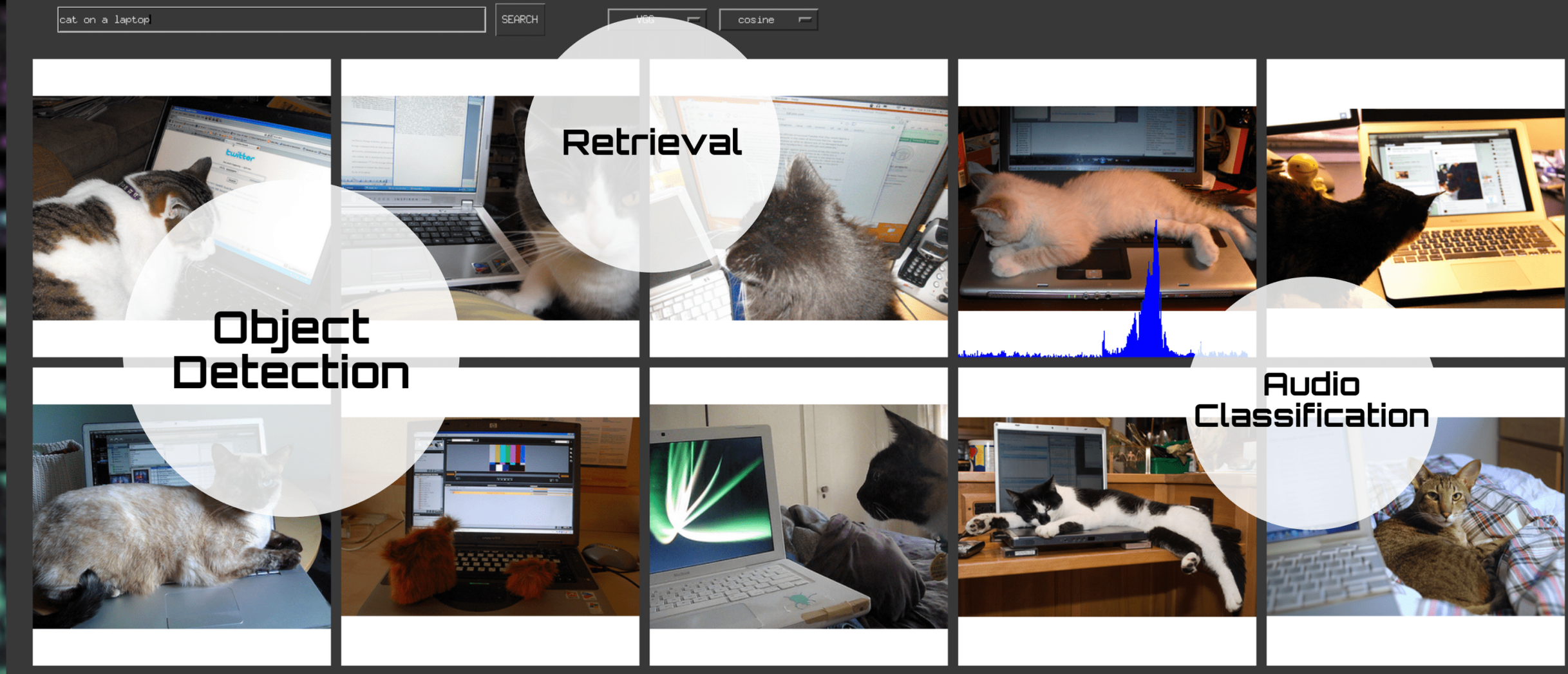
DSIM Project

Content-based IR

Cervero Riccardo 794126
Kasela Pranav 846966
Moiraghi Federico 799735

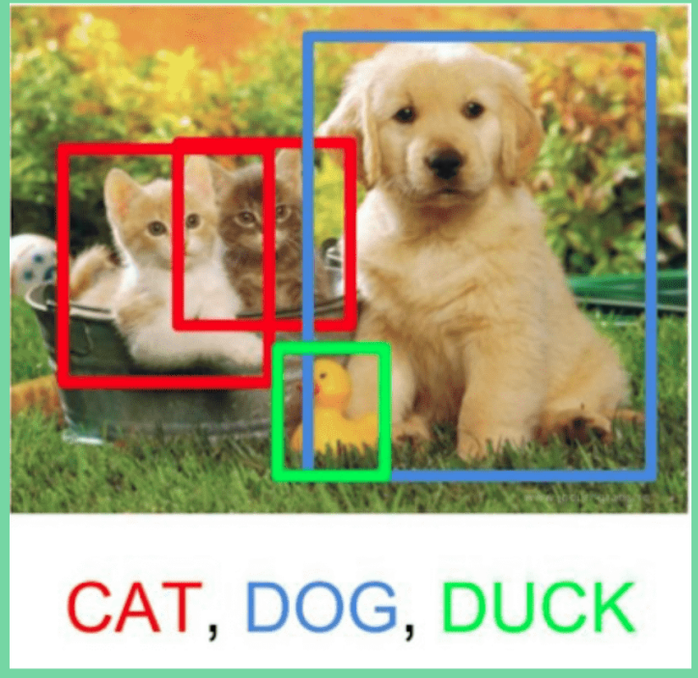
UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA




COCO
Common Objects in Context

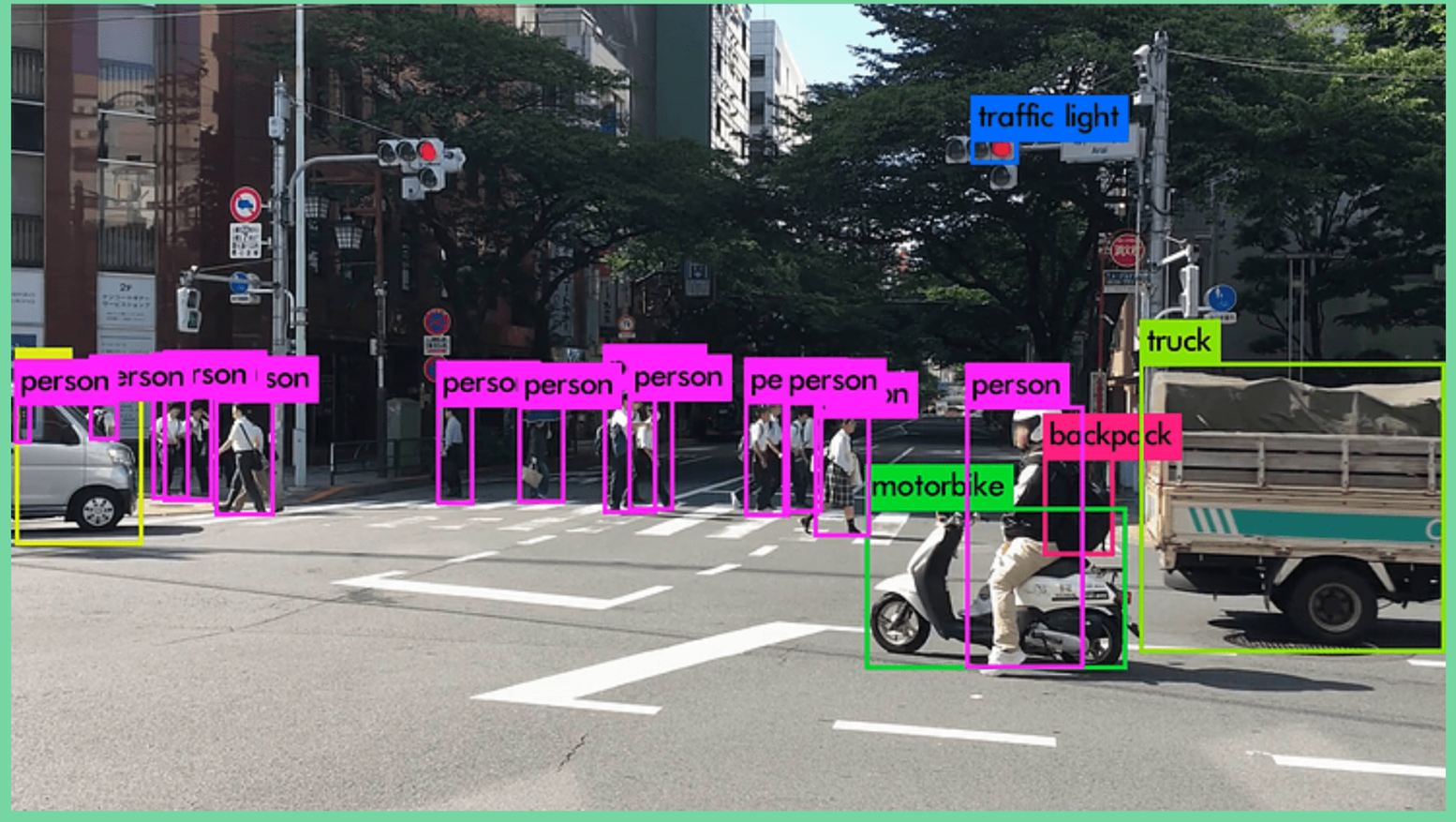
- Detection & Semantic Segmentation
- 328k images
- 80 object categories

Images are "multi-labelled" by tagging their elements



YOU ONLY LOOK ONCE 3v

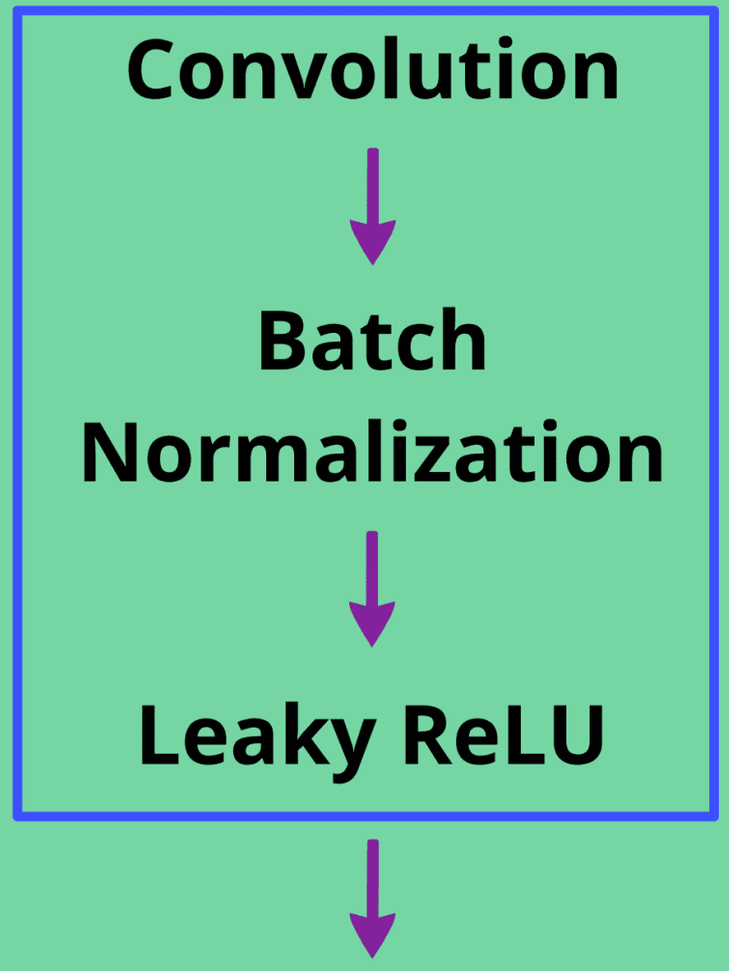
Pre-trained very fast algorithm to detect multiple object location inside images and videos



YOLOv3 Architecture

- Fully Convolutional Neural Network
- Not a R-CNN, but "looks once"
- Deeper feature extractor
- Invariant to size but need to fix dimensions
- Feature pyramid network

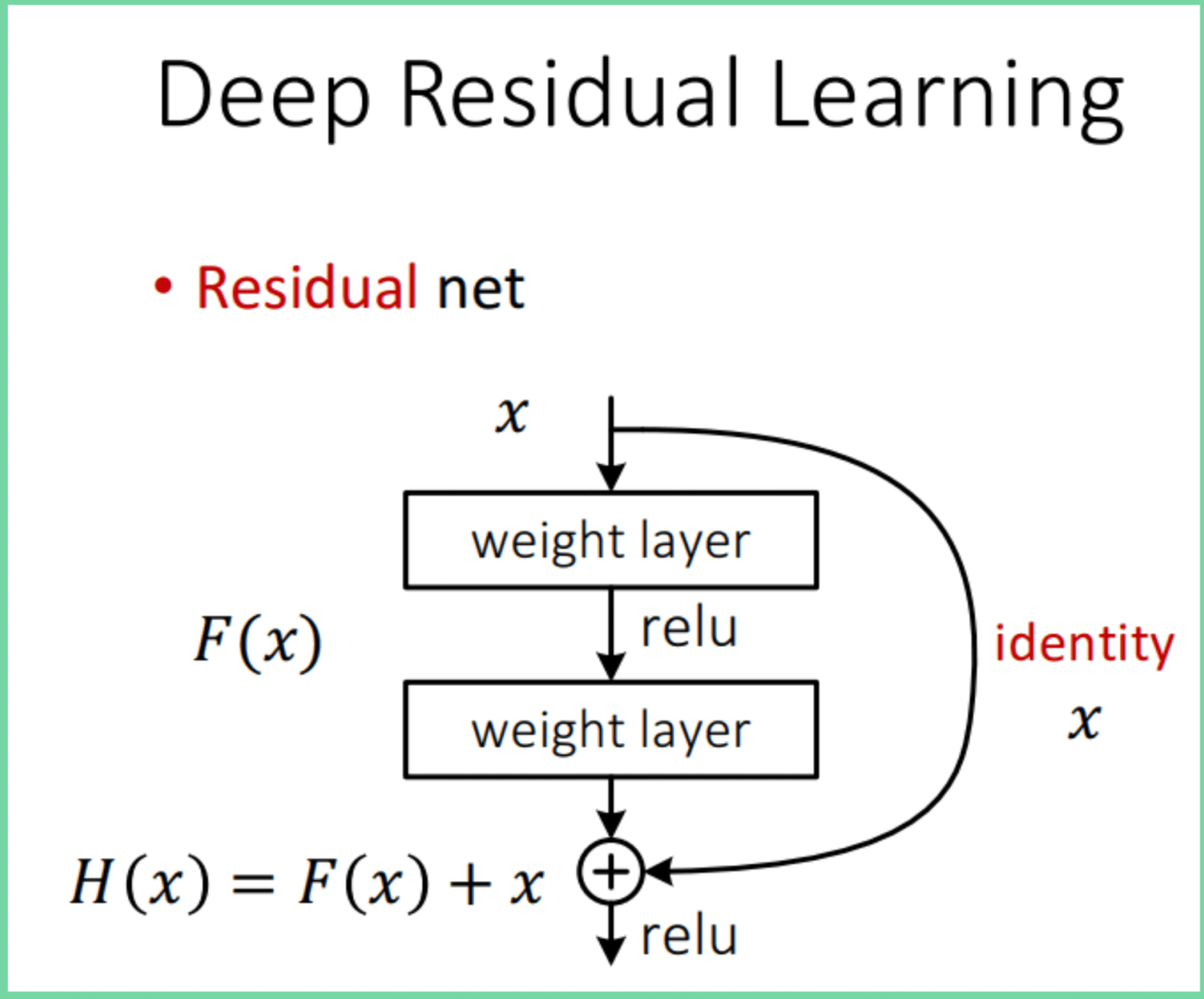
DarkNet Feature Extractor



Convolution 2x2

Residual block

ResNet-alike Architecture



Lower learning complexity

Detecting differently-sized objects

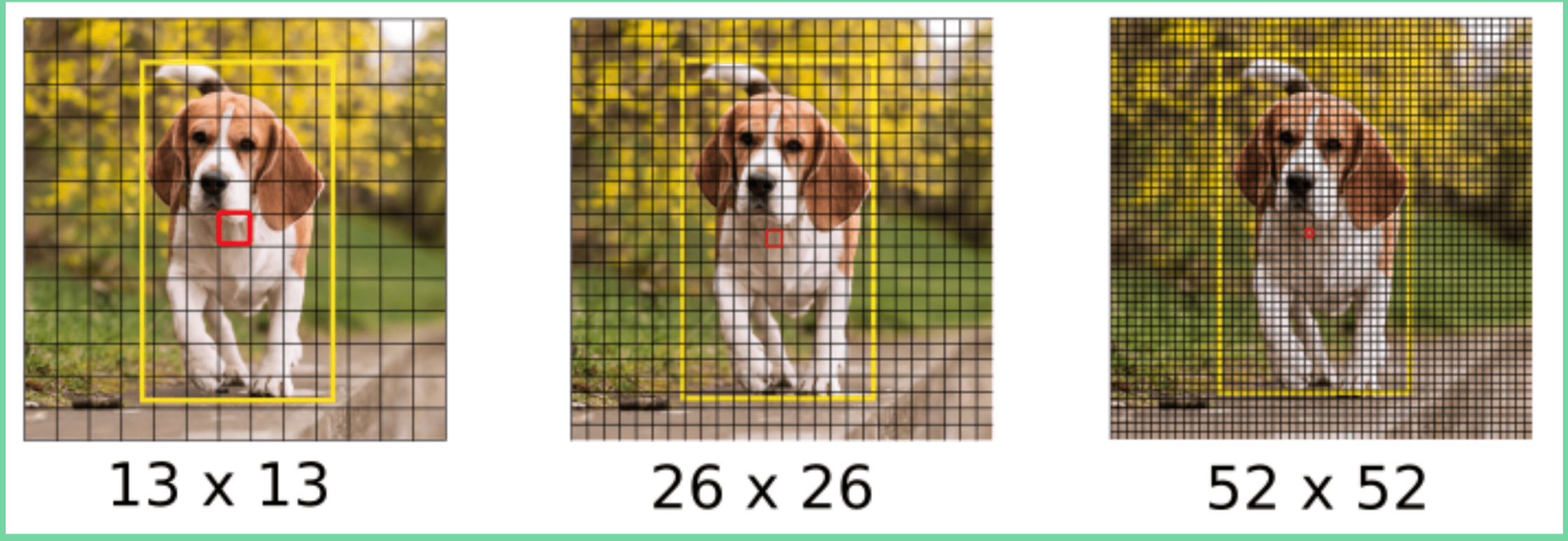
To get objects locations and class, network classifier produces a grid called "feature map"

Detecting differently-sized objects

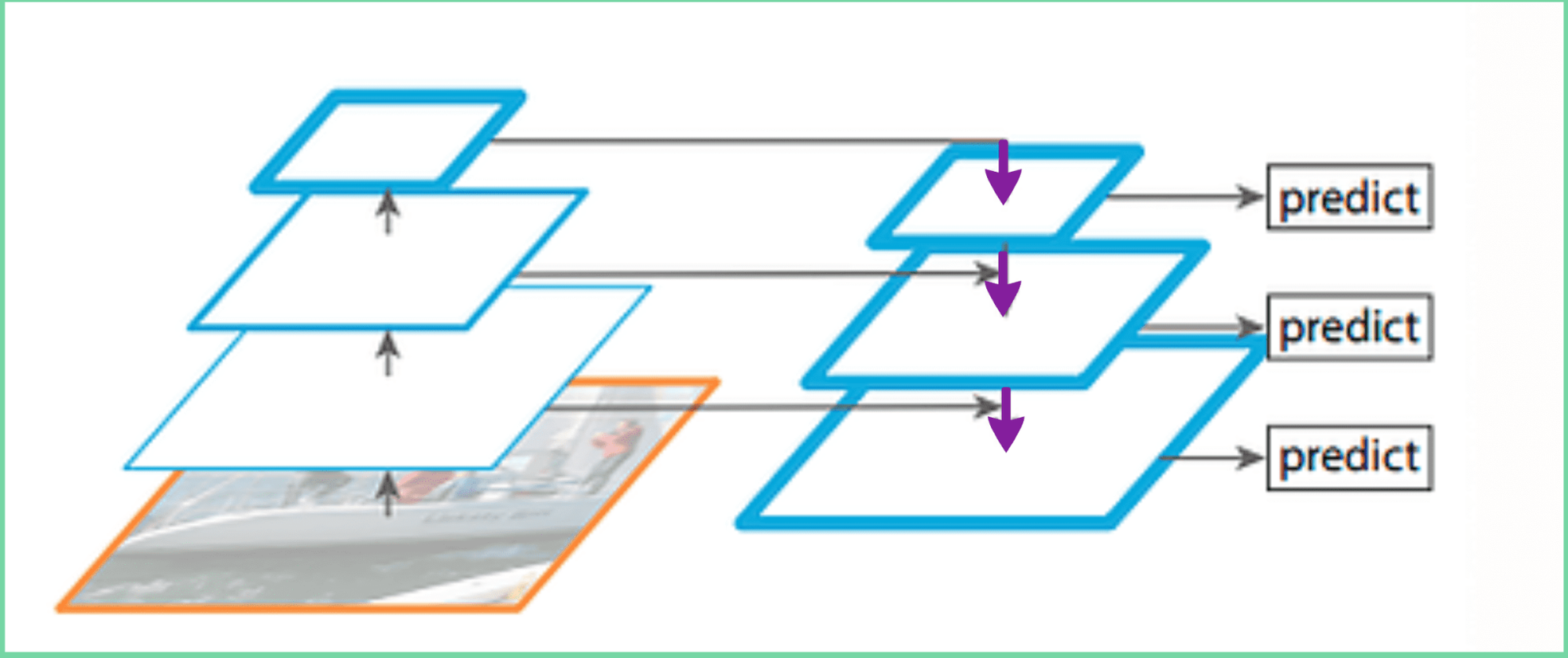
But as the network goes deeper, its feature map gets less fine-grained and could loose ability to catch small details!

YOLO thus outputs instead of a 1D feature map...

...a 3D feature map able to find "large", "medium" and "small" object



Feature pyramid network



Concatenation

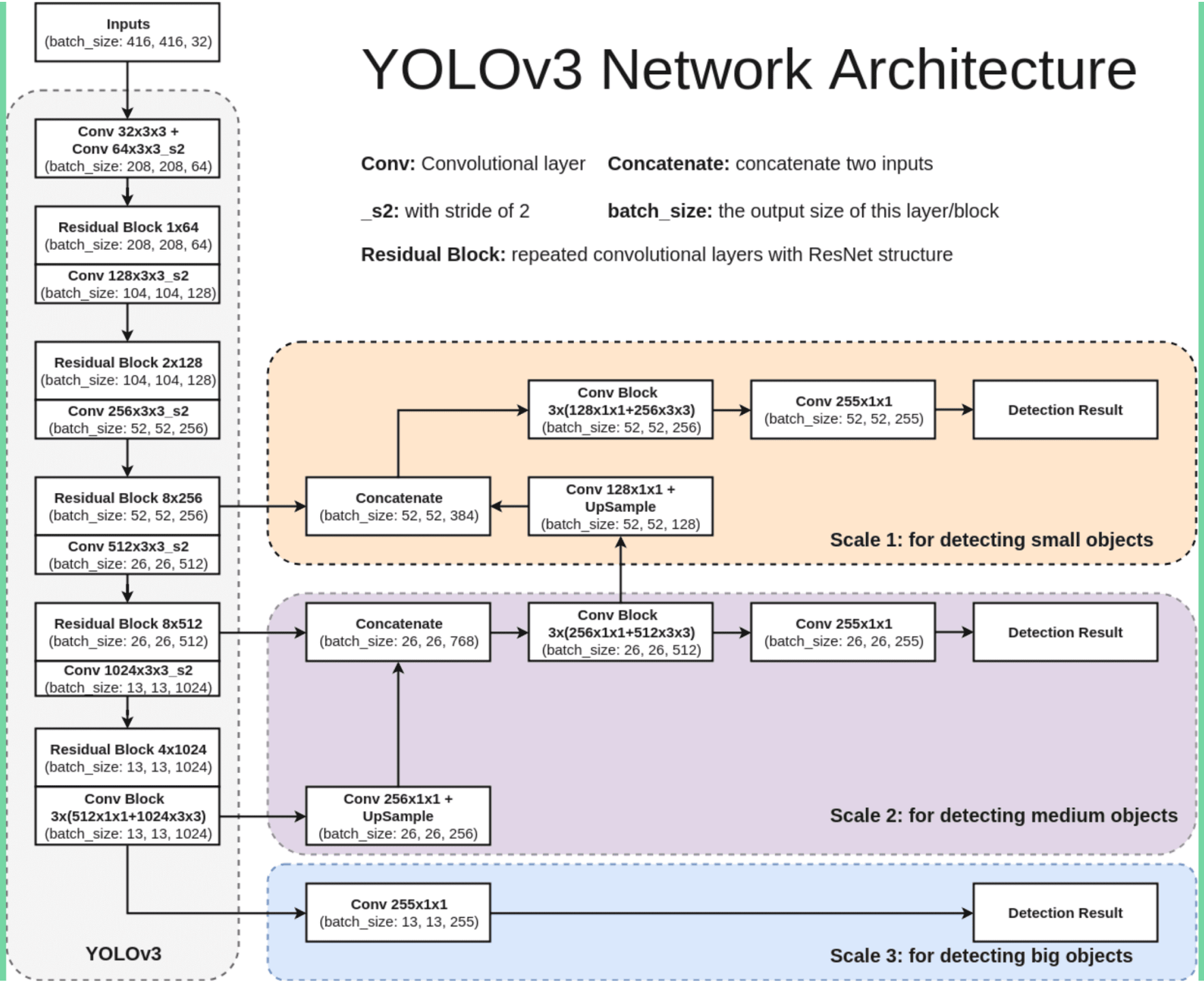
Upsampling

YOLOv3 Network Architecture

Conv: Convolutional layer **Concatenate:** concatenate two inputs

_s2: with stride of 2 **batch_size:** the output size of this layer/block

Residual Block: repeated convolutional layers with ResNet structure

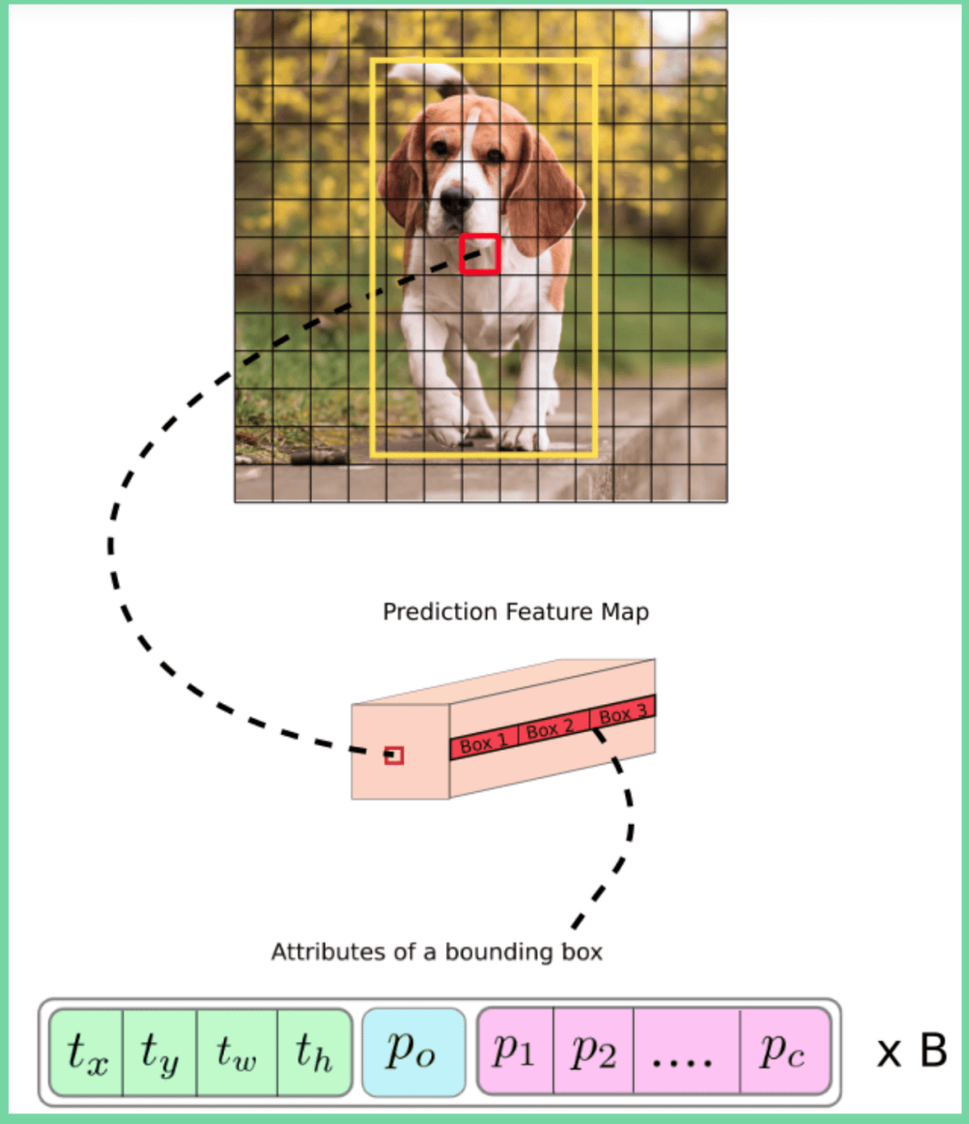


Inference

Given an input image, for each object detected, YOLOv3 outputs:

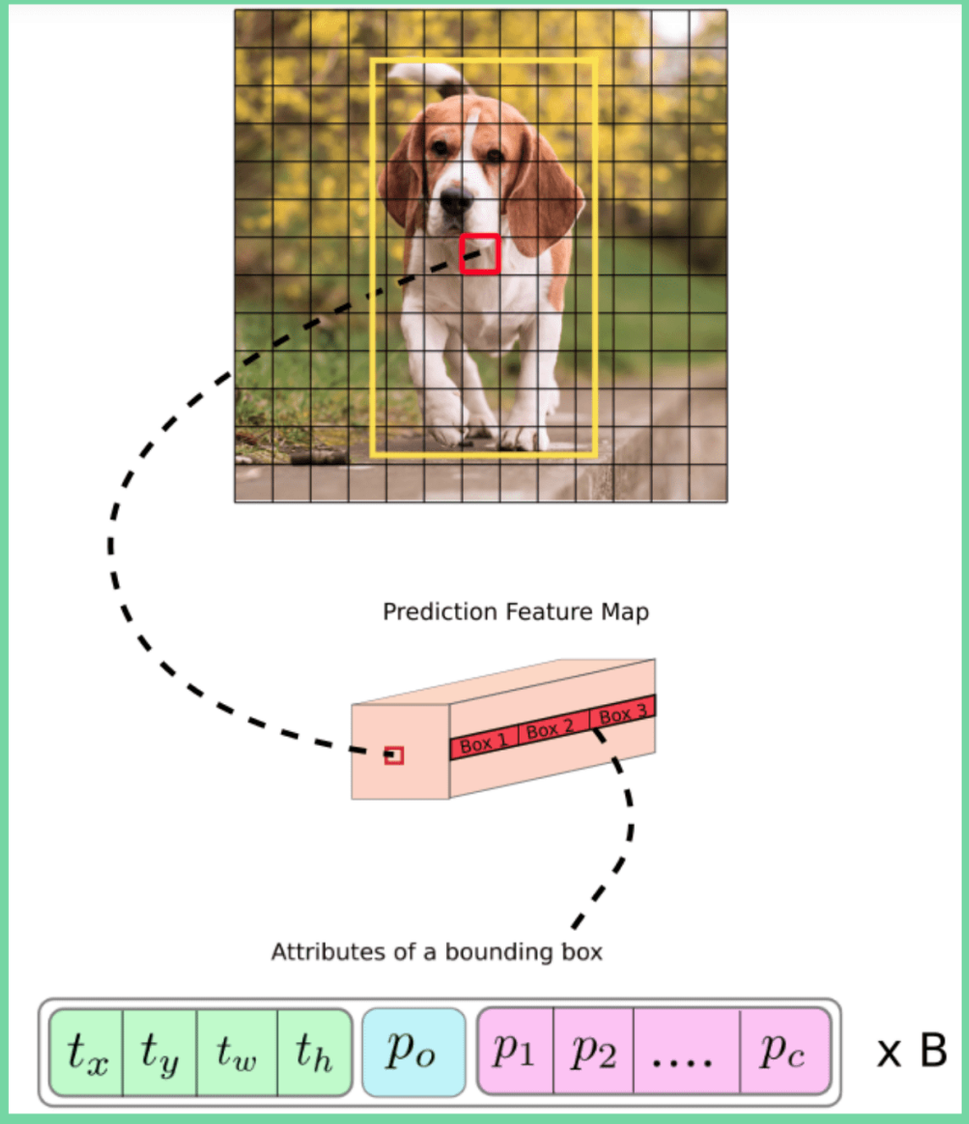
- **One bounding box to identify location**
- **Label of the object**
- **Class probability**

Generation of the input



Each entry of each feature map predicts 3 bounding boxes, with 5+C attributes : center coordinates, dimensions, "objectness score" (by sigmoid) and C, which is the set of class probabilities (by sigmoid).

Generation of the input



The 3 bounding boxes used for each scale are default type, called anchors

Generation of the input (algorithm)

For each anchor of each grid cell, compute probability of contain every class

Select the maximum among all classes and across 3 anchor boxes

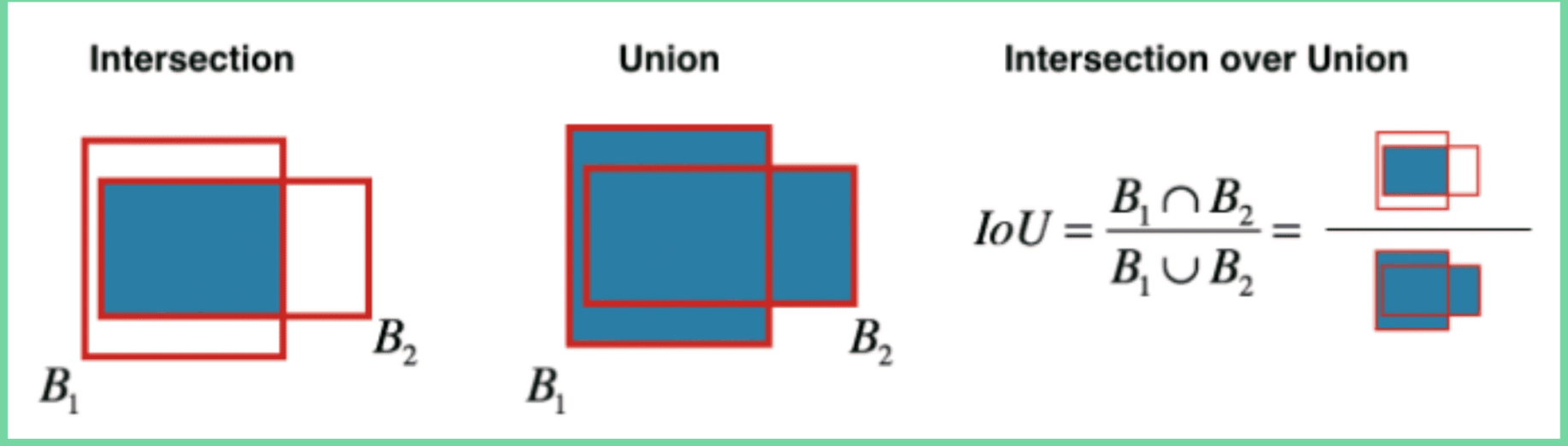
Remove boxes with an objectness score under 0.5

Non-Maximum Suppression

Return optimal boxes to detect all the objects

Non-maximal suppression with IoU

Suppresse overlapping boxes using Intersection over Unit metric



Remove boxes with an IoU > 0.45

Crop features

Tagged objects are then cut out from the image into crops

For IR purposes, these crops need to be described

Crop Descriptors

- Original image ID
- Label
- Importance score
- Feature extracted with Transfer Learning

Feature Extraction

VGG16

(cut before classifier)

YOLOv3

(cut before detection layer)

SIFT

Audio Model

Data Format:

- 2 seconds
- 44100 Hz
- Normalized

981 samples - half with background noises

- 1 multi-classification
- 3 binary

Multi-classification for 'sì', 'no' and 'forse' and the 3 binary for the speaking user

1st Model

2nd Model

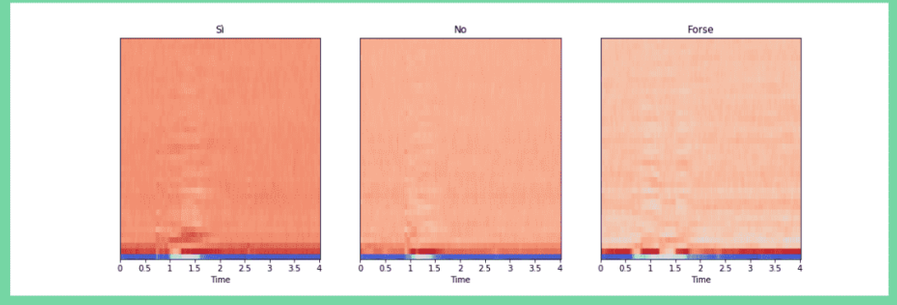
Combine

1st Model

Features Extracted mean of :

- MFCCS

Derives from a inverse fourier transformation of the log frequency



Model Specs

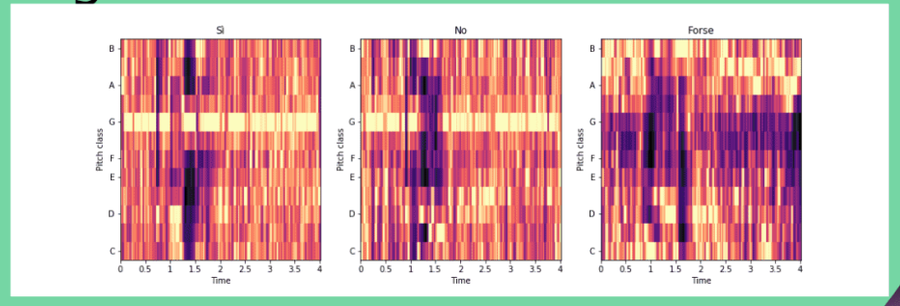
1st Model

Features Extracted mean of :

- MFCCS
- Chroma (stft)

  It indicates the pitch class of the audio (musical notes)

  Might be useful in user identification



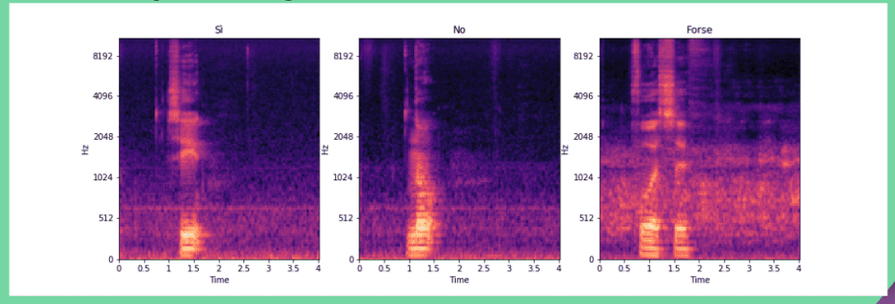
Model Specs

1st Model

Features Extracted mean of :

- MFCCS
- Chroma (stft)
- Melspectrogram

  A spectogram created from the Mel Frequency (same as MFCCS).



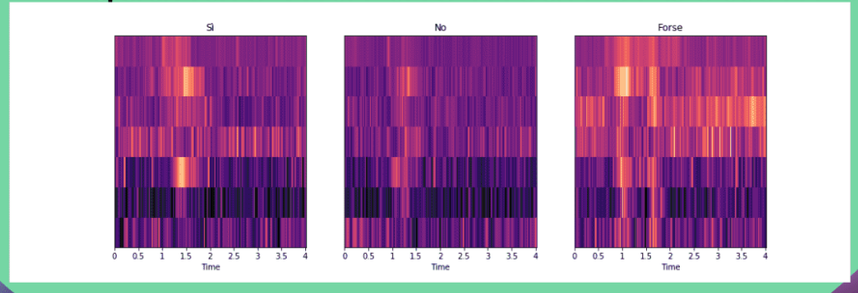
Model Specs

1st Model

Features Extracted mean of :

- MFCCS
- Chroma (stft)
- Melspectrogram
- Contrast (stft)

  Energy contrast for each band of the spectogram (ratio between mean of top quantile with the mean of bottom one)



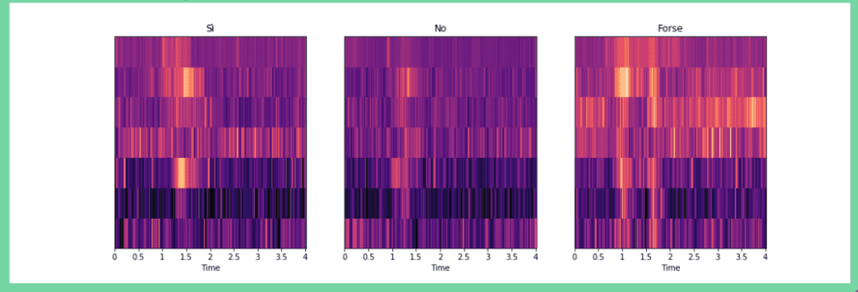
Model Specs

1st Model

Features Extracted mean of :

- MFCCS
- Chroma (stft)
- Melspectrogram
- Contrast (stft)
- Tonnetz

  Computes some tonal features



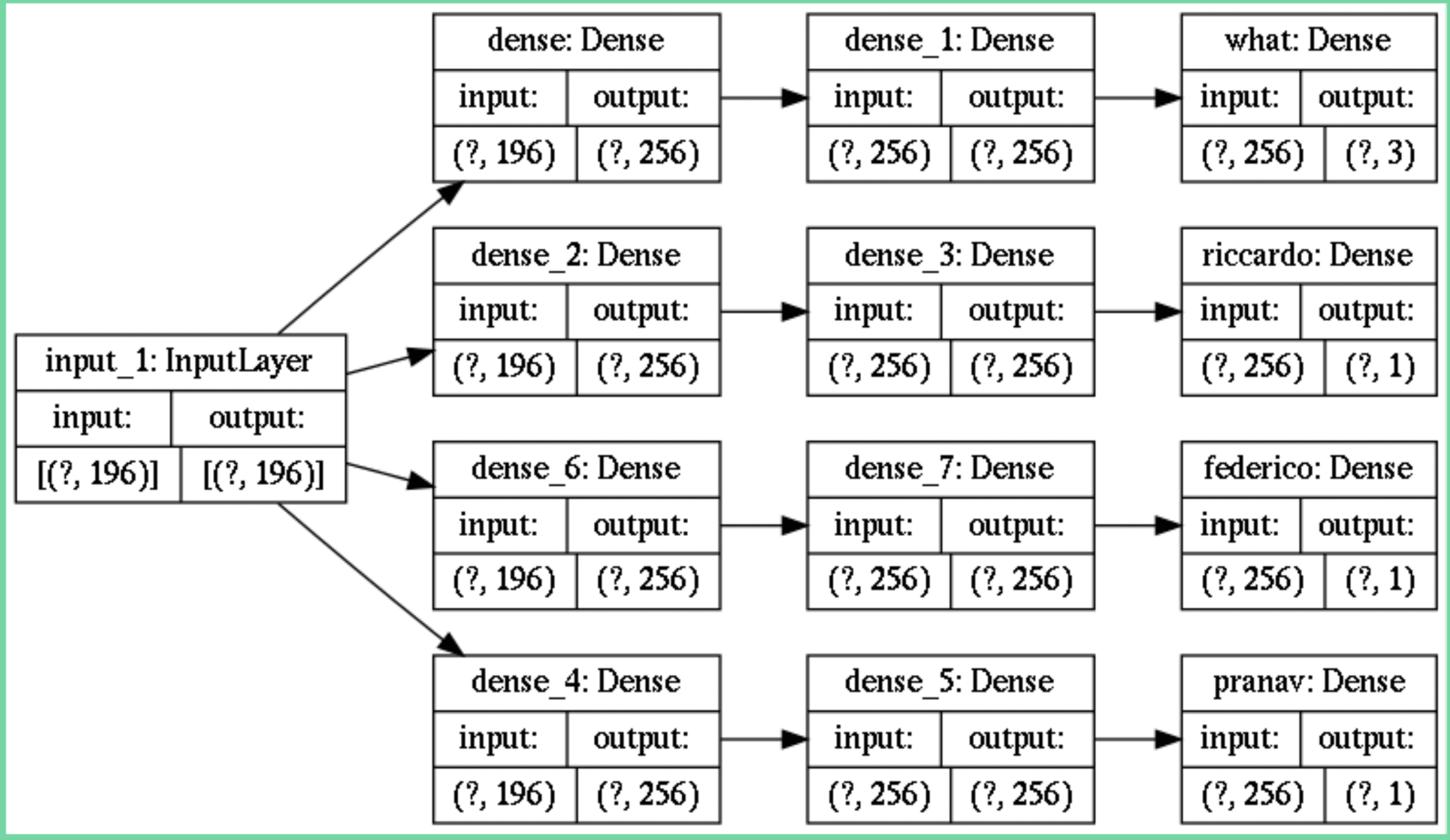
Model Specs

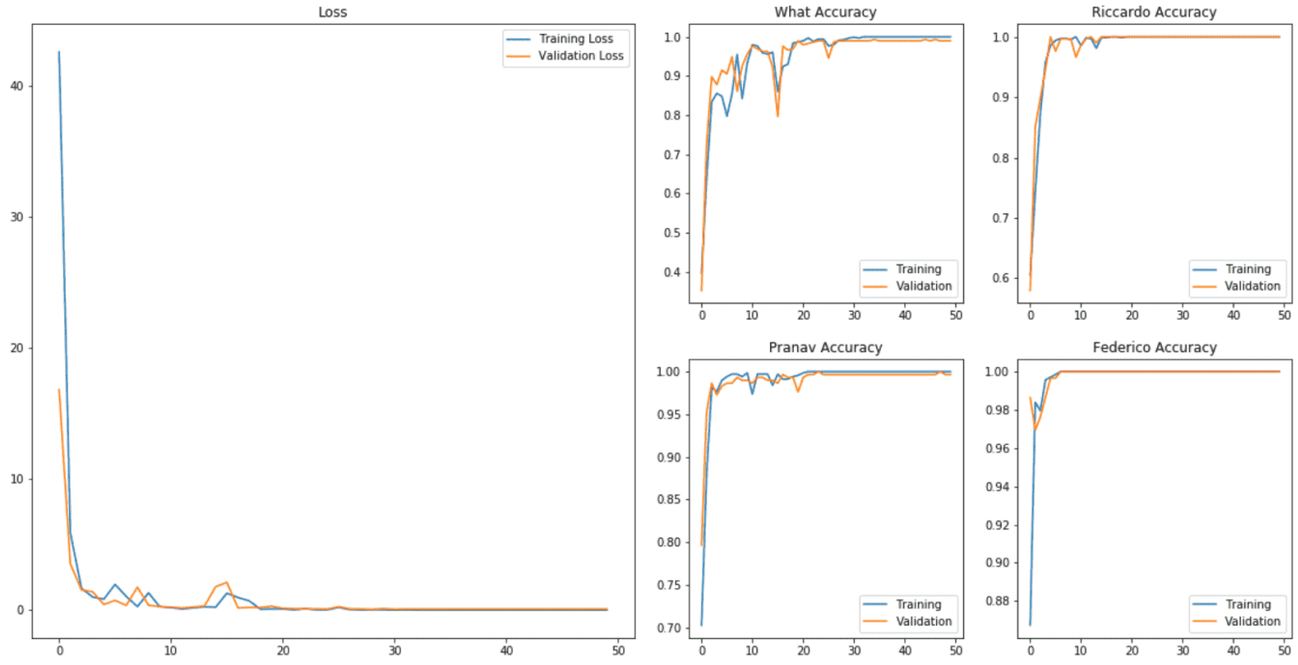
1st Model

Features Extracted mean of :

- MFCCS
- Chroma (stft)
- Melspectrogram
- Contrast (stft)
- Tonnetz
- energy
- standard deviation
- average

Model Specs


input_1: InputLayer
input: [(?, 196)]
output: [(?, 196)]
dense: Dense
input: (?, 196)
output: (?, 256)
dense_1: Dense
input: (?, 256)
output: (?, 256)
what: Dense
input: (?, 256)
output: (?, 3)
dense_2: Dense
input: (?, 196)
output: (?, 256)
dense_3: Dense
input: (?, 256)
output: (?, 256)
riccardo: Dense
input: (?, 256)
output: (?, 1)
dense_6: Dense
input: (?, 196)
output: (?, 256)
dense_7: Dense
input: (?, 256)
output: (?, 256)
federico: Dense
input: (?, 256)
output: (?, 1)
dense_4: Dense
input: (?, 196)
output: (?, 256)
dense_5: Dense
input: (?, 256)
output: (?, 256)
pranav: Dense
input: (?, 256)
output: (?, 1)

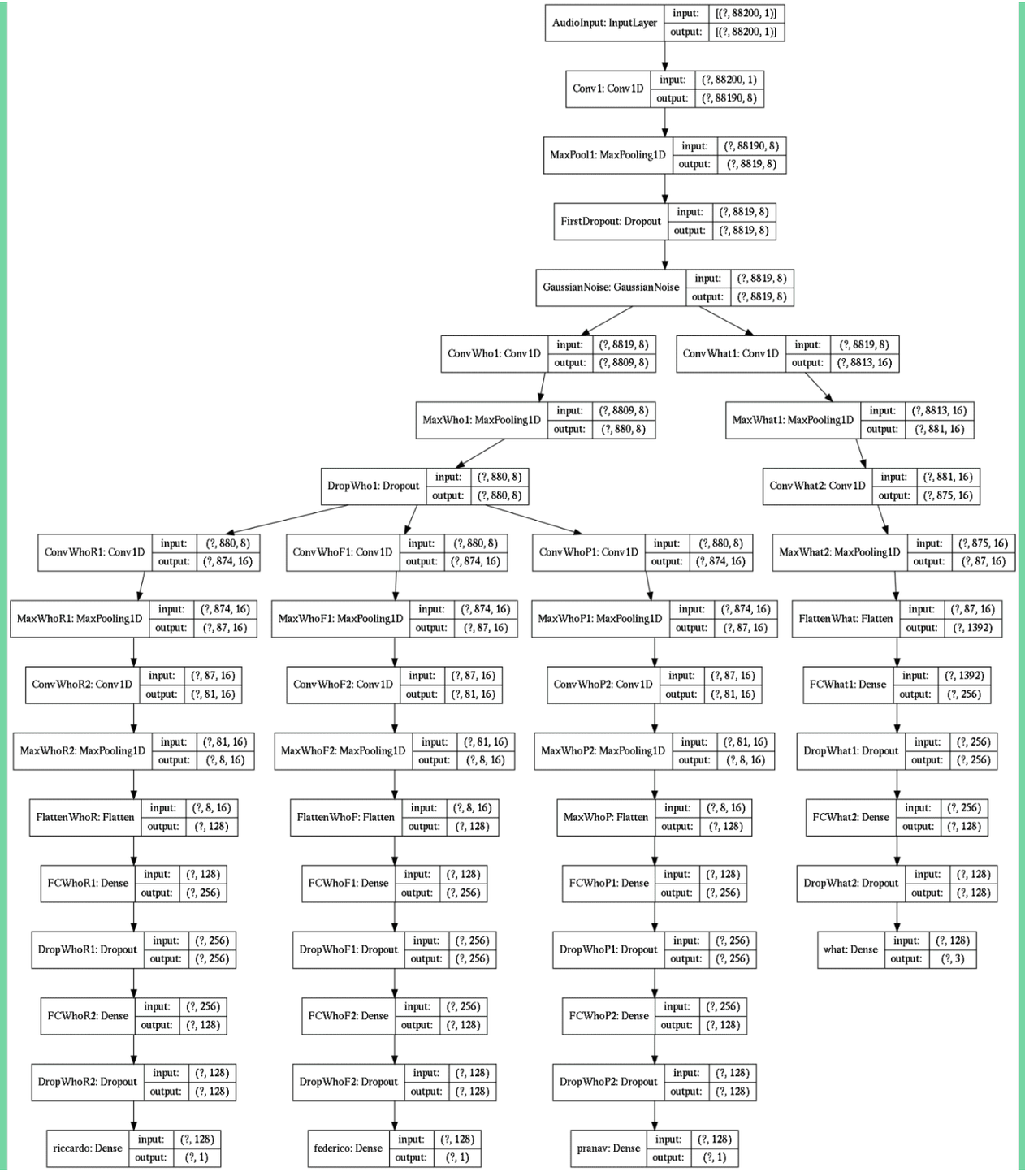


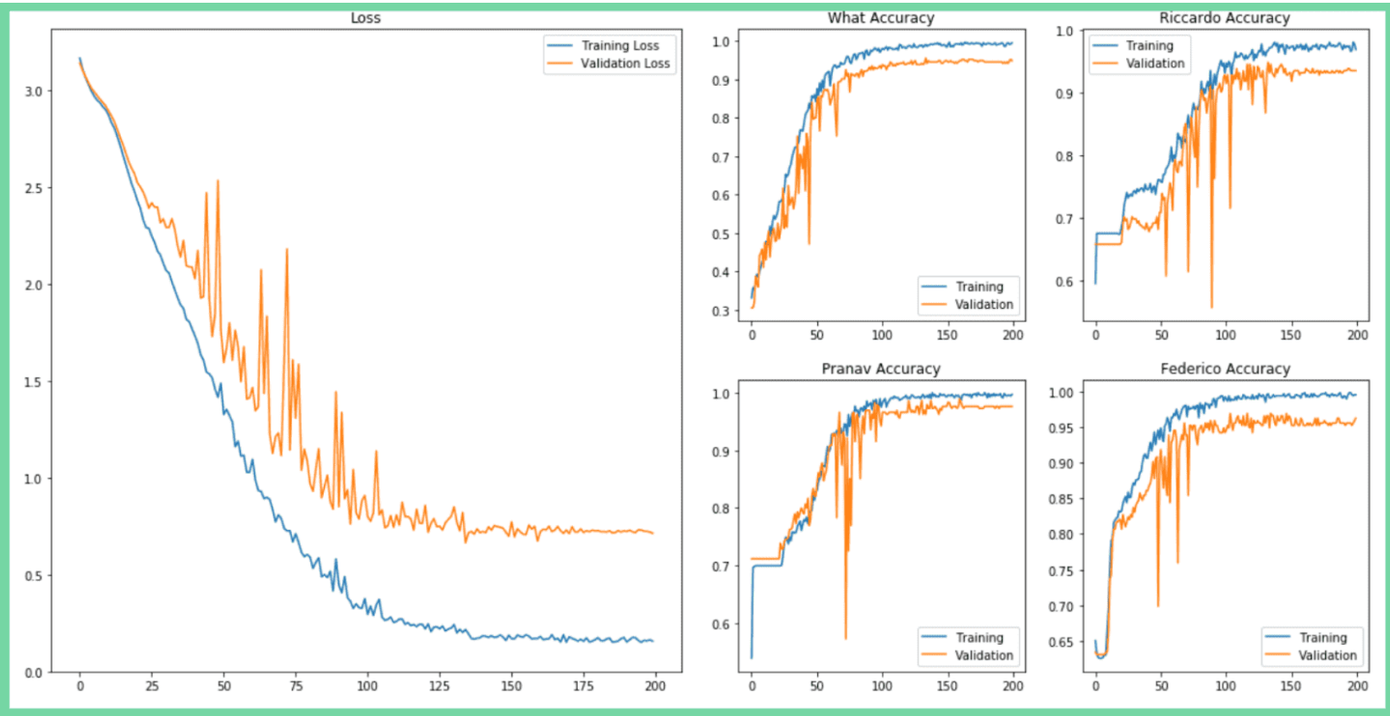
Very accurate when the speaker is one of us but doesn't generalize well to a new speaker!!!

2nd Model

- Way more complex as a model, uses the convolutions to extract features
- Input consists of 88200 neurons
- Gaussian noise to reduce overfitting
- Combines the losses adding them
- Uses two callbacks:
    - ReduceLROnPlateau (patience 5)
    - EarlyStopping (patience 100)
- A lot of shared features between models

Model Specs


AudioInput: InputLayer
input: [(?, 88200, 1)]
output: [(?, 88200, 1)]
Conv1: Conv1D
input: (?, 88200, 1)
output: (?, 88190, 8)
MaxPool1: MaxPooling1D
input: (?, 88190, 8)
output: (?, 8819, 8)
FirstDropout: Dropout
input: (?, 8819, 8)
output: (?, 8819, 8)
GaussianNoise: GaussianNoise
input: (?, 8819, 8)
output: (?, 8819, 8)
ConvWho1: Conv1D
input: (?, 8819, 8)
output: (?, 8809, 8)
ConvWhat1: Conv1D
input: (?, 8819, 8)
output: (?, 8813, 16)
MaxWho1: MaxPooling1D
input: (?, 8809, 8)
output: (?, 880, 8)
MaxWhat1: MaxPooling1D
input: (?, 8813, 16)
output: (?, 881, 16)
DropWho1: Dropout
input: (?, 880, 8)
output: (?, 880, 8)
ConvWhat2: Conv1D
input: (?, 881, 16)
output: (?, 875, 16)
ConvWhoR1: Conv1D
input: (?, 880, 8)
output: (?, 874, 16)
ConvWhoF1: Conv1D
input: (?, 880, 8)
output: (?, 874, 16)
ConvWhoP1: Conv1D
input: (?, 880, 8)
output: (?, 874, 16)
MaxWhat2: MaxPooling1D
input: (?, 875, 16)
output: (?, 87, 16)
MaxWhoR1: MaxPooling1D
input: (?, 874, 16)
output: (?, 87, 16)
MaxWhoF1: MaxPooling1D
input: (?, 874, 16)
output: (?, 87, 16)
MaxWhoP1: MaxPooling1D
input: (?, 874, 16)
output: (?, 87, 16)
FlattenWhat: Flatten
input: (?, 87, 16)
output: (?, 1392)
ConvWhoR2: Conv1D
input: (?, 87, 16)
output: (?, 81, 16)
ConvWhoF2: Conv1D
input: (?, 87, 16)
output: (?, 81, 16)
ConvWhoP2: Conv1D
input: (?, 87, 16)
output: (?, 81, 16)
FCWhat1: Dense
input: (?, 1392)
output: (?, 256)
MaxWhoR2: MaxPooling1D
input: (?, 81, 16)
output: (?, 8, 16)
MaxWhoF2: MaxPooling1D
input: (?, 81, 16)
output: (?, 8, 16)
MaxWhoP2: MaxPooling1D
input: (?, 81, 16)
output: (?, 8, 16)
DropWhat1: Dropout
input: (?, 256)
output: (?, 256)
FlattenWhoR: Flatten
input: (?, 8, 16)
output: (?, 128)
FlattenWhoF: Flatten
input: (?, 8, 16)
output: (?, 128)
MaxWhoP: Flatten
input: (?, 8, 16)
output: (?, 128)
FCWhat2: Dense
input: (?, 256)
output: (?, 128)
FCWhoR1: Dense
input: (?, 128)
output: (?, 256)
FCWhoF1: Dense
input: (?, 128)
output: (?, 256)
FCWhoP1: Dense
input: (?, 128)
output: (?, 256)
DropWhat2: Dropout
input: (?, 128)
output: (?, 128)
DropWhoR1: Dropout
input: (?, 256)
output: (?, 256)
DropWhoF1: Dropout
input: (?, 256)
output: (?, 256)
DropWhoP1: Dropout
input: (?, 256)
output: (?, 256)
what: Dense
input: (?, 128)
output: (?, 3)
FCWhoR2: Dense
input: (?, 256)
output: (?, 128)
FCWhoF2: Dense
input: (?, 256)
output: (?, 128)
FCWhoP2: Dense
input: (?, 256)
output: (?, 128)
DropWhoR2: Dropout
input: (?, 128)
output: (?, 128)
DropWhoF2: Dropout
input: (?, 128)
output: (?, 128)
DropWhoP2: Dropout
input: (?, 128)
output: (?, 128)
riccardo: Dense
input: (?, 128)
output: (?, 1)
federico: Dense
input: (?, 128)
output: (?, 1)
pranav: Dense
input: (?, 128)
output: (?, 1)



Way more noisy training, and seem to over fit the training data, but achieves good performance.

No AutoML :(

So many parameters to optimize.

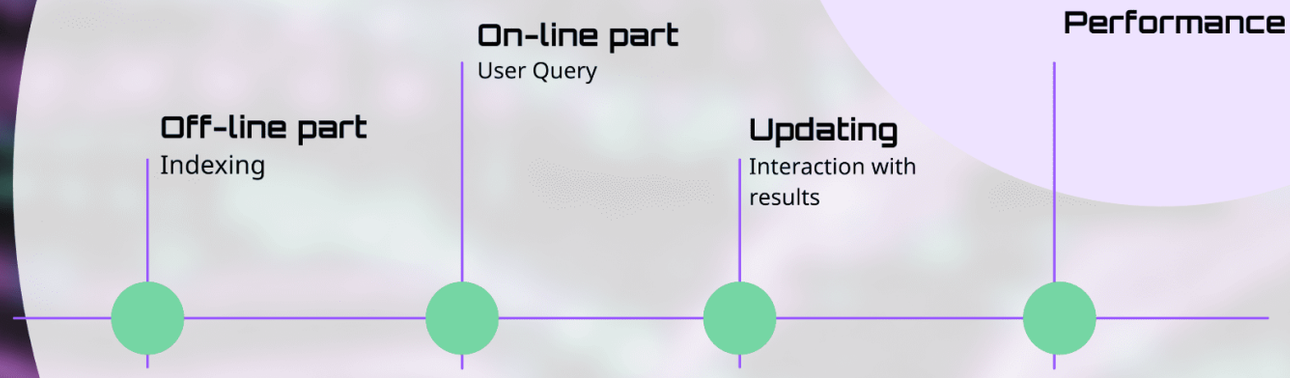
Combine the results

- First model is very accurate
- Second model is a little better at generalizing

Best choice seems to use both of them, so we take the average of both predictions and if the score is bigger than a fixed threshold it is accepted and rejected and classified as external class otherwise.

$$\frac{2 * \text{CNN model} + 1 * \text{No CNN model}}{3}$$

Information Retrieval



Off-line part: indexing

Crop Extraction
(with label)

Feature Extraction
from the Crop

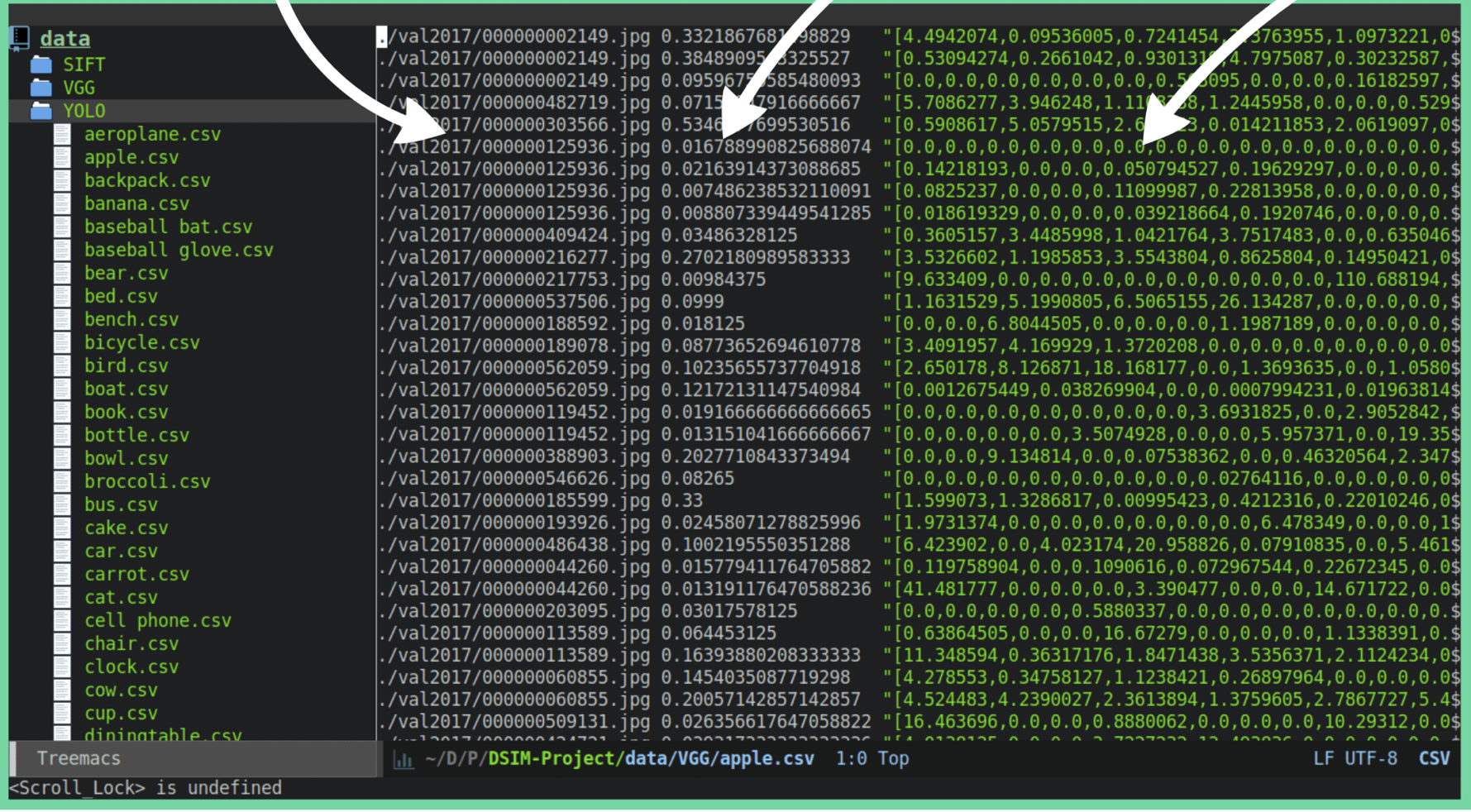
Storing vectors and
Crop Size on HD

Off-line part: indexing

file

crop size

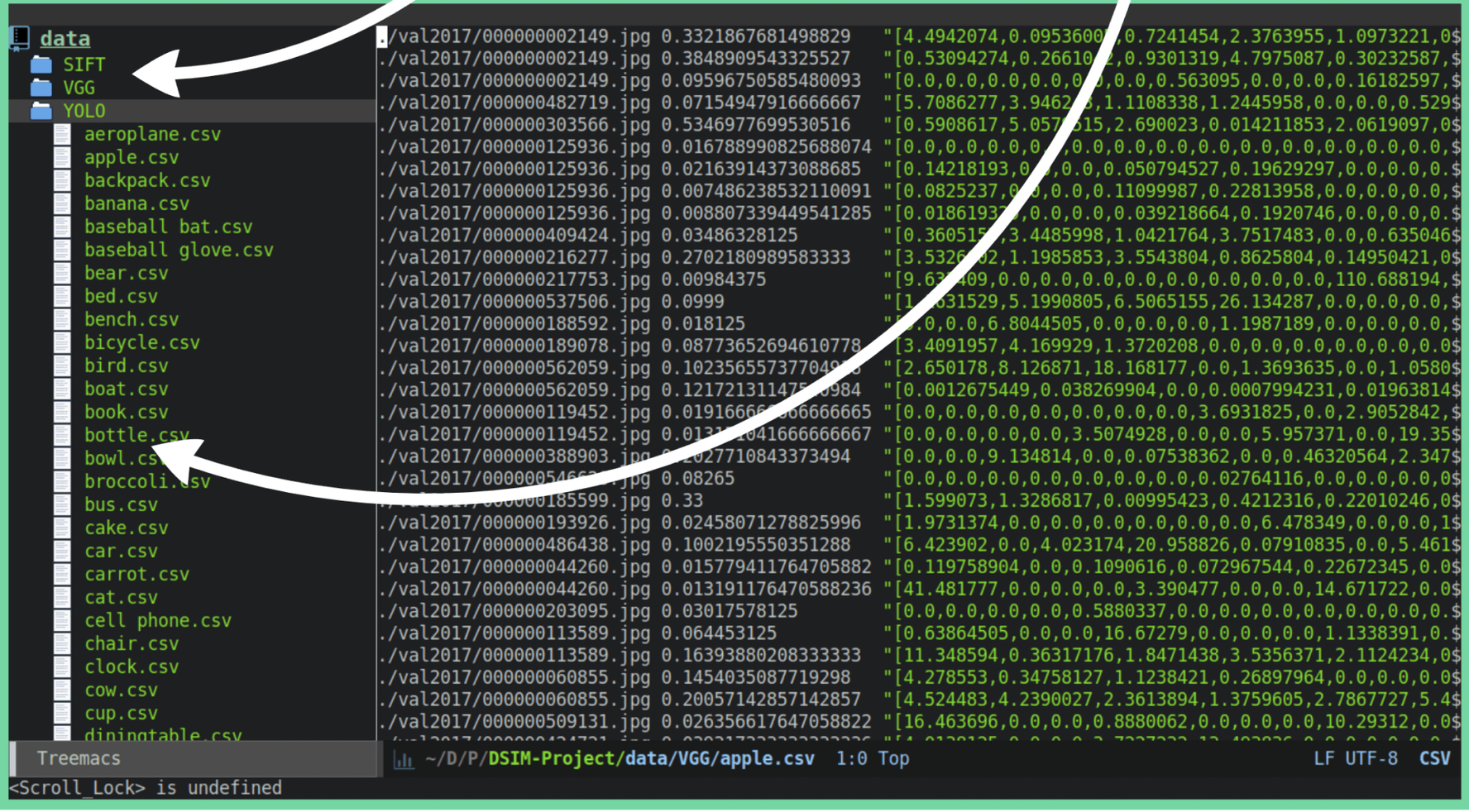
features



Off-line part: indexing

feature extractor

tag



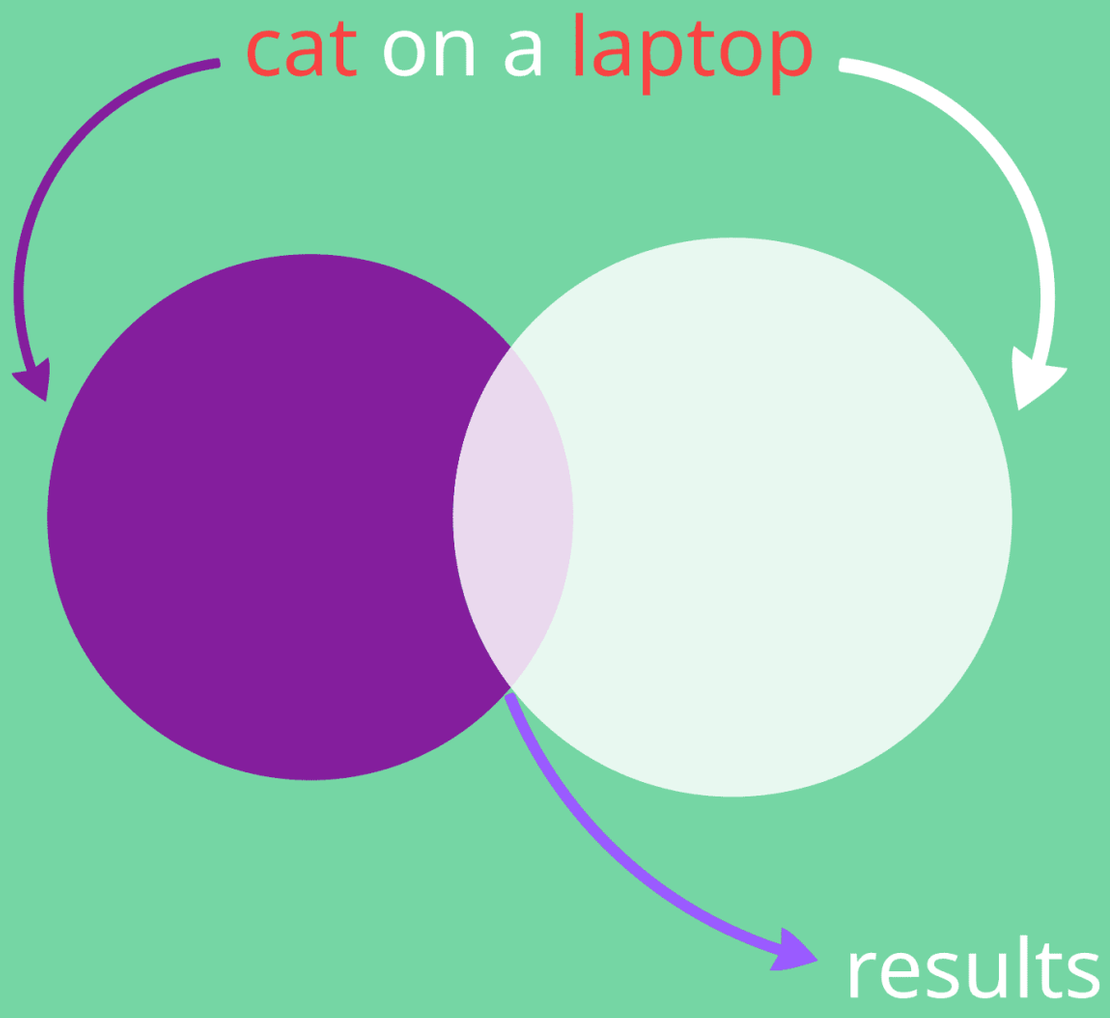
On-line Part: Query

Boolean Matching
of the Text

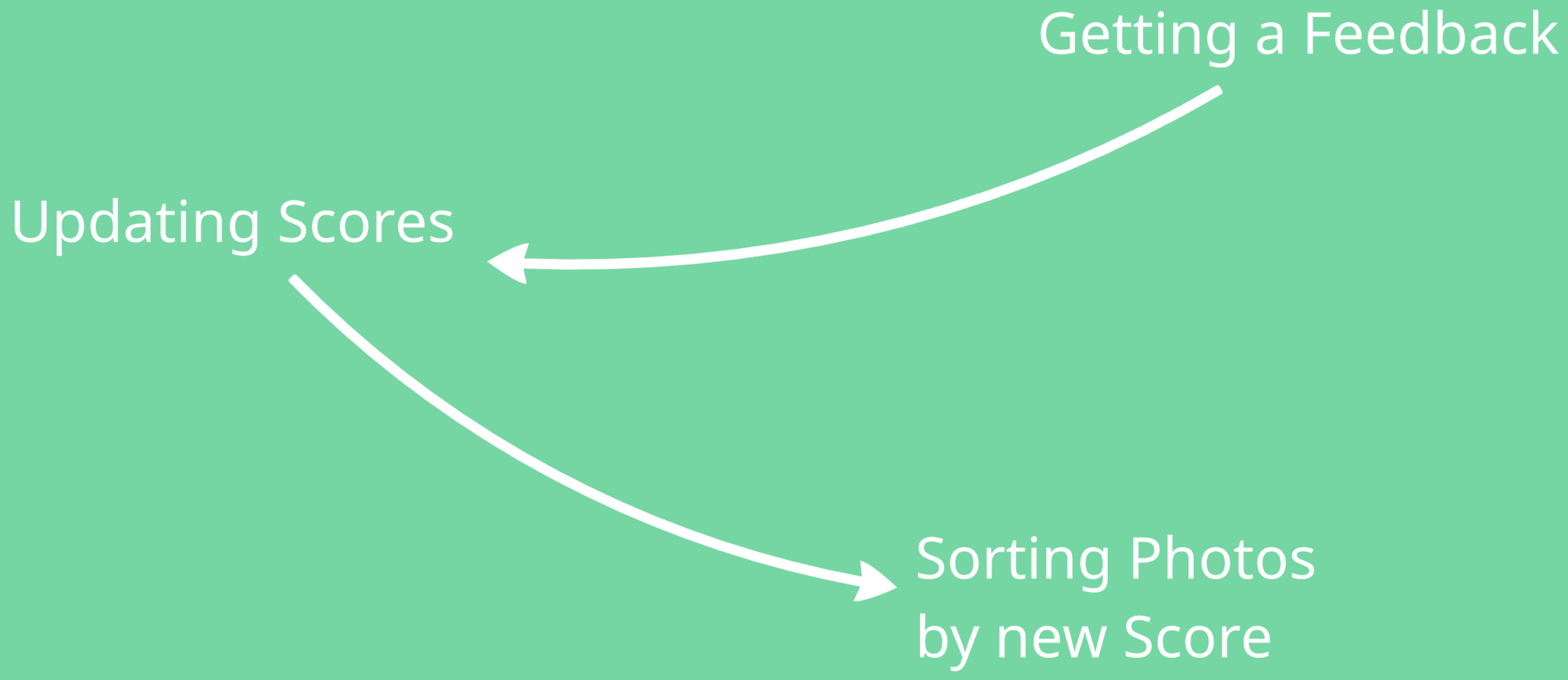
Loading vectors

Sorting Photos
by Crop Size

On-line Part: Query



Interaction: Updating Scores



Interaction: Updating Scores

Fist matches between crops are found
(if I select a cat, I want similar cats)

Then the crop score
is updated

$selected \leftarrow$ selected photo

$w \leftarrow$ weight of the update

$\forall$ photo

$\forall$ crop $\in$ photo

$$dist_{crop,selected} \leftarrow \min_{crop_{selected}} \{dist(crop, crop_{selected})\}$$

$$score_{crop} \leftarrow (1 - w) \cdot score_{crop} + w \cdot dist_{crop,selected}$$

$$score_{photo} \leftarrow mean(score_{crop} \forall \text{ crop } in \text{ photo})$$

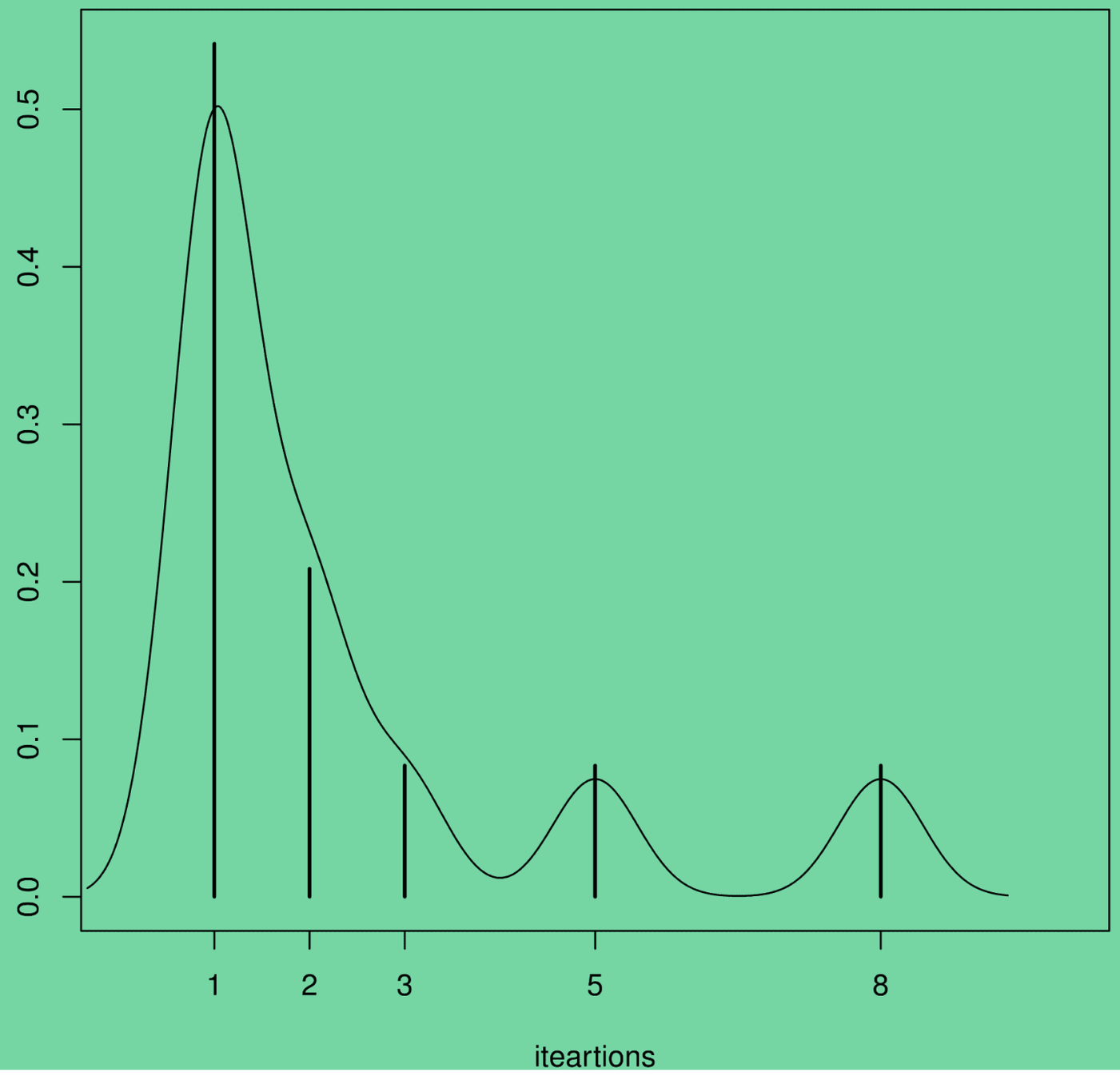
Total score for the photo is
the mean of its crop scores

ALL DISTANCES ARE NORMALIZED

Performance

Efficiency: fast and scalable
O(n)

Effectiveness: most results (> 75%) are found in < 3 iterations


Results found using our Search Engine
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
5
8
iteartions