Challenge Task A

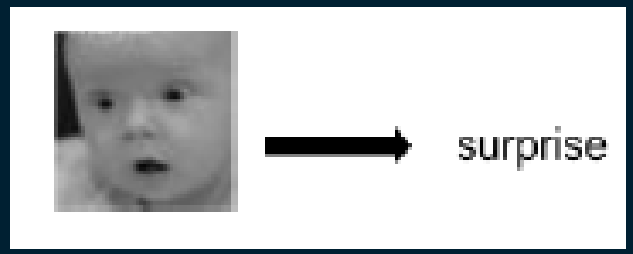
Salvatore Cavallaro

University of Catania

salvatore.cavallaro7@studium.unict.it

Overview of the solution

- Challenge A: Detect human emotions on the basis of people's facial expressions

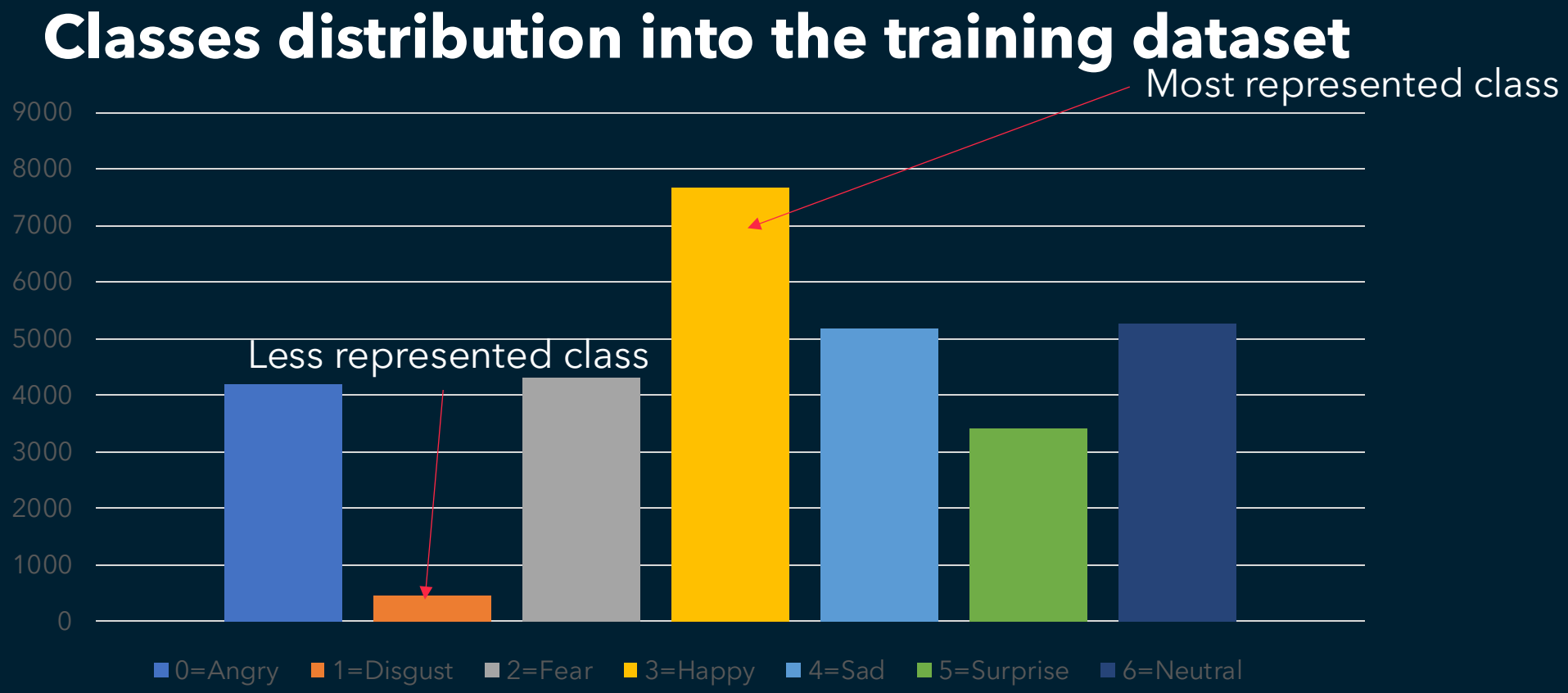


Overview of the solution

Dataset

- Grayscale faces images 48x48 pixels
- Images as .jpg files placed in *images_train* and *images_test*
- 30503 training labeled samples → *challengeA_train.csv*
- 4038 test samples to be labeled
- 3 channel images

Overview of the solution



Loop Q PRIZE

Overview of the solution

- *Python* and *Google Colab* Notebook on *CUDA* runtime

- Images loaded from *training* and *test* directories exploiting *OpenCV* library
  - Slowness due to reading from directory on Google Drive

  - Dataset stored as a Numpy array on Drive

Overview of the solution

- Models realized using Machine Learning and Deep Learning techniques
- *Keras* and *Google Tensorflow* libraries for Deep Learning
- *Scikit-learn* library for Machine Learning

Overview of the solution

- 5 models defined to solve the challenge
- Models choices based on explanatory analysis of the dataset
  - https://towardsdatascience.com/exploratory-data-analysis-ideas-for-image-classification-d3fc6bbfb2d2
- Main issue: what are the features to extract to classify our images?
- Preliminary investigation on methods used by researches to address similar problems

Overview of the solution

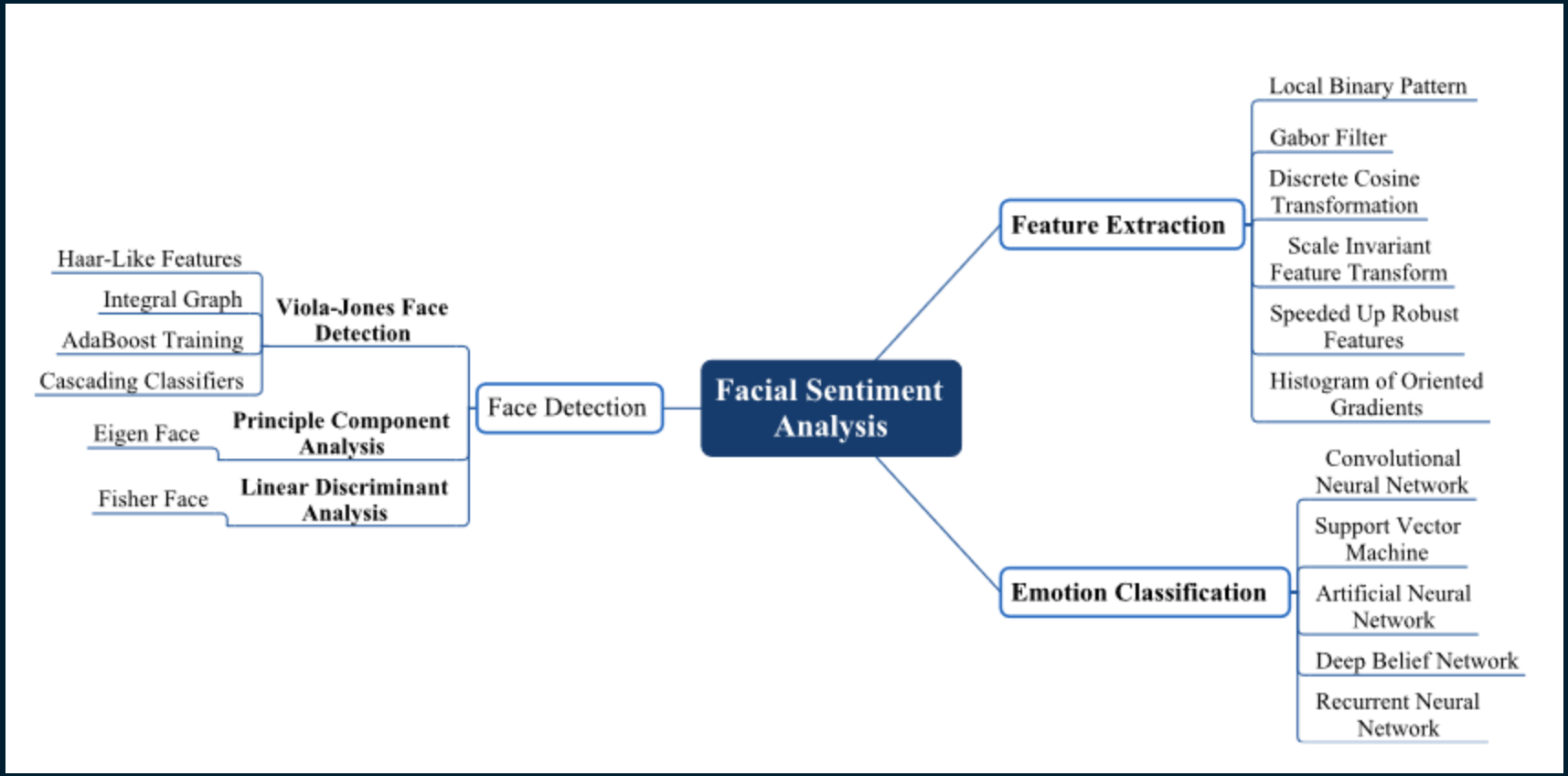
- A first analyzed scientific article:

K. Patel *et al.*, "Facial Sentiment Analysis Using AI Techniques: State-of-the-Art, Taxonomies, and Challenges," in *IEEE Access*, vol. 8, pp. 90495-90519, 2020, doi: 10.1109/ACCESS.2020.2993803.

- State of art and different possible approaches for sentiment classification of images
  - SVM + Computer Vision, Convolutional Neural Networks, Recurrent Neural Networks, Transfer Learning on pretrained models and so on

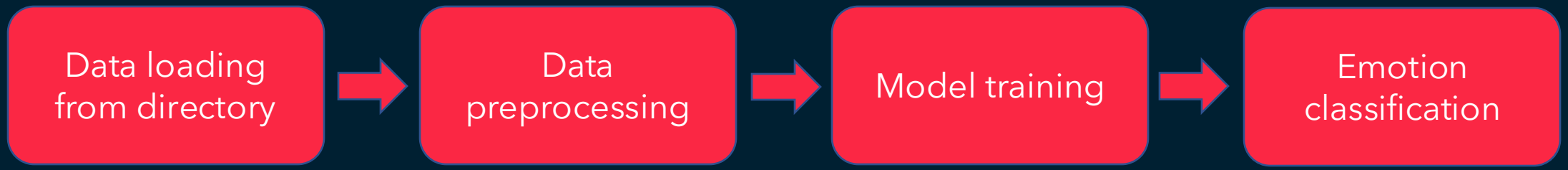
- Discussion of the available datasets
  - FER-2013 dataset

Overview of the solution



Overview of the solution

Data flow



- OpenCV
- Numpy
- Pandas

- Flattening
- Normalization
- Hog transform
- PCA
- CNN
- Data augmentation

- SVM
- MLP NN
- CNN
- Transfer Learning CNN

- Chosen model based on best accuracy

Loop Q PRIZE

Overview of the solution

First Model: Support Vector Machine

- Source: https://rpubs.com/Sharon_1684/454441
- Library: *Scikit-learn* (SVM) and *OpenCV* (Feature extraction)
- Extracted features: *Histogram of Oriented Gradients*
- Feature reduction: *Principal Component Analysis*
- Implemented model: *Support Vector Machine*
- Accuracy: **28% on test** → **Lowest**

Overview of the solution

Second model: Multilayer Perceptron Neural Network

- Library: *Keras* and *Tensorflow 2.0*
- Architecture: *FC Neural Network with 4-layers*
- Parameters:
  - Training Epochs: 50
  - Batch size: 64
  - Activation Function: ReLU
  - Optimizer: Adam with LR=0.001
  - Dataset split: 80% train - 10% validation - 10% test
- Accuracy: **38% on test**

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2048)	14157824
dense_1 (Dense)	(None, 1024)	2098176
dense_2 (Dense)	(None, 512)	524800
dense_3 (Dense)	(None, 7)	3591

Total params: 16,784,391
Trainable params: 16,784,391
Non-trainable params: 0

Loop Q PRIZE

Overview of the solution

Third Model: Convolutional Neural Network

- Library: *Keras* and *Tensorflow 2.0*

- Architecture: *Convolutional Neural Network* with

Batch Normalization, *MaxPooling, Dropout* and *3-layers FC NN*

- Data Augmentation and Dataset Shuffling

- Parameters:
  - Training Epochs: 50
  - Batch size: 64
  - Activation Function: ReLU
  - Optimizer: Adam with adaptive LR
  - Dataset split: 80% train - 10% validation - 10% test

- Accuracy: **88% on test**

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 46, 46, 64)	1792
batch_normalization_8 (Batch	(None, 46, 46, 64)	256
max_pooling2d_4 (MaxPooling2	(None, 23, 23, 64)	0
dropout_5 (Dropout)	(None, 23, 23, 64)	0
conv2d_9 (Conv2D)	(None, 21, 21, 128)	73856
batch_normalization_9 (Batch	(None, 21, 21, 128)	512
max_pooling2d_5 (MaxPooling2	(None, 10, 10, 128)	0
dropout_6 (Dropout)	(None, 10, 10, 128)	0
conv2d_10 (Conv2D)	(None, 8, 8, 256)	295168
batch_normalization_10 (Batc	(None, 8, 8, 256)	1024
max_pooling2d_6 (MaxPooling2	(None, 4, 4, 256)	0
dropout_7 (Dropout)	(None, 4, 4, 256)	0
flatten_5 (Flatten)	(None, 4096)	0
dense_10 (Dense)	(None, 4096)	16781312
dropout_8 (Dropout)	(None, 4096)	0
dense_11 (Dense)	(None, 1024)	4195328
dropout_9 (Dropout)	(None, 1024)	0
dense_12 (Dense)	(None, 7)	7175

Total params: 21,356,423
Trainable params: 21,355,527
Non-trainable params: 896

Overview of the solution

Fourth model: Convolutional Neural Network (II)

- Source: A. Gulli *et al.*, "Deep Learning with Tensorflow 2 and Keras", Chapter 4 , Packt, 2019.
- Library: Keras and Tensorflow 2.0
- Architecture: Deep *Convolutional Neural Network* with *Batch Normalization*, *MaxPooling* and *2-layers FC NN*
- Data Augmentation and Dataset Shuffling
- Parameters:
  - Training Epochs: 50
  - Batch size: 64
  - Activation Function: ReLU
  - Optimizer: Adam with adaptive LR
  - Dataset split: 80% train - 10% validation - 10% test
- Accuracy: **90% on test → Highest** **CHOSEN MODEL!**

Loop Q PRIZE

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	896
batch_normalization (BatchNo	(None, 48, 48, 32)	128
conv2d_1 (Conv2D)	(None, 48, 48, 32)	9248
batch_normalization_1 (Batch	(None, 48, 48, 32)	128
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18496
batch_normalization_2 (Batch	(None, 24, 24, 64)	256
conv2d_3 (Conv2D)	(None, 24, 24, 64)	36928
batch_normalization_3 (Batch	(None, 24, 24, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 64)	0
conv2d_4 (Conv2D)	(None, 12, 12, 128)	73856
batch_normalization_4 (Batch	(None, 12, 12, 128)	512
conv2d_5 (Conv2D)	(None, 12, 12, 128)	147584
batch_normalization_5 (Batch	(None, 12, 12, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 128)	0
conv2d_6 (Conv2D)	(None, 6, 6, 256)	295168
batch_normalization_6 (Batch	(None, 6, 6, 256)	1024
conv2d_7 (Conv2D)	(None, 6, 6, 256)	590080
batch_normalization_7 (Batch	(None, 6, 6, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 2304)	5310720
dropout (Dropout)	(None, 2304)	0
dense_1 (Dense)	(None, 7)	16135

Total params: 6,502,951
Trainable params: 6,501,031
Non-trainable params: 1,920

Overview of the solution

Fifth model: Transfer Learning of a prebuilt CNN

- Source: https://www.kaggle.com/yasserhessein/emotion-recognition-with-efficientnetb0
- Library: Keras and Tensorflow 2.0
- Architecture: *EfficientNetB0* (pretrained on *ImageNet* dataset) and 2-layer *FC NN* with *Dropout*
- Data Augmentation and Dataset Shuffling
- Parameters:
  - Training Epochs: 50
  - Batch size: 64
  - Activation Function: ReLU
  - Optimizer: Adam with adaptive LR
  - Dataset split: 80% train - 10% validation - 10% test
- Accuracy: **85% on test**

Layer (type)	Output Shape	Param #
efficientnetb0 (Functional)	(None, 2, 2, 1280)	4049571
flatten_3 (Flatten)	(None, 5120)	0
dense_6 (Dense)	(None, 5120)	26219520
dropout_3 (Dropout)	(None, 5120)	0
dense_7 (Dense)	(None, 7)	35847

Total params: 30,304,938
Trainable params: 30,262,915
Non-trainable params: 42,023

Strengths of the solution

- Very high accuracy for the chosen Model (~90%)
- Usage of CNN allows to automatically detect the most relevant features during the *training* of the Model
- CNN faster to train than SVM
- Implementation of CNN models requires less (pre)knowledge about Image Processing

Limits and bias

- Unbalanced dataset for certain classes (see slide 4)
- Images contain only faces (no face detection required) but sometimes with low quality
- Some images do not contain faces (e.g. error text)
- Images placed into a directory on disk do not consent fast reading from it in batches → slow training
- Training times decreased through the loading of the entire dataset on RAM/GPU → scalability limitations in case of eventual wider datasets

Thank you!