Segmentation is all you need?

DEBORAH DORE

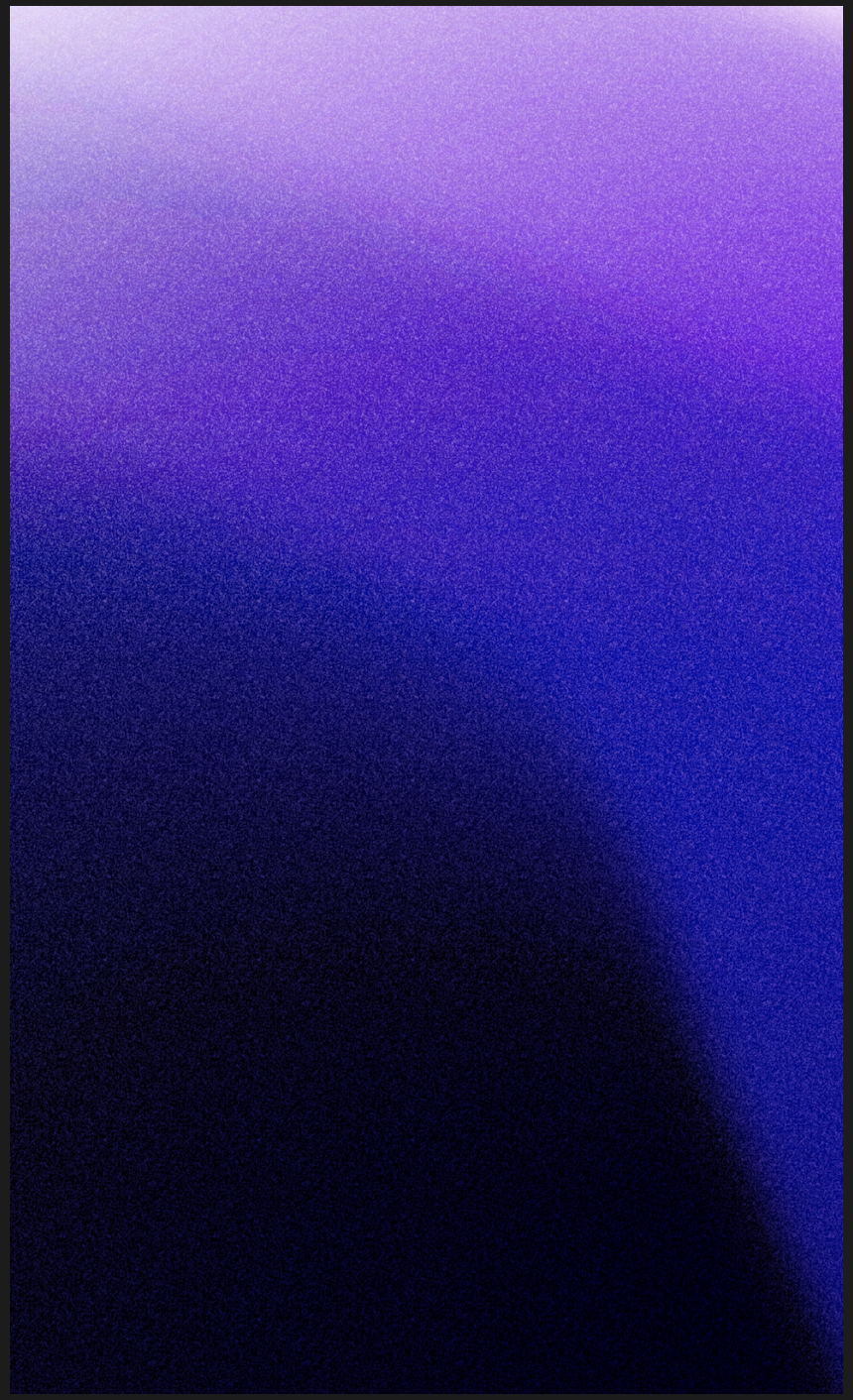


Table of Contents

1

Introduction

Introduction

What does this work deal with?

The problem

Dataset

The goal

Tools

2

Segmentation

Segmentation

WHY IS IT USEFUL?

to simplify an image and reduce complexity

to extract meaningful information that can be exploited

to analyze an image more efficiently reducing time and costs

Segmentation

TECHNIQUES

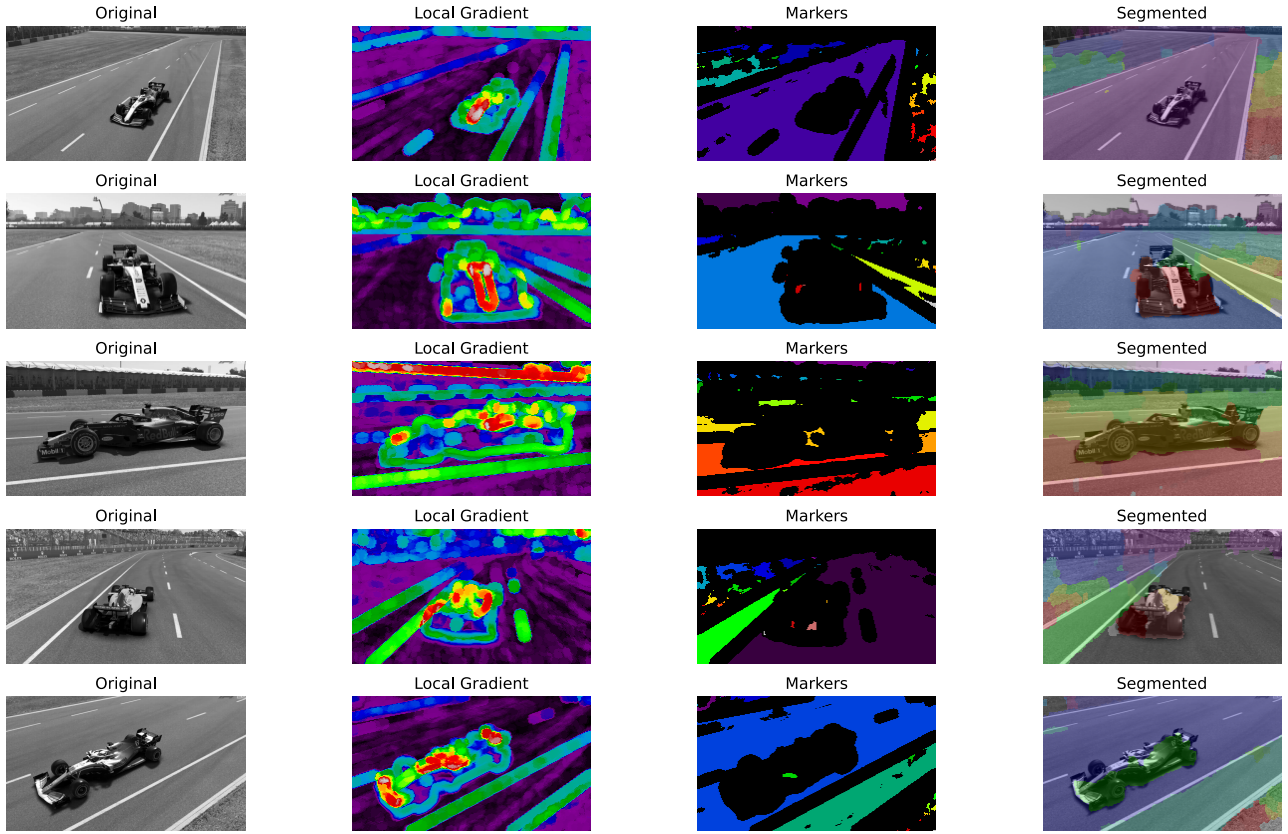
1. Thresholding
2. Watershed
3. K-Means Clustering
4. K-Means Clustering adding pixel's coordinates
5. Mean Shift Clustering

Segmentation: Thresholding

- The simplest type of image segmentation
- Threshold has been found using Otsu's method
- Estimated execution time for one image: ~0.492 seconds

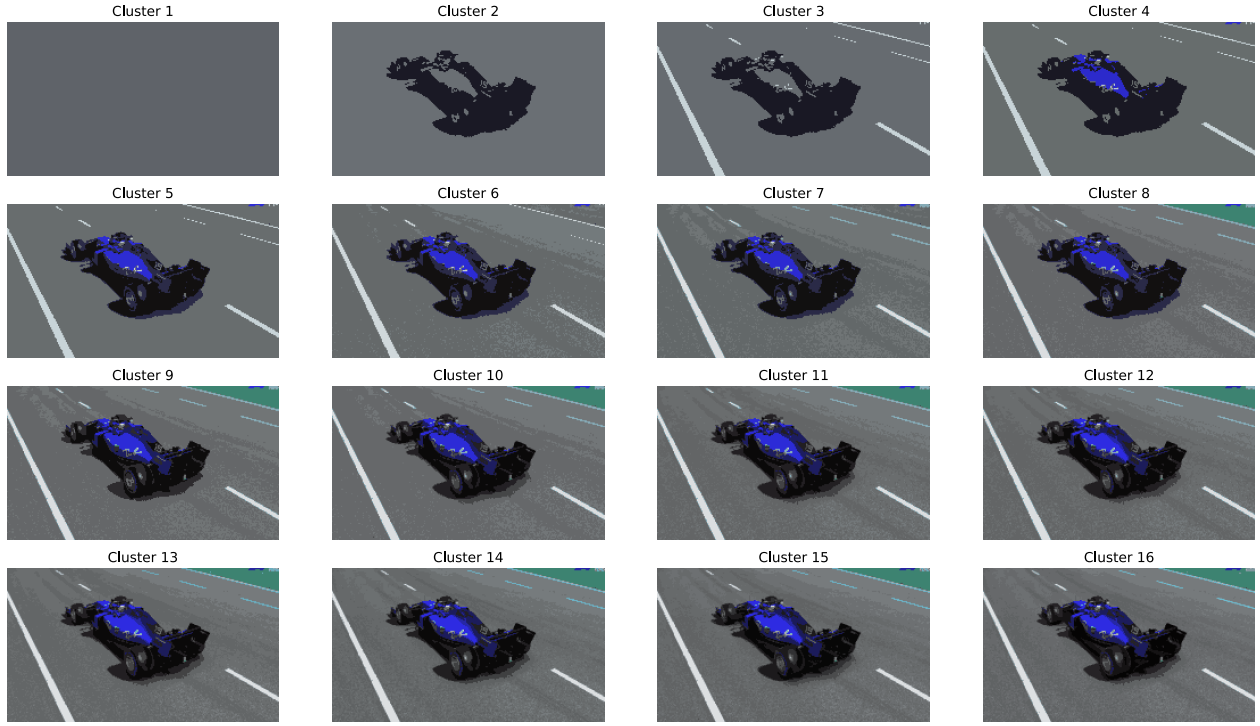
Segmentation: Watershed

- Watershed segmentation is a region-based technique that uses image's morphology
- The watershed transform decomposes an image into catchment basins
- Estimated execution time for one image: ~1.266 seconds



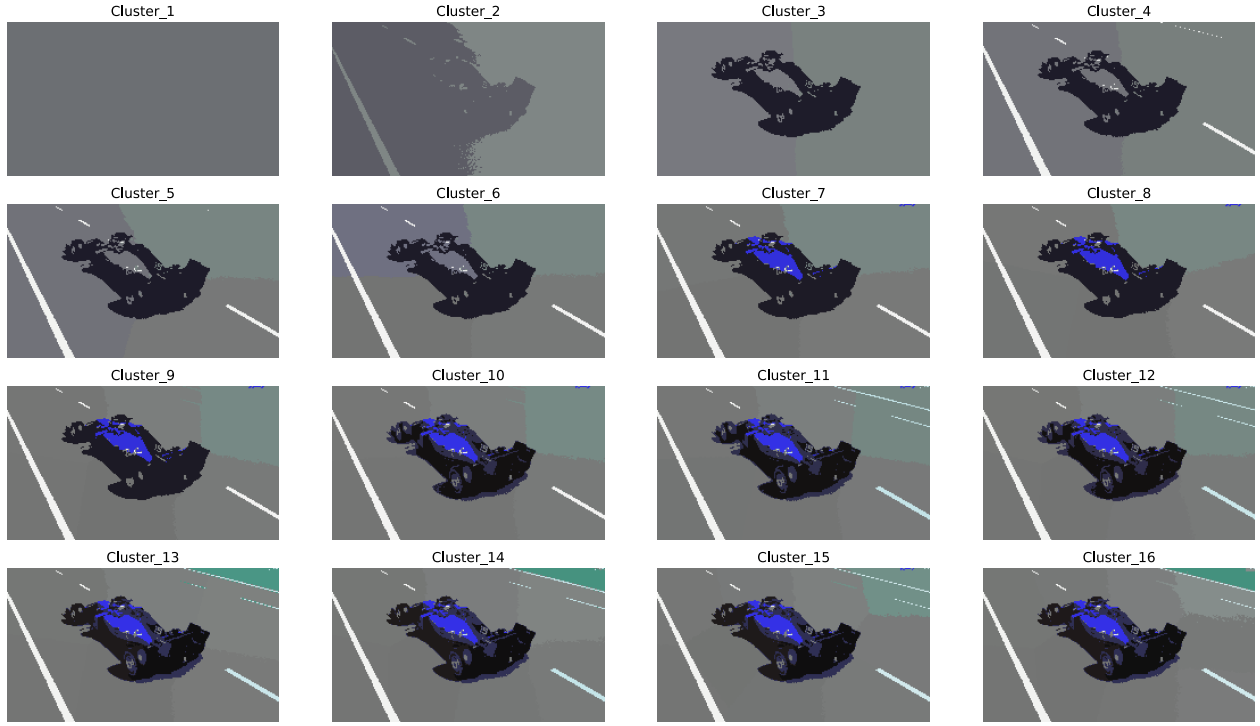
Segmentation: K-Means Clustering

- K-Means clustering is an unsupervised algorithm that can be used to reduce the complexity of the image
- Groups pixel's in n given **clusters** based on a distance functions
- Estimated execution time for one image: ~0.402 seconds (depending on the number of clusters)



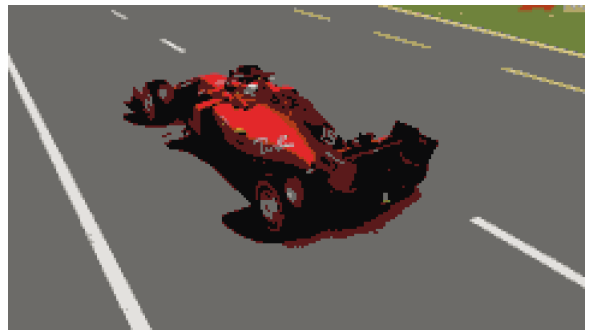
Segmentation: K-Means Clustering adding pixel's coordinates

- An experiment to improve K-Means
- Each pixels also contains the (x,y) coordinates of the matrix
- Pixel that are close in the image have higher probability to be grouped together
- Estimated execution time for one image: ~0.379 seconds (depending on the number of clusters)



Segmentation: Mean Shift Clustering

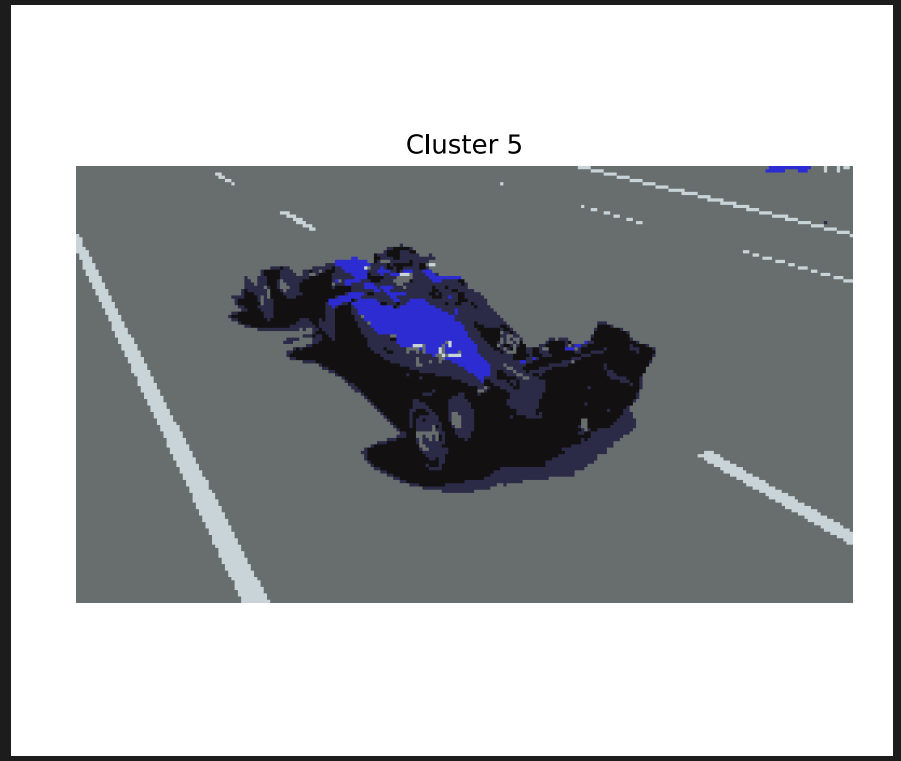
- Unsupervised Learning algorithm that aims to group pixel in n cluster
- The number of clusters is not given in advance
- It discovers blobs in a density of samples
- Computationally Expensive: O(T*n^2) where T is the number of points and n is the number of samples
- Estimated execution time for one image: ~ 2.782 seconds (depending on the given bandwidth)



Segmentation

TECHNIQUES

1. Thresholding
2. Watershed
3. K-Means Clustering
4. K-Means Clustering adding pixel's coordinates
5. Mean Shift Clustering

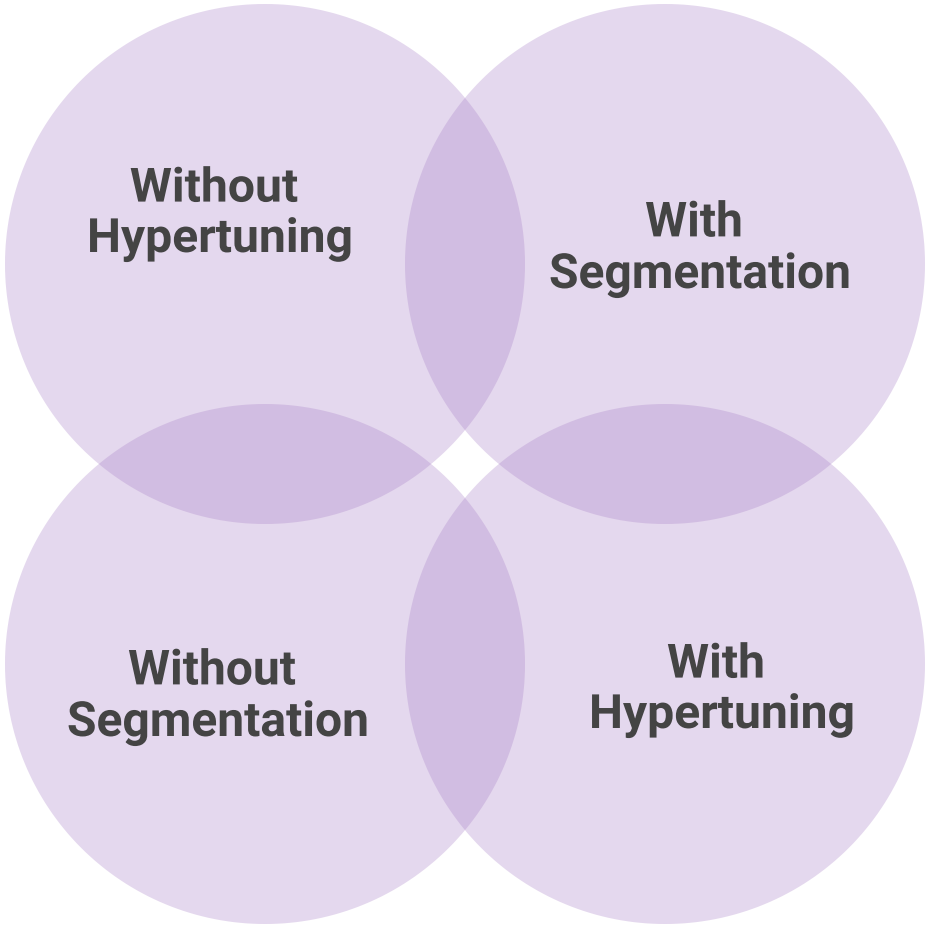


3

Classification

Classification

STRATEGIES



- Open CV 2
- Pytorch Framework
- Ray Tune Framework

Classification

WITHOUT HYPERTUNING

1. Convolutional Neural Network
2. Dropout Layer to avoid over-fitting
3. Simple Data Augmentation
4. Early stopping
5. Weighed Cross Entropy

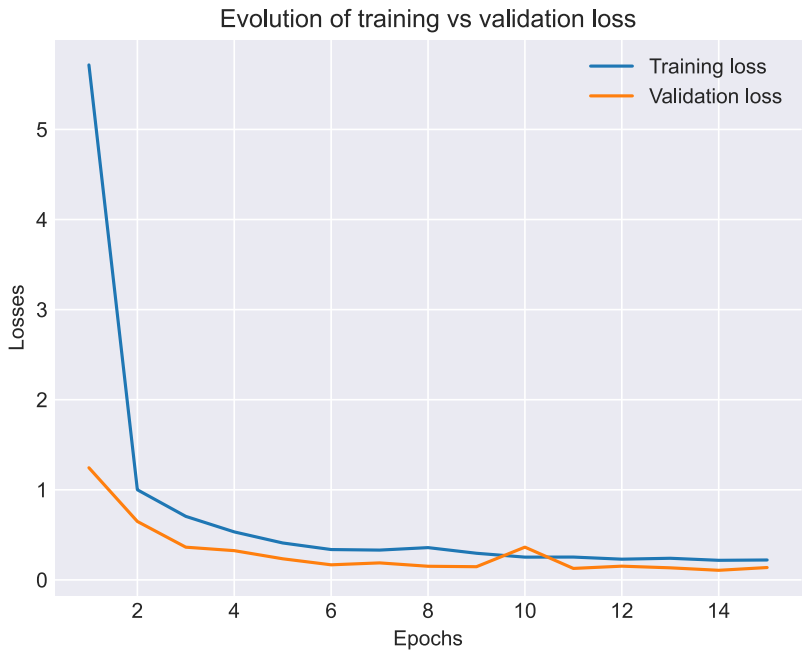
```
ConvNet(
  (layer1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Dropout(p=0.5, inplace=False)
  )
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc): Sequential(
    (0): Linear(in_features=30720, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

Classification

WITHOUT HYPERTUNING

Without Segmentation

- Accuracy: 99%
- Training time: ~52 minutes



With Segmentation

- Accuracy: 95%
- Training time: ~70 minutes

Classification

SEGMENTATION WITH HYPERTUNING

1. Convolutional Neural Network

2. Tuned params:
   - *num of layers*
   - *output of convolutional layers*
   - *output of linear layer*
   - *learning rate*
   - *weight decay*
   - *dropout probability*

```
ConvNet(
  (layer1): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Dropout(p=0.5, inplace=False)
  )
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc): Sequential(
    (0): Linear(in_features=253440, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=10, bias=True)
  )
)
```

Classification

SEGMENTATION WITH HYPERTUNING

Results

- Accuracy: 95%
- Training time: 25 min

Training time reduced from both of the two previous model

Same accuracy of the non-hypertuned model but lower than the model without segmentation

Lower accuracy per class with respect to both previous model

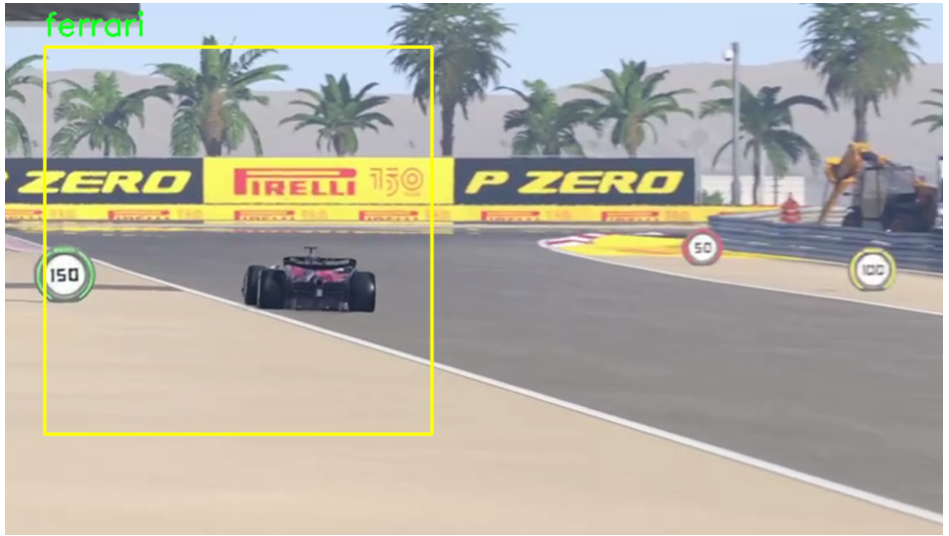
4

Detection

Detection & Annotation

Detect Car using Haar Cascade Classifiers

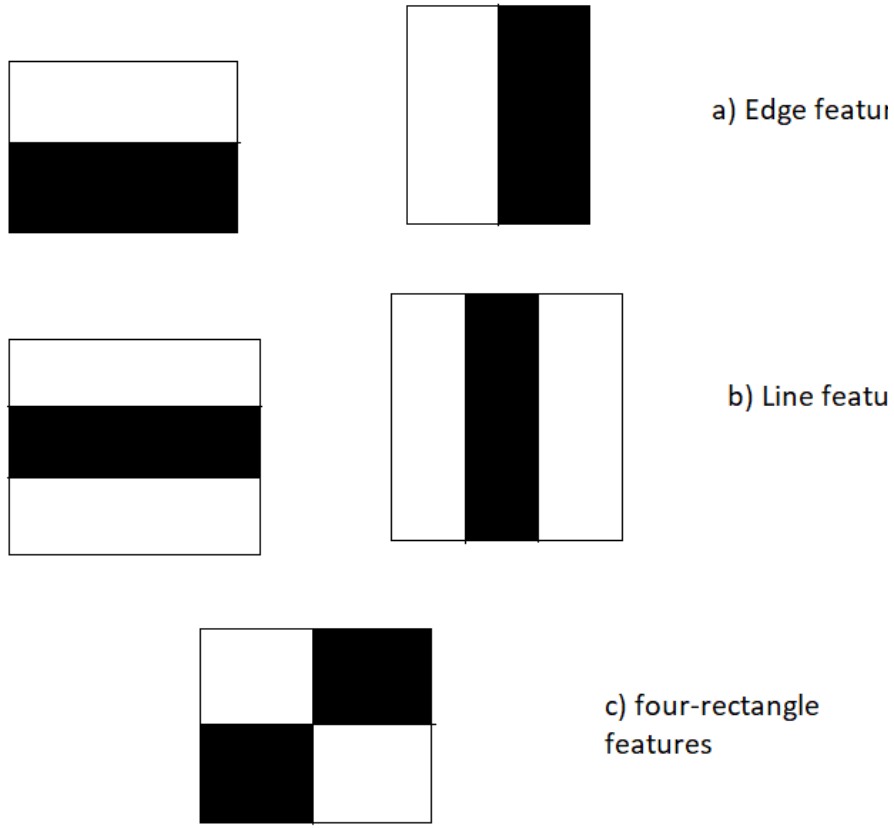
Predict team using model previously trained



Detection & Annotation

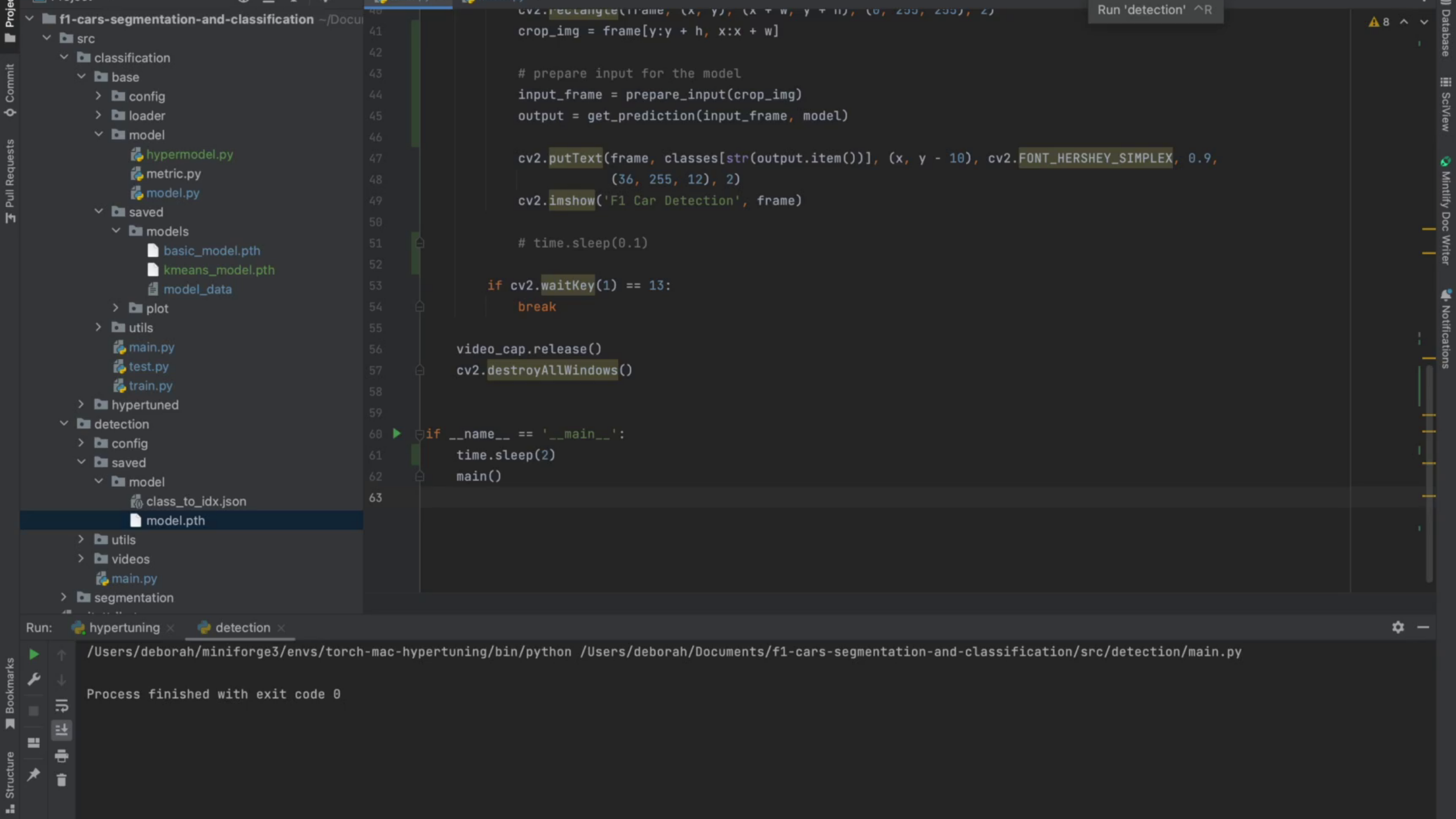
CASCADE CLASSIFIER

- Simple Machine Learning based approach
- Trained on positive and negative examples
- It works as a sliding windows through the image, for every window extracts Haar Features
- Haar Features are computed by subtracting the sum of the intensity of the pixels in the white region and the sum of the intensity of the pixels in the black region



5

Demo

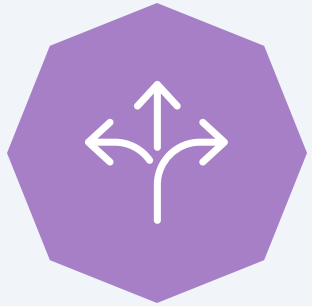

f1-cars-segmentation-and-classification
src
classification
base
config
loader
model
hypermodel.py
metric.py
model.py
saved
models
basic_model.pth
kmeans_model.pth
model_data
plot
utils
main.py
test.py
train.py
hypertuned
detection
config
saved
model
class_to_idx.json
model.pth
utils
videos
main.py
segmentation
Run 'detection' ^R
crop_img = frame[y:y + h, x:x + w]
# prepare input for the model
input_frame = prepare_input(crop_img)
output = get_prediction(input_frame, model)
cv2.putText(frame, classes[str(output.item())], (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
(36, 255, 12), 2)
cv2.imshow('F1 Car Detection', frame)
# time.sleep(0.1)
if cv2.waitKey(1) == 13:
break
video_cap.release()
cv2.destroyAllWindows()
if __name__ == '__main__':
time.sleep(2)
main()
Run:
hypertuning
detection
/Users/deborah/miniforge3/envs/torch-mac-hypertuning/bin/python /Users/deborah/Documents/f1-cars-segmentation-and-classification/src/detection/main.py
Process finished with exit code 0

6

Conclusions

Conclusions

Segmentation is not always step that must be performed



Convolutional Neural Network is always chosen for image classification but other models can be as accurate

Haar Cascade Classifiers are fast but not many models already trained are designed for cars

Thanks for the attention