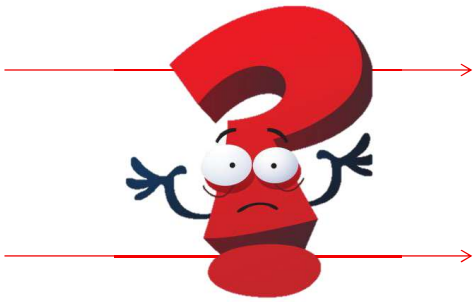
Machine Learning – How it works

Classical Machine Learning

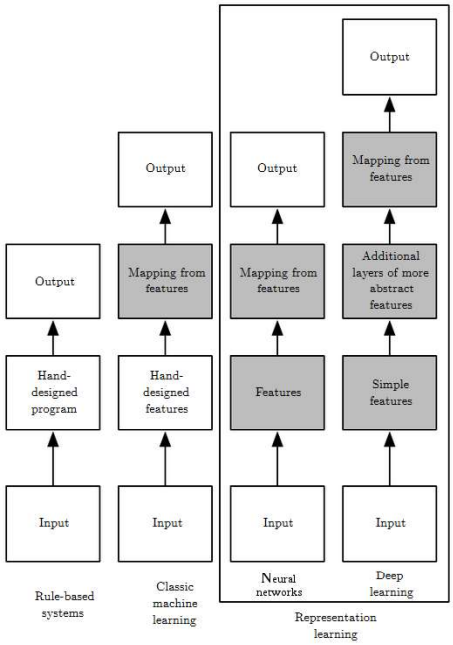


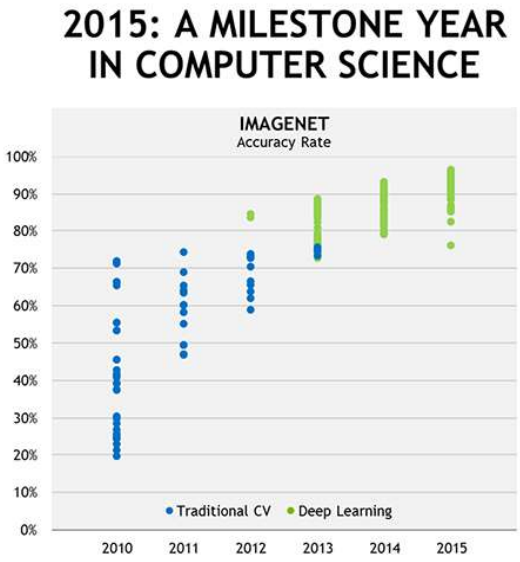
Cat

Burglar breaking in a house

How does my computer do it ???

Machine Learning – How it works





Machine Learning – How it works

Using the RGB pixels that compose the image?

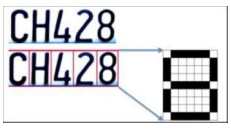
No! Not meaningful enough.
The picture of a cat can have a lot of different colors
Especially if we have the colors of the background

Machine Learning – How it works

Convert RGB pixels to meaningful numbers that can be understood by the machine (hand-crafted features)

=> Feature Extraction

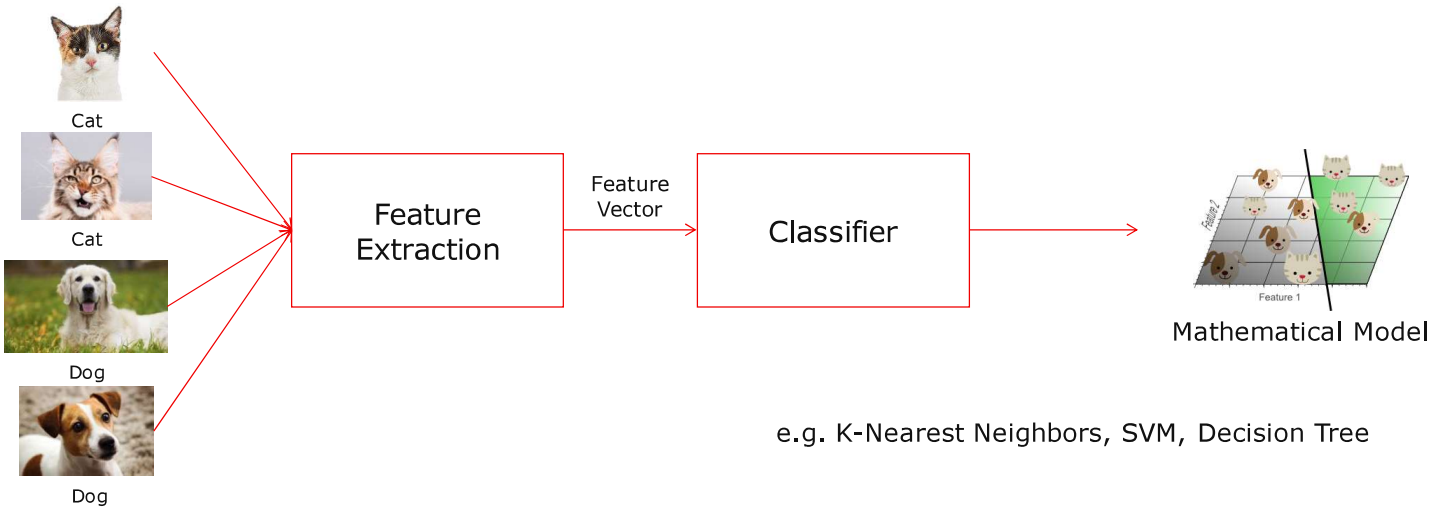
- Colors
- Shapes
- Points of interest
- Motion (video)
- Text/Numbers



Note: We can also build hand-crafted features for audio but that is not in the scope of this track

Machine Learning – How it works

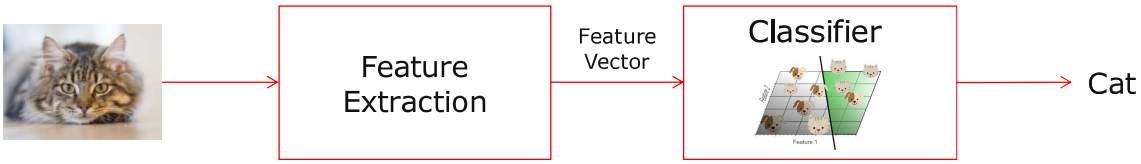
From meaningful features, we can learn patterns !
We feed what we call a "Classifier" with labelled data and it will build a mathematical model (Optimization). This is the training step.



e.g. K-Nearest Neighbors, SVM, Decision Tree

Machine Learning – How it works

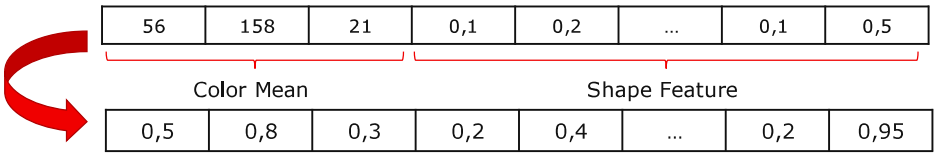
Once the classifier is trained (optimized mathematical model), we can Feed it with new samples and predict its label.



Machine Learning – How it works

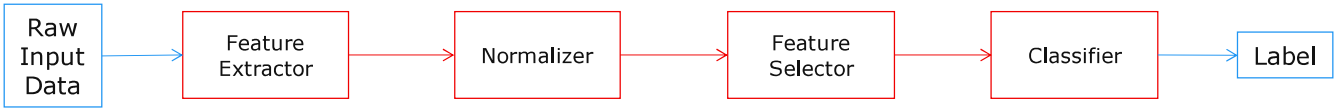
This pipeline is usually too simplistic to perform well

- Reducing the feature values is good for speed
- When concatenating feature vectors, some features can have larger values than others

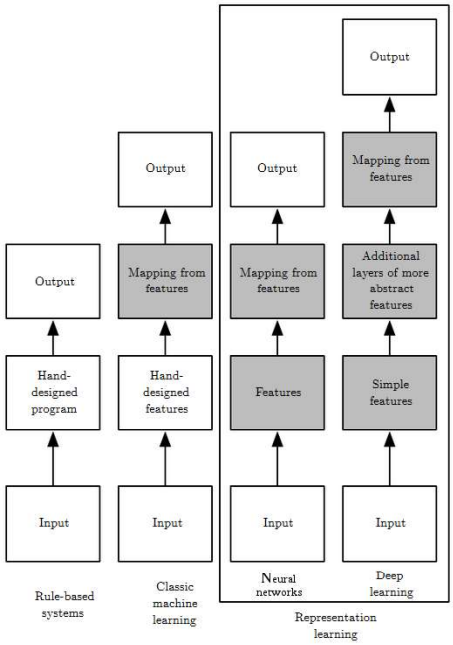


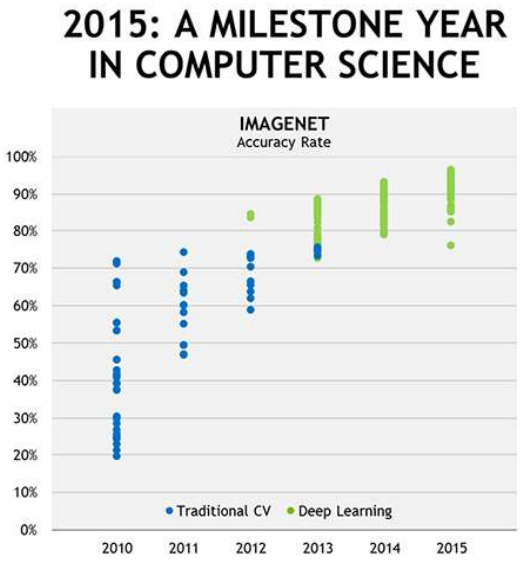
=> Normalize the feature vectors

- Select the relevant features only! E.g. Principal Component Analysis

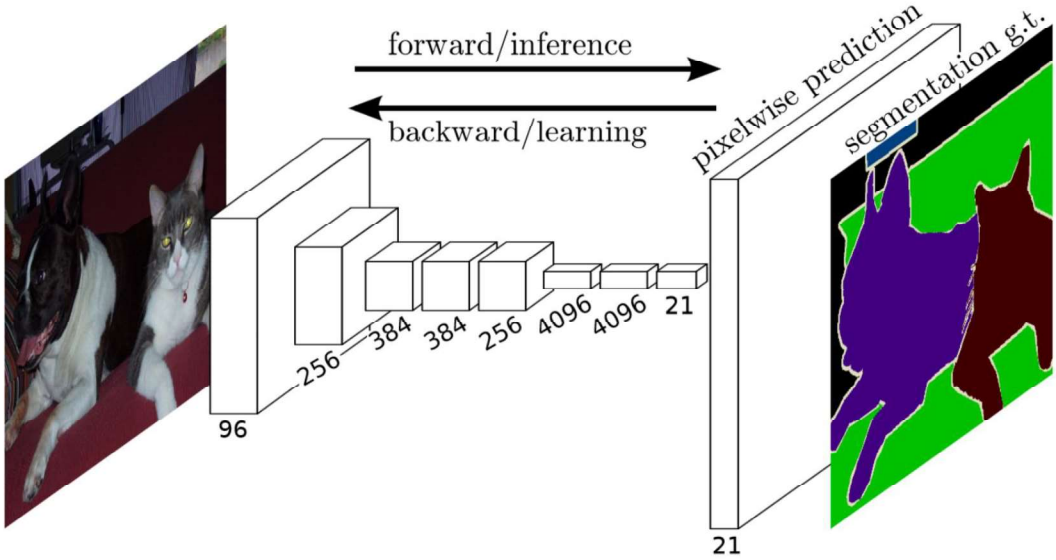


Machine Learning – How it works

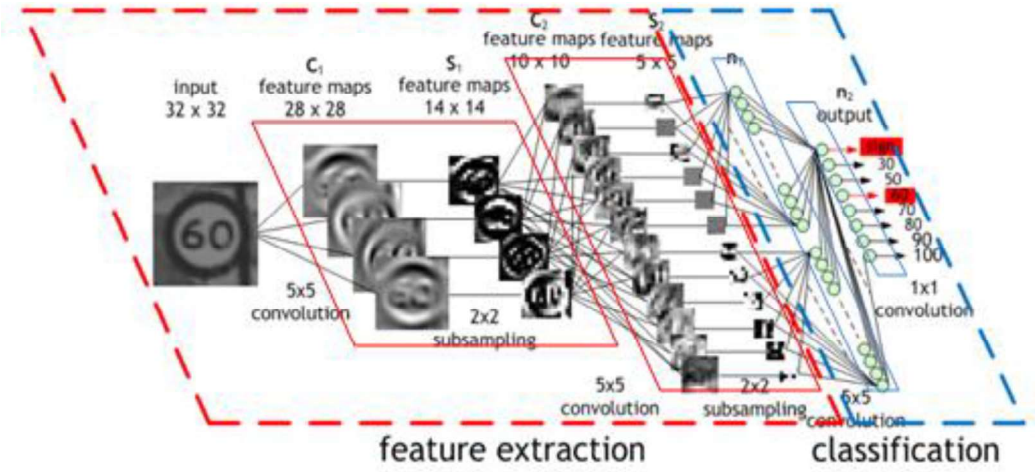




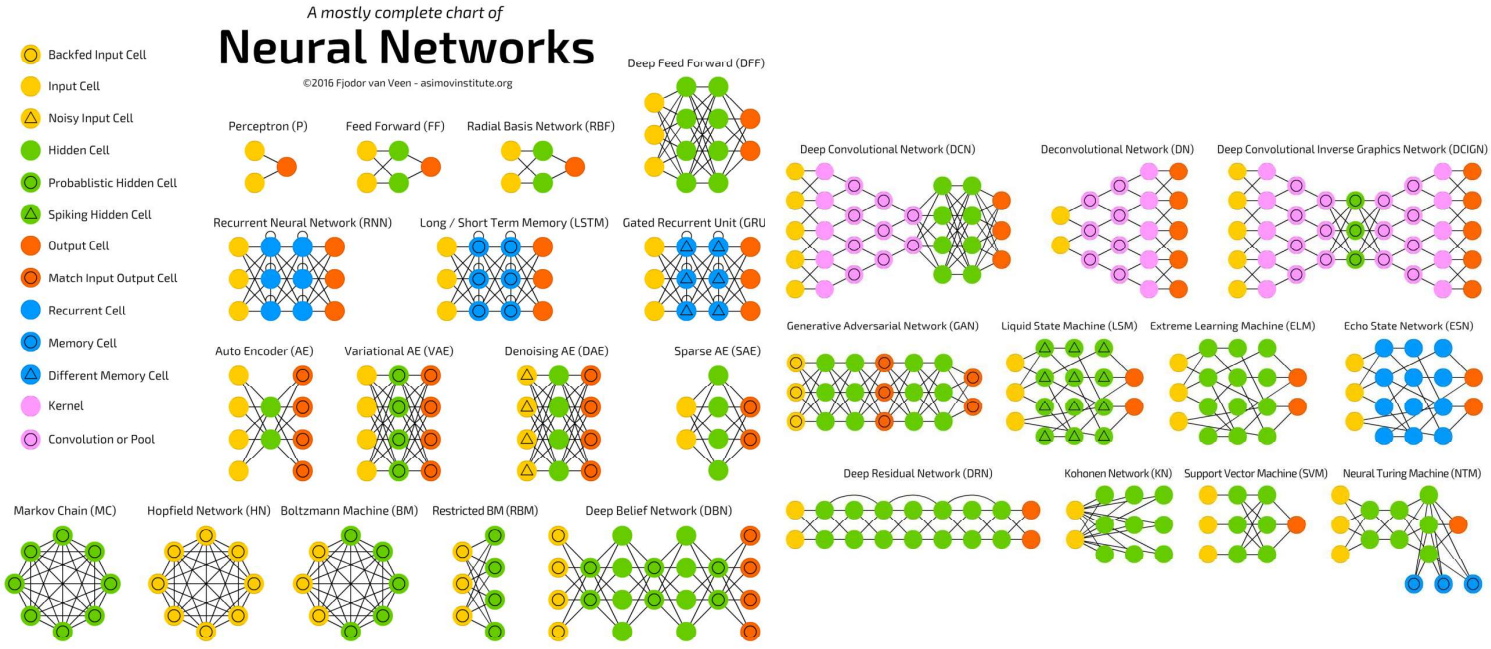
Machine Learning – in short



Machine Learning – in short

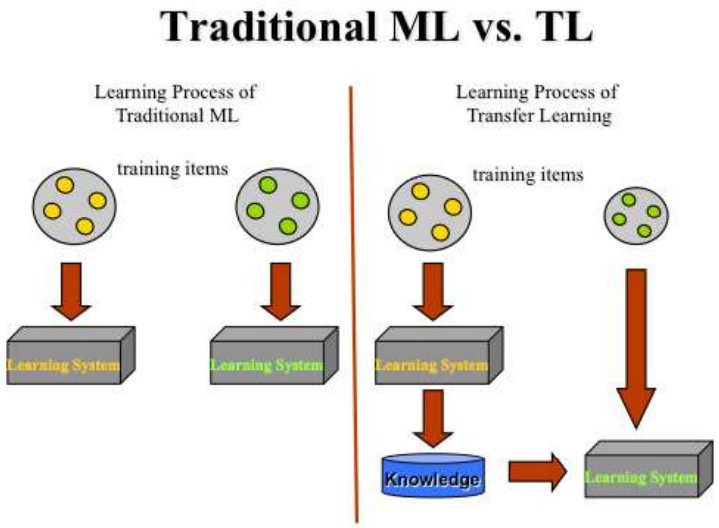


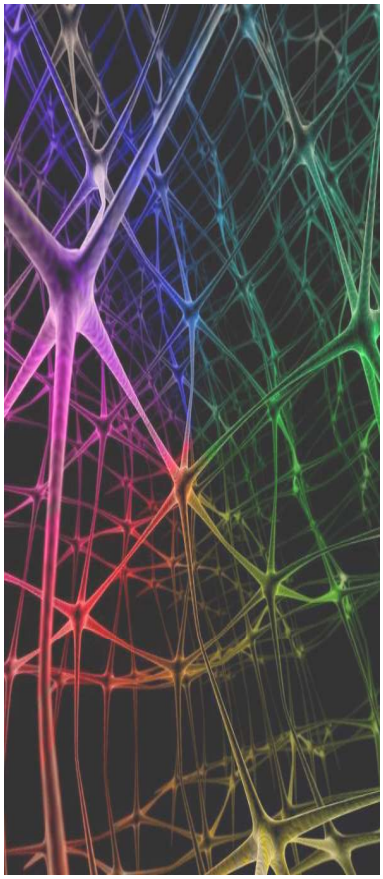
Machine Learning – in short



Machine Learning – in short

- Transfer Learning (TL)
- Start from a pre-trained model
  - No need to try architectures
  - Trains faster
  - Trains with less data





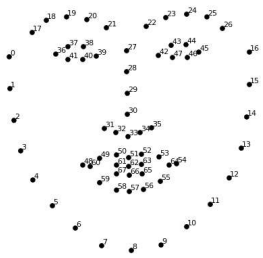
Questions?

About us? Barco? Machine Learning?

Hands-on – Introduction

Some Machine Learning, some coding and … some fun!

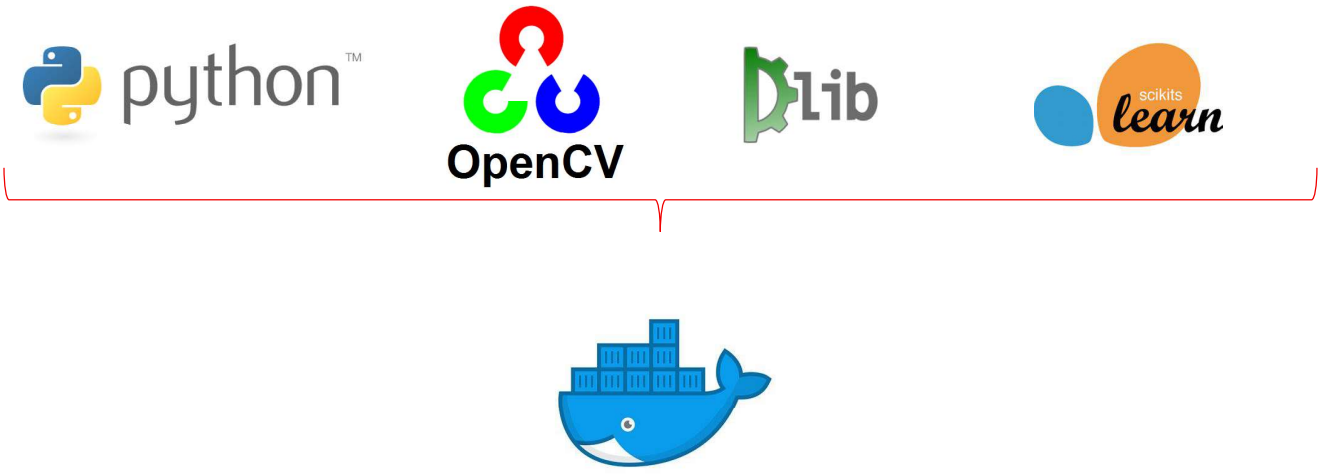
- Machine learning to detect facial landmarks
- Machine learning to detect face configuration and emotion
- Some code to overlay sprites on your face via your web cam



Open Mouth

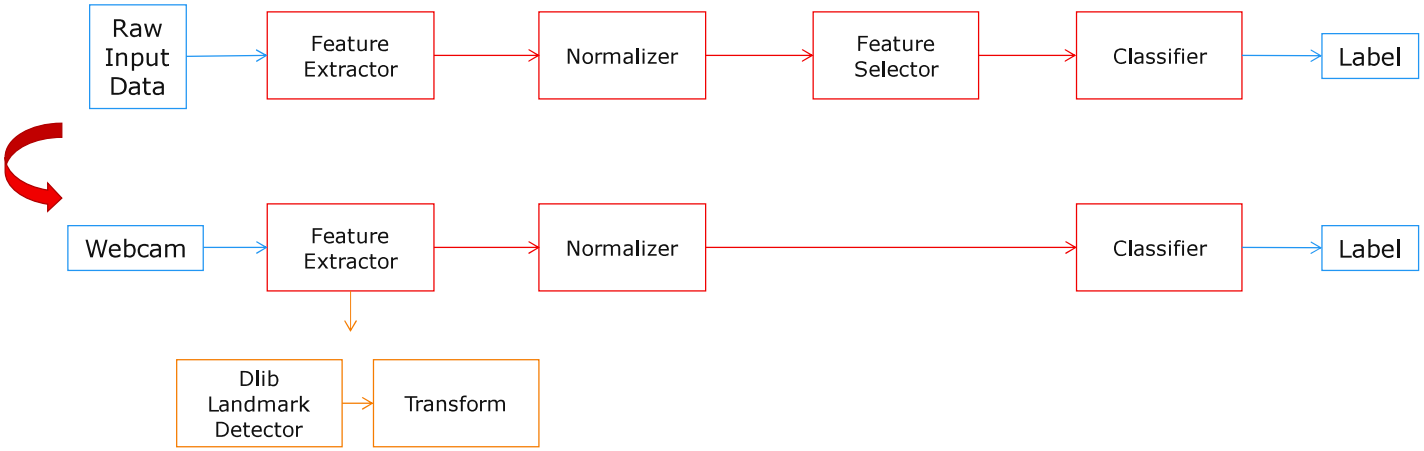
It is show time! … Again!

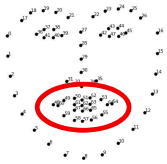
What will we use for this?



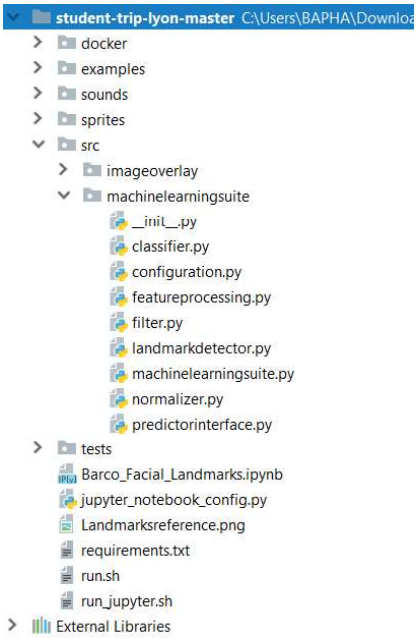
Hands-on – Introduction

Same pipeline as before

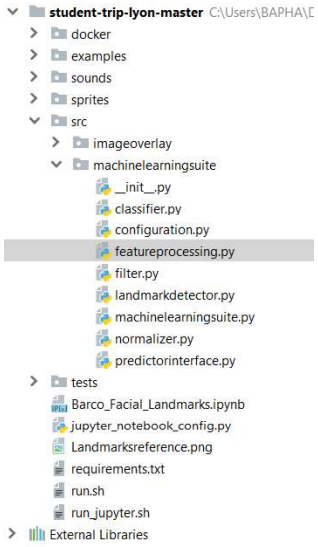




Hands-on – Material

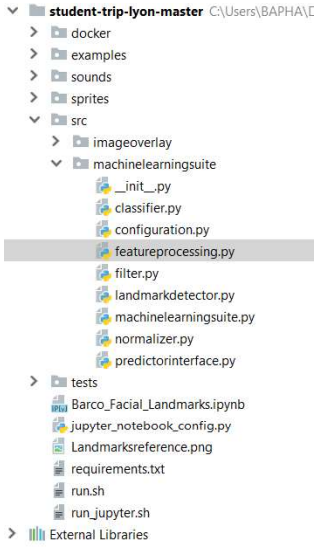


Hands-on – Material



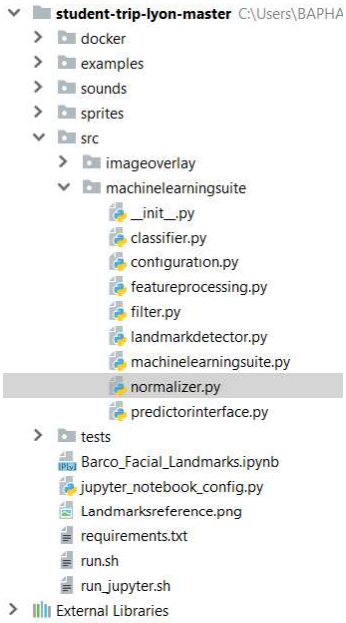

```python
class FeatureProcessing:

    def __init__(self, configuration):
        self.configuration = configuration
        self.parts = []
        self.markers = {
            "left-eyebrow": {
                "idx": list(range(17, 22)),
                "ref": 19
            },
            "right-eyebrow": {
                "idx": list(range(22, 27)),
                "ref": 24
            },
            "left-eye": {
                "idx": list(range(36, 42)),
                "ref": 41
            },
            "right-eye": {
                "idx": list(range(42, 48)),
                "ref": 46
            },
            "nose": {
                "idx": list(range(27, 36)),
                "ref": 30
            },
            "mouth": {
                "idx": list(range(48, 68)),
                "ref": 57
            },
            "jaw": {
                "idx": list(range(0, 17)),
                "ref": 8
            }
        }
```


Hands-on – Material



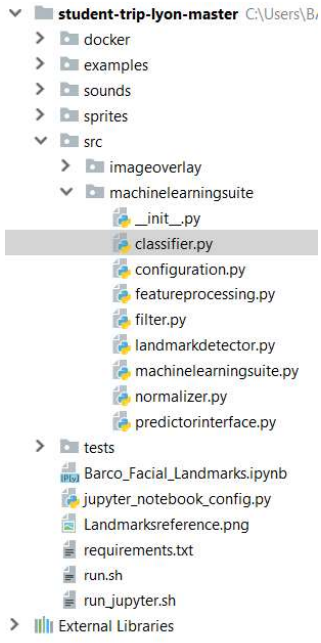

```python
def process(self, landmarks):
    if landmarks:
        if not len(landmarks[0]) == 68:
            return []
    else:
        return []

    for part in self.parts:
        if not part in self.markers:
            print("{} is not accepted as a face part".format(part))
            return []

    raw_feature_vector = []
    for part in self.parts:
        ref = self.markers[part]["ref"]
        markers = np.asarray(self.markers[part]["idx"])

        part_landmarks = [landmarks[0][i] for i in markers]
        ref_landmark = landmarks[0][ref]

        for landmark in part_landmarks:
            landmark_diff = np.subtract(landmark, ref_landmark)
            for element in landmark_diff:
                raw_feature_vector.append(element)
    return raw_feature_vector
```

Hands-on – Material



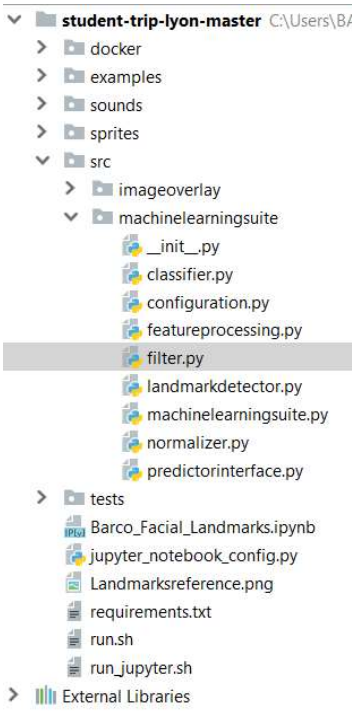

```python
class Normalizer:
    def __init__(self, configuration):
        self.configuration = configuration
        self.normalizer = preprocessing.Normalizer()

    def load_configuration(self):
        if self.configuration.normalizer:
            self.normalizer = self.configuration.normalizer

    def save_configuration(self):
        self.configuration.normalizer = self.normalizer

    def train(self):
        X = self.configuration.data_values
        self.normalizer.fit(X)
        self.configuration.data_values_normalized = self.normalizer.transform(X)
        self.save_configuration()

    def normalize(self, data):
        data_normalized = self.normalizer.transform(data)
        return data_normalized
```


Hands-on – Material



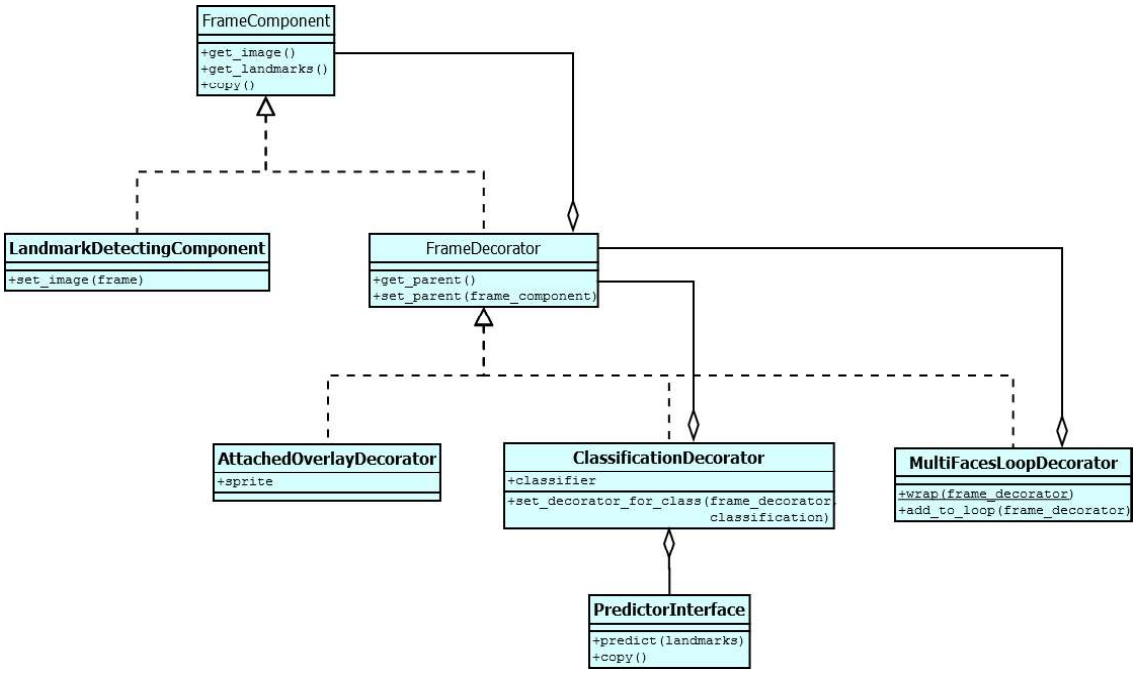

```python
class Classifier:
    def __init__(self, configuration):
        self.configuration = configuration
        self.classifier = svm.SVC(kernel='linear', C=1000)
        # self.classifier = svm.SVC()
        # self.classifier = svm.LinearSVC()
        # self.classifier = svm.SVC(decision_function_shape='ovo')

    def load_configuration(self):
        if self.configuration.classifier:
            self.classifier = self.configuration.classifier

    def save_configuration(self):
        self.configuration.classifier = self.classifier

    def train(self):
        X = self.configuration.data_values_normalized
        y = self.configuration.data_labels
        self.classifier.fit(X, y)
        self.save_configuration()

    def predict(self, data):
        return self.classifier.predict(data)
```

Hands-on – Material

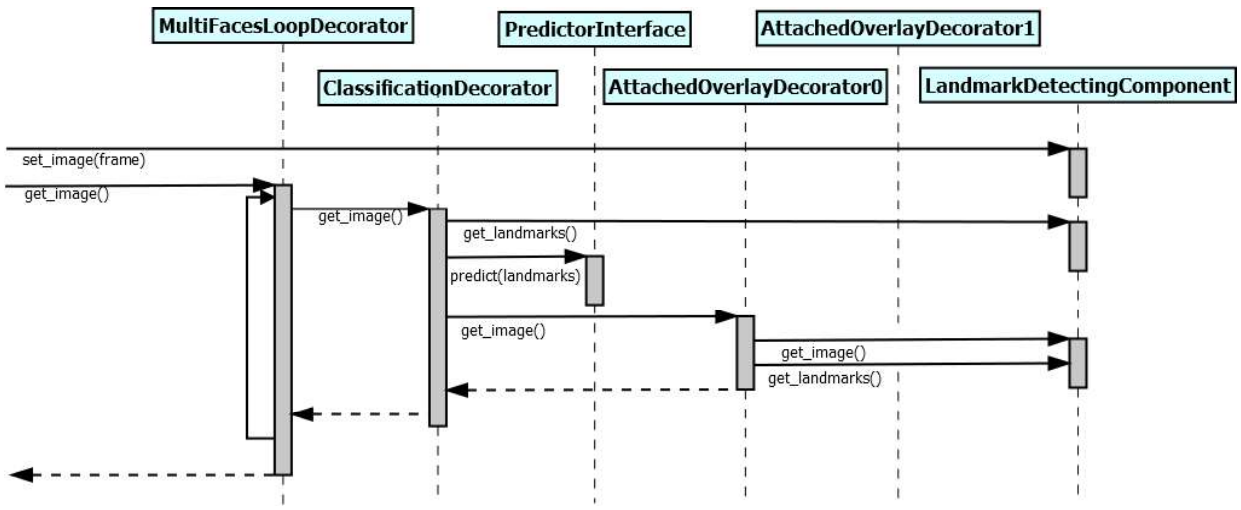



```python
class Filter:

    def __init__(self, queue_size):
        self.queue_size = queue_size
        self.queue = collections.deque(maxlen=queue_size)

    def filter(self, prediction):
        self.queue.append(prediction[0])
        if len(self.queue) == self.queue_size:
            queue = np.asarray(self.queue)
            return np.bincount(queue).argmax()
        else:
            return None
```


Hands-on – Material



Hands-on – Material



Hands-on – Material

```
    def predicting_example():
        # Instanciate a new landmark detector
        detector_data_path = '../../data/shape_predictor_68_face_landmarks.dat'
        landmark_detector = LandmarkDetector(predictor_file=detector_data_path)

        # Create a frame component with landmarks
        base_component = LandmarkComponent(landmark_detector)

        # Instantiate and initialize the trained predictor
        predictor = PredictorInterface('../examples/emotion.pkl')
        predictor.initialize()

        # Add decorator for the predictor
        predictor_decorator = ClassDecorator(parent_component=base_component, classifier=predictor)
        sunglasses = SpriteDecorator(base_file_name='../sprites/sunglasses')
        eyes = SpriteDecorator(base_file_name='../sprites/eyes')
        predictor_decorator.set_decorator_for_class(sunglasses, 0)
        predictor_decorator.set_decorator_for_class(eyes, 1)

        multifaces = AllFaces.wrap(predictor_decorator)
```

```
            base_component.set_image(frame)
            output = multifaces.get_image()
```

Hands-on – Get Started

- Get docker image from:
  docker pull bapha/student-trip-lyon
- Get source code from:
  git clone https://github.com/bhanssens/student-trip-lyon.git
- ./run.sh
- Start ‘pycharm’ or ‘jupyter notebook’
- Have a look in ./examples

Hands-on – Practice

1. Show facial landmarks
2. Add overlay
3. Train classifier: Open/Closed mouth
4. Link overlay to classifier
5. Go crazy!

Hands-on – Further Exercises

- Create a text overlay
- Face swap
- Draw a new sprite at run-time
- Create animations
- Program a (competitive) game (e.g. eating dots of the screen)
- Train a classifier for tilting your head in one or the other direction
- Create a nodding yes or no classifier
- Use the color of pixels to make a prediction (e.g. eye or hair color)
- Check out examples in dlib to label and train new object detectors and shape predictors

Final Note

Thanks for attending! Hope you enjoyed it!

The team


Steven Delputte
Technology Manager


Jürgen Slowack
NTI Project Manager


Thijs Vermeir
Research Engineer


Karel Moens
Research Engineer


Baptiste Hanssens
Research Engineer


Jonas De Vylder
Research Engineer


Samuel Dauwe
Research Engineer


You ?

- **User Experience**: 3D rendering frameworks, web technologies, human machine interaction,…
- **GPU**: high performance, low delay, high throughput image processing (from almost driver level till CUDA/OpenCL, VULKAN,…)
- **Security**: product risk assessments, ethical hacking,…

ENABLING BRIGHT OUTCOMES

| youtube.com/BarcoTV
| linkedin.com/company/Barco
| twitter.com/Barco
| facebook.com/Barco