IAPR 2024 - Project

Coin Detection Challenge

Théo Heng, Océane Voland & Maïka Nogarotto

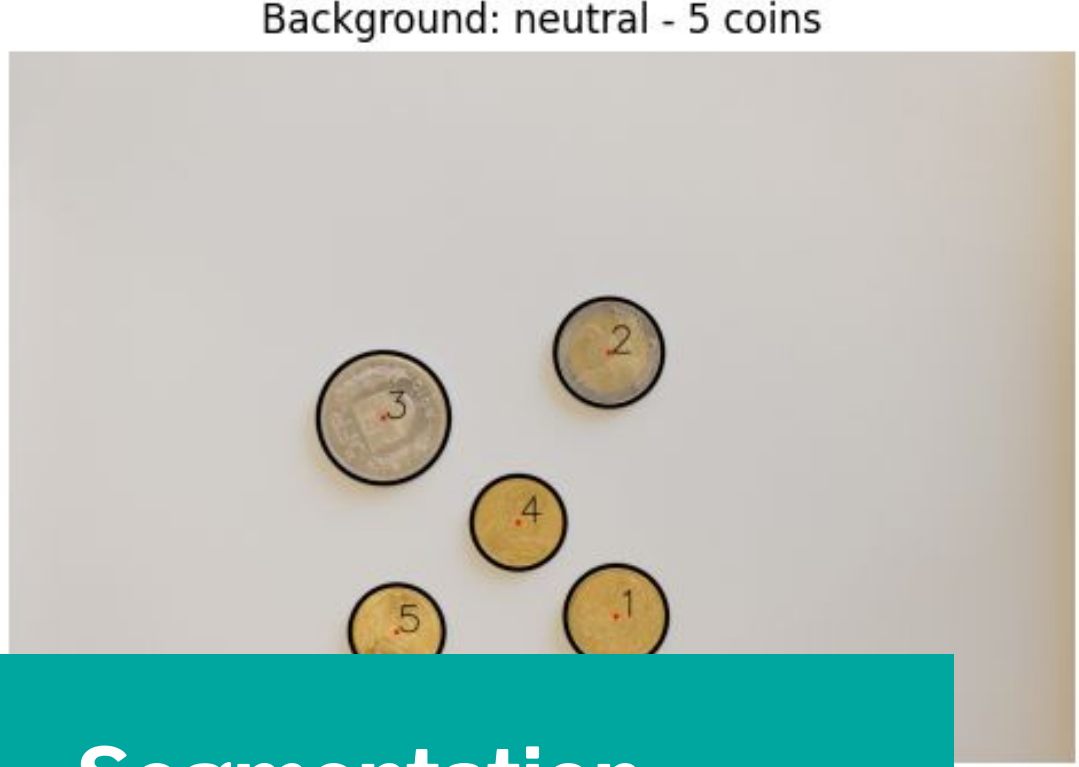
EPFL - 31th May 2024

Project overview

General structure

- **Segmentation**
  - Background detection
  - Coins recognition
  - Coins isolation
- **Feature analysis**
  - Area extraction
  - Contours enhancement
- **Classification**
  - **Template matching**
  - **Deep learning model**

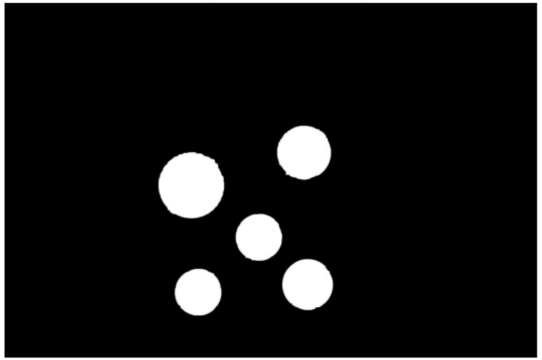
Illustration created by Oliver Lemer, a former student of the IC faculty. (olivier.lemer@alumni.epfl.ch)



1. Segmentation

1. Segmentation

General pipeline



Machine Learning

Pattern Matching

Classification

1. Segmentation

Background detection

- **Type of background detected using thresholds on specific colors**

A lot of pink/red

⬇

Hand background

A lot of blue/green

⬇

Noisy background

Else

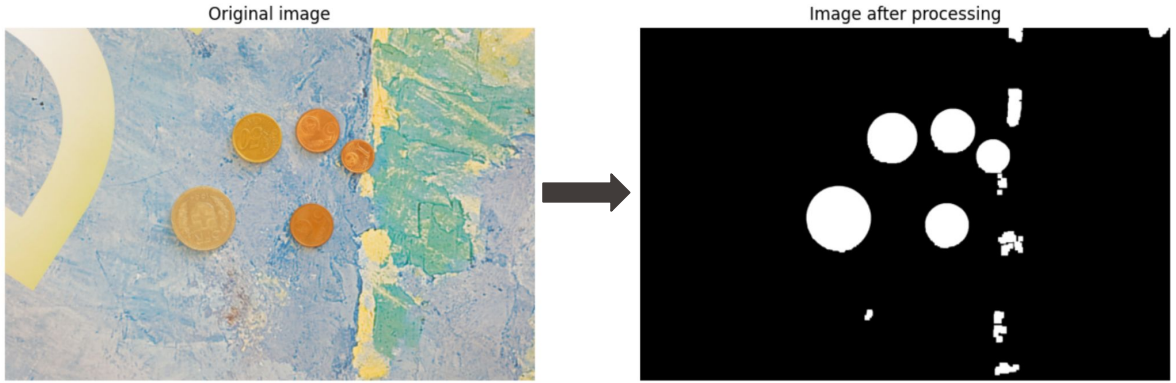
⬇

Neutral background

1. Segmentation

Segmentation

- **Blurring** (Median, Gaussian)
- **Thresholding** to binarize the image into foreground and background
  - Thresholds depend on background and are based on coin's colors
- **Morphological operations**
  - Closing : remove holes
  - Opening : get smoother contours

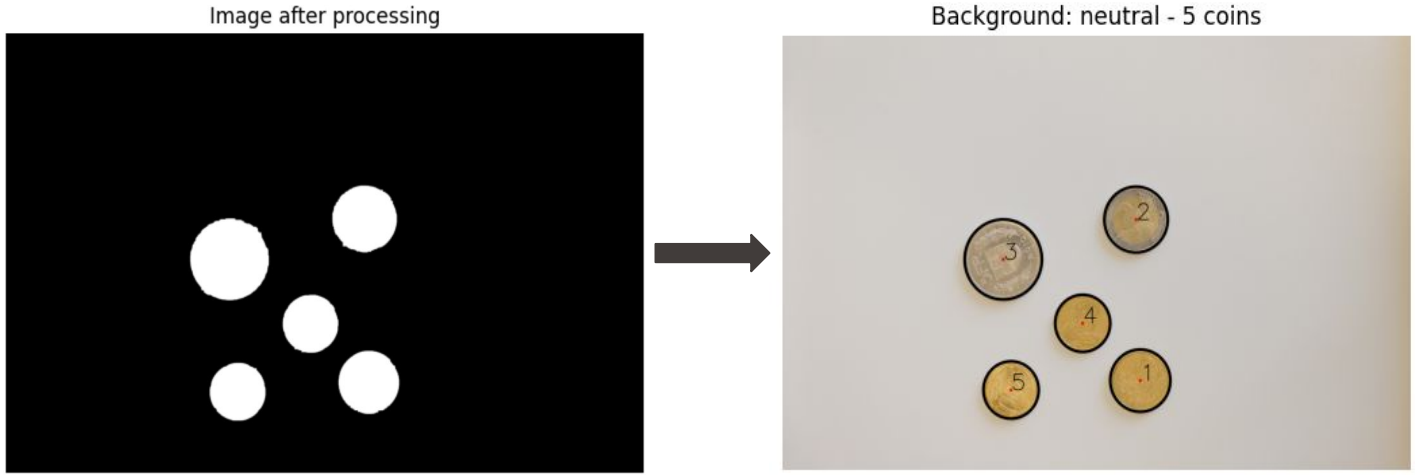


Some noise remains

1. Segmentation

Coin identification

- **Find circles in images (cv2.HoughCircles)**
  - Optimal parameters ensure that noise is not considered as coins
  - Parameters depend on background
  - Returns center and radius of each circle

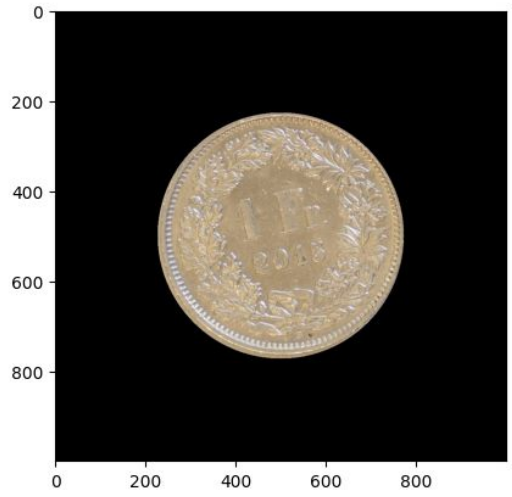


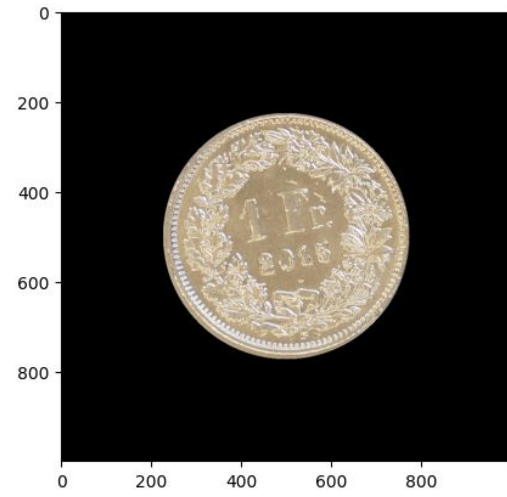
Segmentation

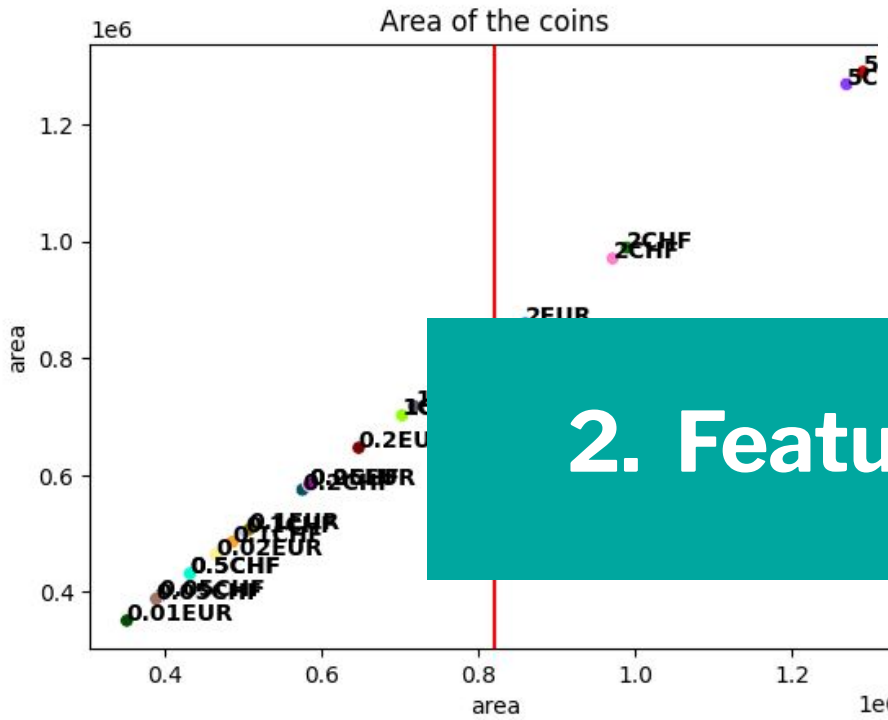
Coin isolation

- **Using the detected circles, we can create sub-images containing individual coin:**
  - 1000x1000 pixels
  - Black background
  - Coin centered

- **Segmentation process 100% automatic**
- **100% of the coins detected**
- **Not robust to external images to the dataset**





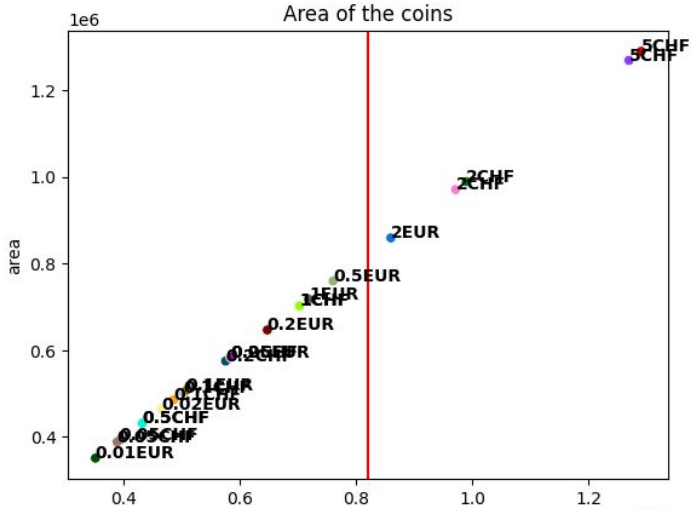


2. Features Analysis

2. Features Analysis

Area extraction

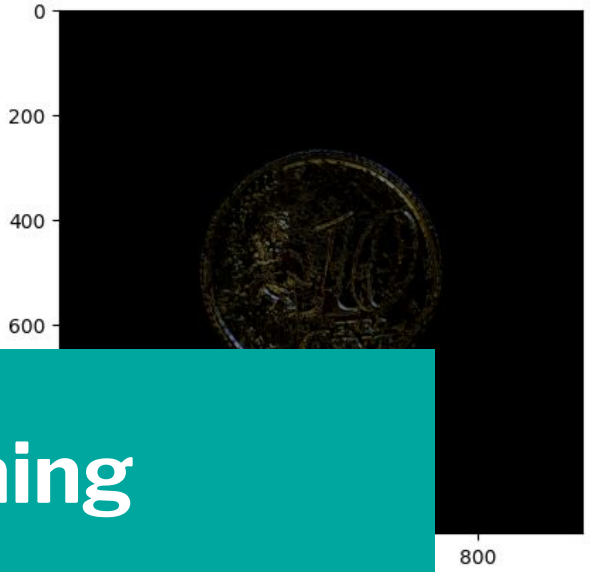
- **Get the number of non-black pixels on segmented images**
  - Faster way than computing the radius again
- **Can analysis the classes depending on their area**
  - Can be used to distinguish small from big coins
    - Useful to save computation resources and time
  - Not meaningful enough to be used alone for classification
    - Small coins have too similar area



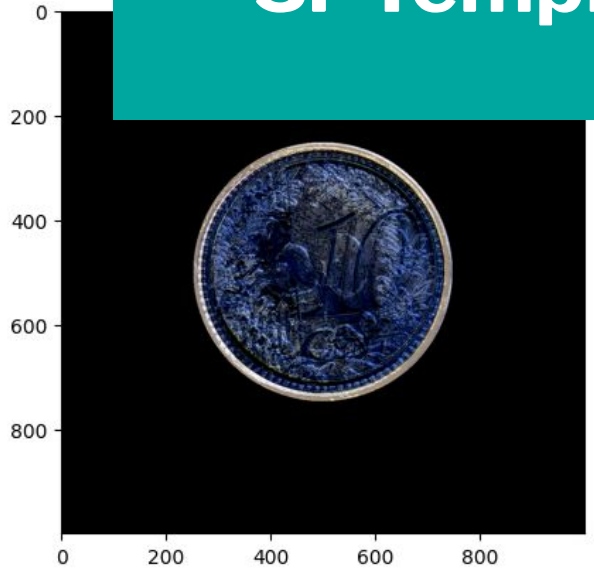
2. Features Analysis

Contour extraction and enhancement

- **Contours extracted using Canny edge detection**
  - Use histogram equalisation to enhance the contrast
  - Find the contours
- **Contour enhancement**
  - Get the images with equalized detected edges
  - Increase the value of the pixel in the image at the contours location
- **Benefits**
  - Allows a better differentiation of the coins



3. Template Matching



3. Template Matching

Principle and concepts

- **Method in general**
  - Set an image as a template
  - Find on the image where the template matches the best

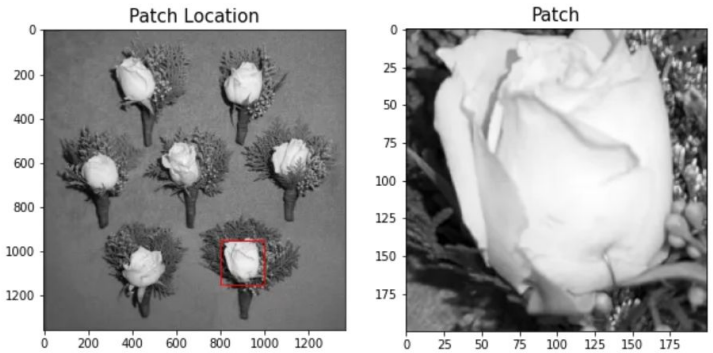


Figure 2: Template/Patch Image (Image by Author)

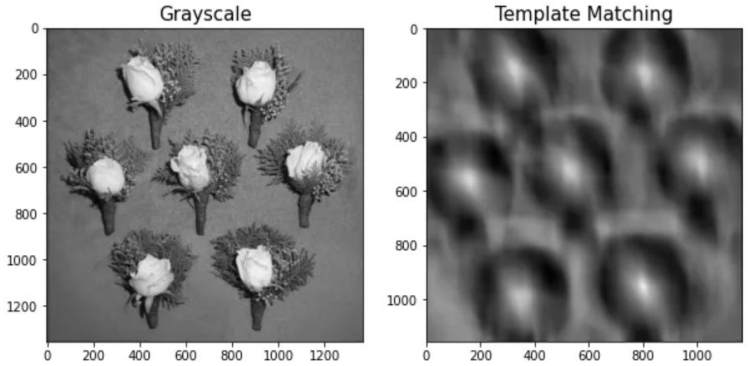


Figure 3: Performing Template Matching (Image by Author)

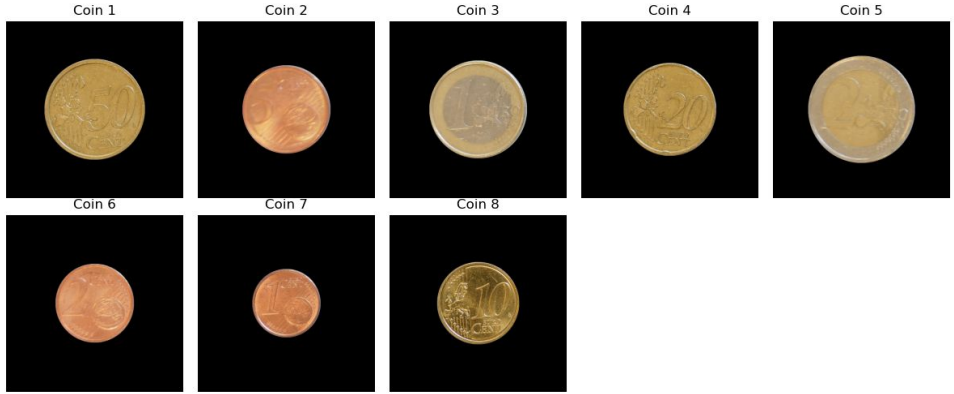
https://medium.com/analytics-vidhya/image-processing-template-matching-aac0c1cbe2c0

3. Template Matching

Our adaptation

Reference coins

- **Which reference coin is our testing coin the closest ?**
  - Compare our testing coin to each reference coin and take the most similar one



3. Template Matching

Application

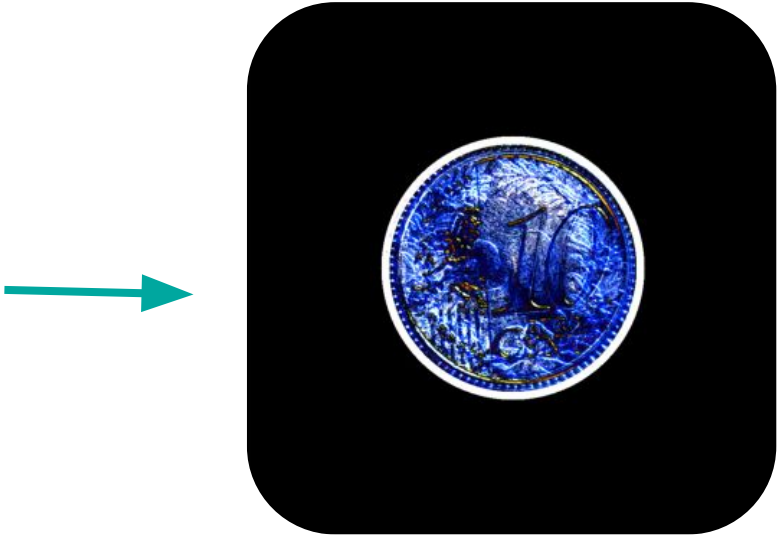
- **Methods used**
  - Squared difference

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x', y+y'))^2$$

Test

VS

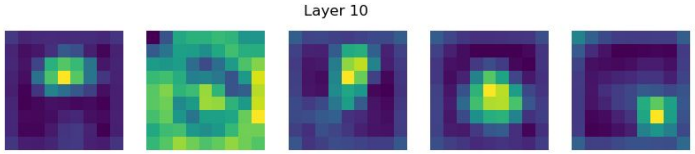
Reference



3. Template Matching

Results

- **Accuracy : 66,24%**
- **Advantages**
  - Only need one example of each coins type
  - Don’t need training
  - Fast
  - Transparent pipeline
- **Drawbacks**
  - Lack of precision
  - Outperformed by most ML technics
  - Struggle with outliner coins
  - Cropping imprecisions can dramatically impact the result



4. Machine Learning

google

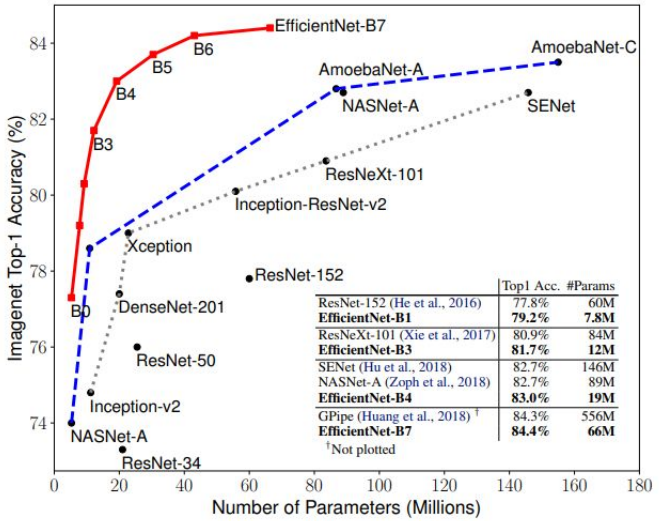
/efficientnet-b3

huggingface.co

4. Machine Learning

Motivations

- **Why a pretrained model?**
  - Reduced Training Time and Computational Resources
  - Better Generalization and Robust Feature Extraction
  - Effective Use of Limited Data
- **Why EfficientNet B3?**
  - Lightweight and SOTA performances
  - Well documented and easy to use
  - Nice trade-off performance/size



4. Machine Learning

Training specifications

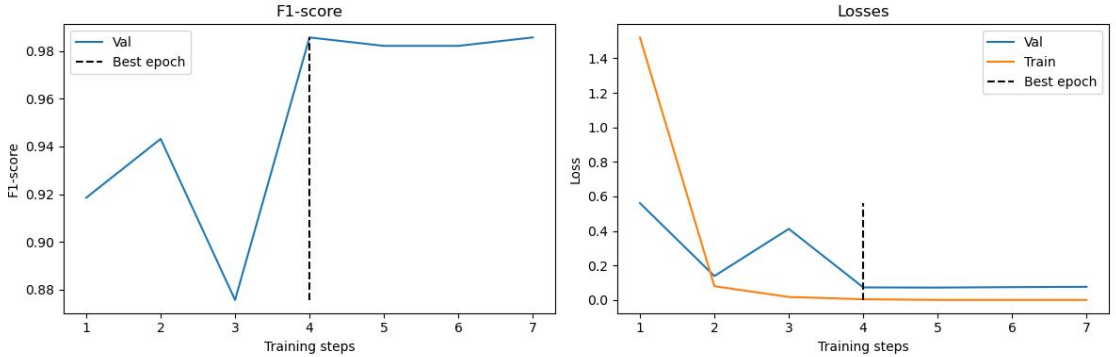
- **Dataset**
  - Data Augmentation -> 6000 train images - 800 validation images
- **Special features**
  - Adaptive learning rate with scheduler
  - Gradient accumulation to emulate bigger batch sizes without RAM overflow
  - Confusion Matrix to identify classification flaws

	Epochs	Batch size with accumulation	Learning Rate	Training time
EfficientNet B3	10	32 x 2 = 64	5e-5	60 mins

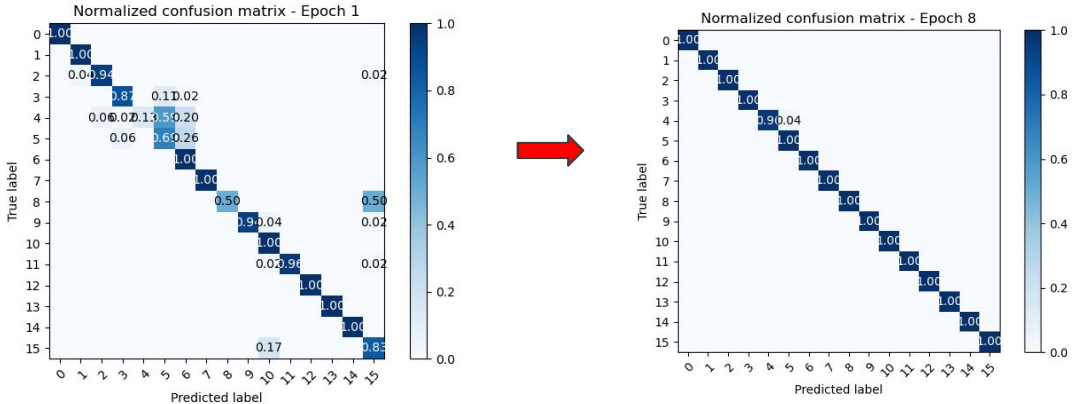
4. Machine Learning

Results

- **Training loss:**



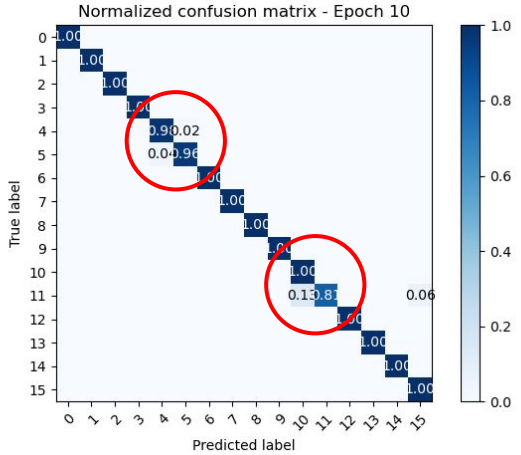
- **Confusion matrices:**



- **Results:** **98.4%**

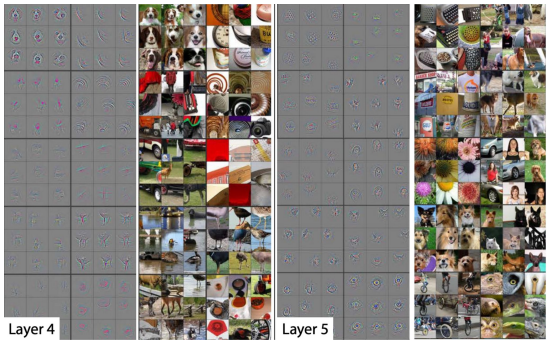
4. Machine Learning

Analysis

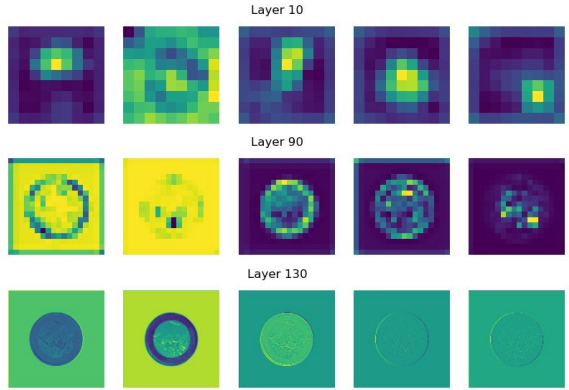


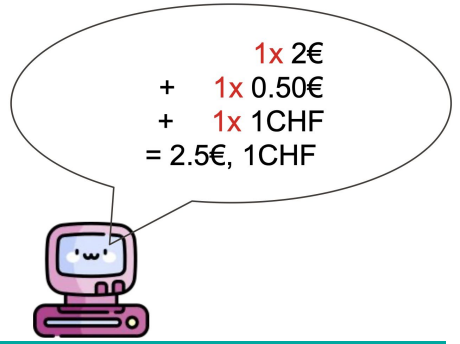
- **Why is 100% so hard to reach ?**
  - 20 and 10 cents often confused together
  - Not enough data to fix this

- **What are the features extracted by the model?**



Example from EE-451 slides





5. Conclusion

5. Conclusion

Project summary

- **Segmentation:** 100% of coins detected and extracted
- **Template matching:** 66.24%
  - Require very few data and no training
- **Deep learning model:** 98.4%
  - Require training and manual annotation

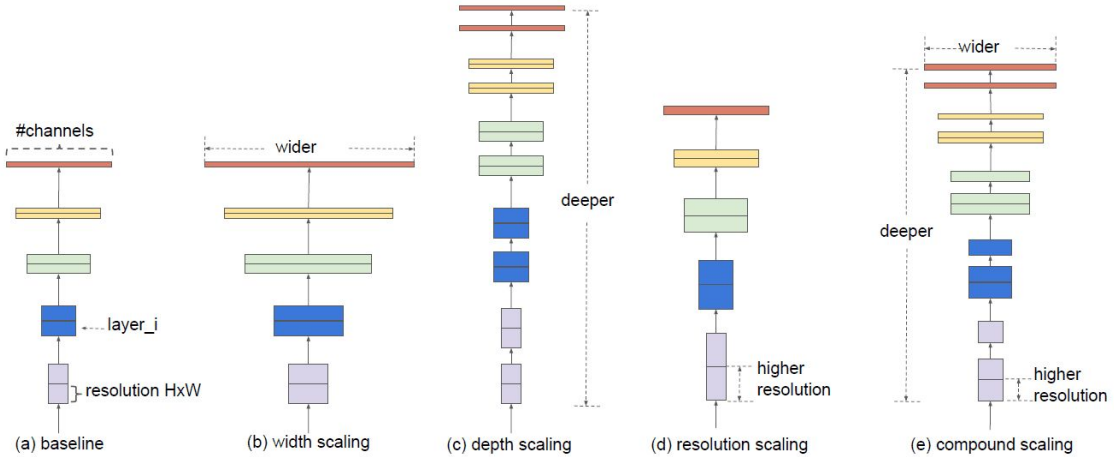
2nd place !

DALL·E 2

6. Annexe

EfficientNet B0 -> EfficientNet B3

	NB parameters	Resolution	Model size	Output channels
EfficientNet B0	5.3 M	224 x 224	16.4 M	1280
EfficientNet B3	12 M	300 x 300	43.4 MB	1536

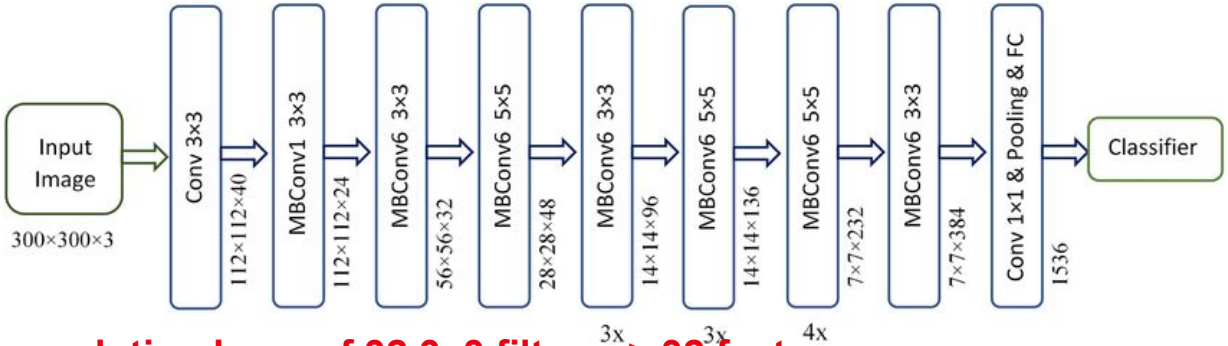


Compound Scaling:
scaling factor to depth, width, and resolution based on a compound coefficient (φ).
-> ensures that the network grows in a balanced manner.

6. Annexe

EfficientNet backbone

$$\text{swish}(x) = x\,\text{sigmoid}(\beta x) = \frac{x}{1 + e^{-\beta x}}.$$

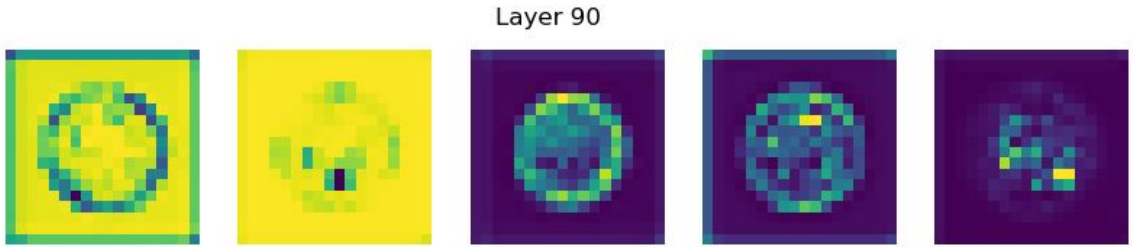


- **1st convolution layer of 32 3x3 filters -> 32 features map**
- **Successive sets of MBConv blocks**
  - Expansion, Depthwise convolution, projection, skip connection, batch normalisation+activation
- Each sets increases the number of filters and may change the spatial dimensions by downsampling
- **Global average pooling layer ->** average each feature map to a single dimension
- **Fully connected (dense) layer ->** combines the features extracted to produce final logits
- **Output layer ->** Use softmax to produce the final class predictions

6. Annexe

Extracting features from the model

- Hook Function: Captures outputs of convolutional layers.
- Iterates over all layers in the model and registers the hook function to each convolutional layer
- Disables gradient calculation to save memory and computations, as we are only interested in the outputs, not in training.
- Forward Pass: Passes the input image through the model to gather feature maps.
- Removes all the hooks after the forward pass to avoid any side effects on the model.



6. Annexe

Choice of Optimizer

- **ADAM (Adaptive Moment Estimation)**
  - uses estimates of first and second moments of the gradients (mean and uncentered variance) to adapt the learning rate for each parameter
  - computationally efficient and requires little memory
- **ADAMW**
  - improvement over Adam by decoupling weight decay from the gradient update. Theoretical better convergence
  - Requires careful tuning of the weight decay parameter. The improvement in performance might not be significant for all tasks compared to standard Adam
- **SVG (Stochastic average gradient)**
  - Reduces variance of gradient estimates by averaging gradients from all past iterations. Can converge faster than SGD in some cases
  - Requires storing the entire gradient history -> higher memory usage. Not as widely used or well-understood as Adam.

2. Features analysis

Overview

- **Area extraction**
  - Get the number of non-black pixels
  - Allows to separate big from small coins
- **Contours enhancement**
  - Get the edge using Canny edge detection
  - Deep learning model

3. Template matching

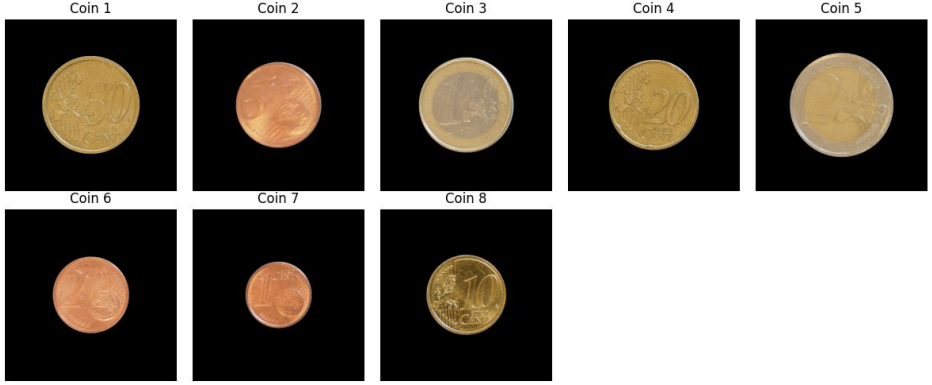
Overview

- **Principle and concepts**
  - Method in general
  - Our adaptation
- **Application**
  - Require data
  - Methods used
- **Results**
  - Advantages
  - Drawbacks
- **Possible improvements**

3. Template matching

Application

- **Data required**
  - Only requires the reference images



3. Template matching

Possible improvements

- **Create an outlier “database”**
  - Require to use the training images
- **Add more images to reference sets**
  - Require more computational resources and time
- **Rotate the image during the comparison**
  - Require more computational resources and time
- **Get more general reference images**
  - Fuse all coin of the same time together
  - Use SVD or KLT to get eigenvectors
  - Require to use the training images
- **Use other comparison metrics**