__file__).parent.resolve() #restituisce il path del progetto
76 |     FINAL_IMAGE_TRAIN_PATH=os.path.join(PROJECT_PATH,IMAGE_TRAIN_PATH)
77 |
78 |     list_subfolders_with_paths = [f.path for f in os.scandir(FINAL_IMAGE_TRAIN_PATH) if f.is_dir()]
79 |
80 |     numero_gruppi = 16
81 |
82 |
83 |     x_train = []
84 |     y_train = []
85 |
86 |     for directory in list_subfolders_with_paths:
87 |
88 |         #cancella le 2 righe sotto al commento
89 |         #cartella=os.path.basename(os.path.normpath(directory))
90 |         #if cartella=="noAlzateLaterali":
91 |             #directory=FULL_VIDEO_PATH
92 |         print("sto processando le img in: "+ directory )
93 |         for file in os.listdir(directory):
94 |             path_directory = os.path.join(FINAL_IMAGE_TRAIN_PATH, directory)
95 |             filename = os.fsdecode(file)
96 |             final_file_path = os.path.join(path_directory, filename)
97 |
98 |             img = cv.imread(final_file_path,0)
99 |             hist = cv.calcHist([img],[0],None,[256],[0,256])
100 |
101 |             dimensione_gruppo = len(hist) / numero_gruppi
102 |
103 |             counter = 0
104 |             array_of_hist = []
105 |             sum = 0
106 |             for i in range(0, len(hist)):
107 |                 counter+=1
108 |
109 |                 sum = sum + hist[i][0].astype(int)

```

This is one of our main functions used to extract the features from multiple images contained in the ‘images/train’ folder within the workspace.

In this ‘train’ folder we have other 4 folders called ‘blackboard’, ‘slide’, ‘slide-and-talk’, ‘talk’ and inside these folders we have all the images used to train the model.

The function selects all these images and extracts their features and annotates them to their category.

The features extracted are:

- Color histogram divided into 16 groups of colors
- People’s face presence (0 = no people’s face, 1 = one face, 2 = two or more faces)
- Position (xmedio) of the person’s face
- Faces’ size (area of the bounding box)

This function returns two lists:

- X_train: contains all the features of the images
- y_train: contains the category which is associated (0 = blackboard, 1 = slide, 2 = slide-and-talk, 3 = talk)

```

175 | # Extract features from a single image (either image path or the image already read)
176 | def singleImageFeatureExtraction(final_file_path = None, img = None):
177 |     numero_gruppi = 16
178 |     flagException = False
179 |
180 |     try:
181 |         if (final_file_path == None) and (img == None):
182 |             flagException = True
183 |     except:
184 |         pass
185 |
186 |     try:
187 |         if ((final_file_path == None) and (img.all() == None)):
188 |             flagException = True
189 |     except:
190 |         pass
191 |
192 |     if flagException == True:
193 |         raise Exception("inserire un parametro alla funzione singleImageFeatureExtraction")
194 |
195 |     if final_file_path != None:
196 |         img = cv.imread(final_file_path,0)
197 |
198 |     hist = cv.calcHist([img],[0],None,[256],[0,256])
199 |
200 |     dimensione_gruppo = len(hist) / numero_gruppi
201 |
202 |     counter = 0
203 |     array_of_hist = []
204 |     sum = 0
205 |     for i in range(0, len(hist)):
206 |         counter+=1
207 |
208 |         sum = sum + hist[i][0].astype(int)
209 |         if counter == dimensione_gruppo:
210 |             array_of_hist.append(sum)
211 |             counter = 0
212 |             sum = 0

```

This function is similar to the previous one, but it takes only 1 image path or 1 image already read.

It is used to extract the single image's features, then passed to the prediction of the model. This function returns the list of features.

```

263 | # Print the model's score
264 | def print_model_score(model):
265 |     x_train, y_train = featureExtraction(IMAGE_TRAIN_PATH)
266 |     cross_score = cross_val_score(model, x_train, y_train, cv=5)
267 |     print("Model's score: %f accuracy with a standard deviation of %f" % (cross_score.mean(), cross_score.std()))

```

This function is used to evaluate our model.

```
270 | # Function to train the XGBoost model
271 | def train():
272 |     X_train, y_train = featureExtraction(IMAGE_TRAIN_PATH)
273 |     X_test, y_test = featureExtraction(IMAGE_TEST_PATH)
274 |
275 |
276 |     # # SVM training
277 |     # print("# SVM --- Best HP training: ")
278 |     # study=svm_module.findBestHyperparameters(X_train, y_train)
279 |     # model=svm_module.train(X_train, y_train, study.best_params)
280 |     # y_pred = model.predict(X_test)
281 |     # print("----- y_pred")
282 |     # print(y_pred)
283 |
284 |
285 |     scenes = [0,1,2,3]
286 |     # # confusionMatrix(y_test, y_pred, scenes)
287 |     # print(confusion_matrix(y_true=y_test, y_pred=y_pred, labels=scenes))
288 |
289 |
290 |     # XGBoost training
291 |     print("# XGBoost --- Best HP training: ")
292 |     study=xgboost_module.findBestHyperparameters(X_train, y_train)
293 |     model=xgboost_module.train(X_train, y_train, study.best_params)
294 |     y_pred = model.predict(X_test)
295 |     print_model_score(model)
296 |     print(confusion_matrix(y_true=y_test, y_pred=y_pred, labels=scenes))
```

This function is used to train our machine learning model. We use specifically the extreme gradient boosting algorithm along with Optuna (hyperparameters optimization framework) to train the model.

svm_module.py

This module was used the first time to test the paper's approach. But later we switched to 'xgboost_module.py'.

```
7 | # Transform from one hot encoding to 1-dimensional list
8 | def oneHot_to_1D(y):
9 |
10 |     y_final = []
11 |     for i in range(0,len(y)):
12 |         for j in range(0, 4):
13 |             if y[i,j] == 1:
14 |                 y_final.append(j)
15 |                 break
16 |
17 |     return y_final
18 |
19 |
20 | # Transform from 1-dimensional list to one-hot-encoding
21 | def oneD_to_oneHot(y):
22 |     y_final = []
23 |     for i in range(0,len(y)):
24 |         if y[i] == 0:
25 |             y_final.append([1,0,0,0])
26 |         elif y[i] == 1:
27 |             y_final.append([0,1,0,0])
28 |         elif y[i] == 2:
29 |             y_final.append([0,0,1,0])
30 |         else:
31 |             y_final.append([0,0,0,1])
32 |     return y_final
```

These functions are used to switch from one-hot-encoding to 1-dimensional list and viceversa.

```

35 | # Used for cross validation score
36 | def train_and_score(X_train, X_test, y_train, y_test):
37 |
38 |     y_train_new = y_train
39 |     y_test_new = y_test
40 |
41 |     clf = svm.SVC(random_state=0)
42 |     clf.fit(X=X_train, y=y_train_new)
43 |     y_pred = clf.predict(X_test)
44 |     score = clf.score(X_test,y_test_new)
45 |
46 |     ac_score = metrics.accuracy_score(y_test_new, y_pred)
47 |     print("\nSVM score:\t", score)
48 |     print("metrics.accuracy score SVM:\t"+ str(ac_score))
49 |
50 |     cross_score = cross_val_score(clf, X_train, y_train_new, cv=5)
51 |     print("cross val score SVM:\t\t" + str(cross_score))
52 |     print("cross val score SVM mean:\t" + str(cross_score.mean()))
53 |     print("%f accuracy with a standard deviation of %f" % (cross_score.mean(), cross_score.std()))
54 |
55 |     y_pred = oneD_to_oneHot(y_pred)
56 |     return y_pred
57 |

```

This function is used to calculate the cross validation score and returns the prediction.

```

59 | # Train and save to file the model
60 | def train(X_train,y_train,best_params):
61 |
62 |     model = svm.SVC(**best_params, random_state=0)
63 |     model.fit(X_train, y_train) # Training del modello con i dati
64 |
65 |     cross_score = cross_val_score(model, X_train, y_train, cv=5) # training accuracy
66 |     print("best: %f accuracy with a standard deviation of %f" % (cross_score.mean(), cross_score.std()))
67 |     # save the model to disk
68 |     filename = 'svm.sav'
69 |     pickle.dump(model, open(filename, 'wb'))
70 |
71 |     return model

```

This is used to train and obtain our model.

```

74 |     # Find the best hyperparameters with Optuna
75 |     def findBestHyperparameters(X_train, y_train):
76 |
77 |         def objective(trial):
78 |             dtc_params = dict(
79 |                 kernel=trial.suggest_categorical('kernel',['rbf','poly','linear','sigmoid']),
80 |                 C=trial.suggest_float("C",0.1,3.0,log=True),
81 |                 gamma=trial.suggest_categorical('gamma',[ 'auto','scale']),
82 |                 degree=trial.suggest_int("degree",1,3,log=True),
83 |             )
84 |             DTC = svm.SVC(**dtc_params, random_state=0)
85 |             cross_score = cross_val_score(DTC, X_train, y_train, cv=5)
86 |             error = 1.0 - cross_score.mean()
87 |
88 |
89 |
90 |             # 3. Create a study object and optimize the objective function.
91 |             study = optuna.create_study() # di default è minimize, quindi di minimizzare l'errore
92 |             study.optimize(objective, n_trials=1000)
93 |
94 |             print(study.best_params) # Printa i migliori parametri
95 |             print(1.0 - study.best_value) # Printa l'accuracy
96 |             return study

```

This function is used to find the best hyperparameters using Optuna, which tries to find the optimal hyperparameters combination by minimizing the error.

xgboost_module.py

This module is used to train our xgboost model.

```

8 > def oneHot_to_1D(y): ...
18
19 > def oneD_to_oneHot(y): ...

```

These are the same functions used by svm_module.py to switch between one-hot-encoding and 1-dimensional list.

```

33 def train_and_score(X_train, X_test, y_train, y_test):
34
35     y_train_new = y_train
36     y_test_new = y_test
37
38     clf = XGBClassifier(use_label_encoder=False, eval_metric = 'mlogloss', random_state=0)
39     clf.fit(X=X_train, y=y_train_new)
40     y_pred = clf.predict(X_test)
41     score = clf.score(X_test,y_test_new) # testing accuracy
42
43     ac_score = metrics.accuracy_score(y_test_new, y_pred)
44     print("\nExtreme gradient boosting score:\t", score)
45     print("metrics.accuracy score XGBoost:\t"+ str(ac_score))
46
47     cross_score = cross_val_score(clf, X_train, y_train_new, cv=5) # training accuracy
48     print("cross val score XGBoost:\t\t" + str(cross_score)) # array di 5 elementi
49     print("cross val score XGBoost mean:\t" + str(cross_score.mean()))
50     print("%f accuracy with a standard deviation of %f" % (cross_score.mean(), cross_score.std()))
51
52     y_pred = oneD_to_oneHot(y_pred)
53
54     return y_pred # testing accuracy

```

This function is used to calculate the cross validation score and returns the prediction.

```

57 def train(X_train,y_train,best_params):
58
59     model = XGBClassifier(**best_params, use_label_encoder=False, eval_metric = 'mlogloss', random_state=0)
60     model.fit(X_train, y_train) # Training del modello con i dati
61
62     cross_score = cross_val_score(model, X_train, y_train, cv=5) # training accuracy
63     print("best: %f accuracy with a standard deviation of %f" % (cross_score.mean(), cross_score.std()))
64     # save the model to disk
65     filename = 'xgboost500.sav'
66     pickle.dump(model, open(filename, 'wb'))
67
68     return model

```

This function is used to train with XGBoost algorithm and save the model

```

71 def findBestHyperparameters(X_train, y_train):
72
73     def objective(trial):
74         dtc_params = dict(
75             max_depth=trial.suggest_int("max_depth", 2, 10),
76             learning_rate=trial.suggest_float("learning_rate", 1e-4, 1e-1, log=True),
77             n_estimators=trial.suggest_int("n_estimators", 100, 2000),
78             min_child_weight=trial.suggest_int("min_child_weight", 1, 10),
79             colsample_bytree=trial.suggest_float("colsample_bytree", 0.2, 1.0),
80             subsample=trial.suggest_float("subsample", 0.2, 1.0),
81             reg_alpha=trial.suggest_float("reg_alpha", 1e-4, 1e2, log=True),
82             reg_lambda=trial.suggest_float("reg_lambda", 1e-4, 1e2, log=True),
83             gamma=trial.suggest_float("gamma", 0, 50),
84         )
85         DTC = XGBClassifier(**dtc_params, use_label_encoder=False, eval_metric = 'mlogloss', random_state=42)
86         cross_score = cross_val_score(DTC, X_train, y_train, cv=5)
87         error = 1.0 - cross_score.mean()
88
89     return error
90
91
92     # 3. Create a study object and optimize the objective function.
93     study = optuna.create_study() # di default è minimize, quindi di minimizzare l'errore
94     study.optimize(objective, n_trials=500)
95
96     print(study.best_params) # Printa i migliori parametri
97     print(1.0 - study.best_value) # Printa l'accuracy
98     return study

```

This is used to find the best hyperparameters for our model by minimizing the error.

testTFmodel.py

This script is used to test the Tensorflow model trained to recognize the blackboards as previously cited.

```

36 IMAGE = "images/cas1.jpg"          You, a day ago • final touch
37 img = cv2.imread(IMAGE)
38 image_np = np.array(img)
39
40 input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
41 detections = detect_fn(input_tensor)
42
43
44 num_detections = int(detections.pop('num_detections'))
45 detections = {key: value[0, :num_detections].numpy()
46 |   |   |   for key, value in detections.items()}
47 detections['num_detections'] = num_detections
48
49 print("Num detections: ", num_detections)

```

To test an image, just change the path given to “IMAGE” variable and run the script.

testYOLOv5-lavagne.py

This script is used to test the YoloV5 custom trained to detect blackboards.

```
4  fileName = "models/best-senzaAug.pt"
5  model = yolov5.load(fileName) # se ho il modello in locale
6
7  # non funziona bene, rileva lavagne in ogni immagine. Il problema può essere la scarsità e qualità del dataset.
8  def rilevaLavagna(model, frame):
9      model.conf = 0.6
10
11     isLavagna = False
12
13     results = model(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB), size=400)
14
15     print("--- Lavagne trovate ---")
16     print(results.pandas().xyxy[0])
17
18     labels, cord_thres = results.xyxy[0][:, -1].numpy(), results.xyxy[0][:, :-1].numpy() # estraggo labels e coordinate
19     classes = model.names
20
21     # show results
22     for i in labels:
23         # print(classes[int(i)])
24         if(classes[int(i)]=="lavagna"):
25             |   isLavagna=True
26
27     # show results
28     results.print()
29     results.show()
30     print(isLavagna)
31
32     frame = cv2.imread("images/lec8.jpg")
33     rilevaLavagna[model, frame] You, 2 months ago • per testare fare print(framewithfaces e listofrec...
```

The model ‘best-senzaAug.pt’ is trained using a Google Colab script where we’ve loaded the dataset.

You can test the model by changing the parameter at line 32.

HOW TO USE

```
Orphe@PC-Gabba MINGW64 ~/Desktop/Bibbia 2/Semantic Web Technologies/Progetto_SWT_TFOD/TFODCourse (main)
$ source "c:/Users/Orphe/Desktop/Bibbia 2/Semantic Web Technologies/Progetto_SWT_TFOD/TFODCourse/tfod/Scripts/activate"
(tfod)
Orphe@PC-Gabba MINGW64 ~/Desktop/Bibbia 2/Semantic Web Technologies/Progetto_SWT_TFOD/TFODCourse (main)
$ python main.py "C:\Users\Orphe\Desktop\Bibbia 2\Semantic Web Technologies\Progetto_SWT_TFOD\TFODCourse\prova2persone.mp4" 5
```

Run the main.py script in the terminal with 2 arguments: the first is the path of the video file (for example “C:/User/Video/filename.mp4”) and the second is an integer (in seconds) that represents the number of seconds we need to wait for the frame to be processed.
(Remember to activate your virtual environment before running the script)

OUTPUT FORMAT

Terminal output

```
----- Frame at 0 min and 5 seconds -----
YOLOv5: PERSONA RICONOSCIUTA
OGGETTI RILEVATI: ['person', 'laptop']
Mediapipe: Faccia rilevata.
isPersonDetected: True
Probabilità del tipo di scena: [[ 0.4767  0.090823  0.16924   0.26324]]
Differenza tra le due probabilità più grandi: 0.21346 (21.35%)
Scene: Blackboard

----- Frame at 0 min and 10 seconds -----
YOLOv5: PERSONA RICONOSCIUTA
OGGETTI RILEVATI: ['person', 'person', 'tv']
Mediapipe: Faccia rilevata.
Mediapipe: Faccia rilevata.
isPersonDetected: True
Probabilità del tipo di scena: [[ 0.32166  0.18542   0.22668   0.26625]]
Differenza tra le due probabilità più grandi: 0.05541 (5.54%)
Scene: Blackboard

----- Frame at 0 min and 15 seconds -----
YOLOv5: PERSONA RICONOSCIUTA
OGGETTI RILEVATI: ['person', 'person']
Mediapipe: Faccia rilevata.
Mediapipe: Faccia rilevata.
isPersonDetected: True
Probabilità del tipo di scena: [[ 0.30002  0.17747   0.22323   0.29929]]
Differenza tra le due probabilità più grandi: 0.00073 (0.07%)
Scene: Blackboard

----- Frame at 0 min and 20 seconds -----
YOLOv5: PERSONA RICONOSCIUTA
OGGETTI RILEVATI: ['person', 'person', 'laptop', 'tv']
Mediapipe: Faccia rilevata.
Mediapipe: Faccia rilevata.
isPersonDetected: True
Probabilità del tipo di scena: [[ 0.28661  0.21786   0.21342   0.28212]]
Differenza tra le due probabilità più grandi: 0.00449 (0.45%)
Scene: Blackboard
```

The screenshot above illustrates the output obtained from running the main script. It tells you that at second 5, YoloV5 has found a person in the frame, 2 objects detected based on COCO classes, you can notice that on seconds 10 we have 2 rows of “Mediapipe: Faccia rilevata.” this means that Mediapipe has found 2 faces in that frame. “Probabilità del tipo di scena” represents the probability of the scene in the given frame. As stated before, we have:

- first index as ‘blackboard’
- second index as ‘slide’
- third index as ‘slide-and-talk’
- forth index as ‘talk’

Then we have “Differenza tra le due probabilità più grandi” which is the difference between the top 2 largest probabilities obtained and this gives us a measure of certainty. If the

difference is big, then we will have a better certainty of the scene's prediction, else if the difference is small, then we will have a bit of uncertainty of the prediction.

Video summary

```
- - - - - VIDEO SUMMARY - - - - -
Le facce trovate sono: [5, 10, 1]
La faccia con più apparizioni è quella con id: 1 ed ha avuto 10 apparizioni con una percentuale di 34.48%

Scene:
Blackboard: 7 frames (24.14%)
Slide: 8 frames (27.59%)
Slide-and-talk: 0 frames (0.0%)
Talk: 14 frames (48.28%)

Segmenti:
Segmento 1: Blackboard da 0.0 a 35.0
Segmento 2: Slide da 40.0 a 50.0
Segmento 3: Talk da 55.0 a 120.0
Segmento 4: Slide da 125.0 a 148.0
- - - - -
```

At the end of video elaboration, we will have a video summary which gives us:

- Faces' found are: [5, 10, 1] — this means that in the entire video 3 unique faces are found, where the first face appeared 5 times, the second face appeared 10 times and the third face appeared once. Hence, our host/speaker will be the second face that appeared the most
- Scene: a summary of the scenes predicted for the entire video
- Segments: prints out each segment with its starting and ending time

JSON data structure format

List of objects (objects are Segment type)

```
|-- (for each Segment)
    |-- id_label
    |-- start_time
    |-- end_time
    |-- list_of_same_scene (list of Record objects)
        |-- (for each Record)
            |-- num_frame
            |-- time
            |-- list_words
            |-- isPersonDetected
            |-- isSpeaker
            |-- idFaces
            |-- scene
            |-- objects_detected
```

This data structure is saved as JSON format in a text file and you can read it with the following snippet code.

```
702 # Read video data from json file
703 with open('data_prova2persone.txt') as f:
704     data = json.load(f)
705
706 print(type(data))
707
708 # Example of reading data structure from JSON file
709 for i in data:
710     for j in i["list_of_same_scene"]:
711         for k in j["list_words"]:
712             if k=='\\':
713                 print("!!!!!!")
714     #print(i)
715
```