ADVANCED DATA SCIENCE PROJECT

AUTOMOTIVE SEMICONDUCTOR INDUSTRY _

PICTURE CLASSIFIER IN AUTOMATIC OPTICAL INSPECTION (AOI) AFTER SAWING

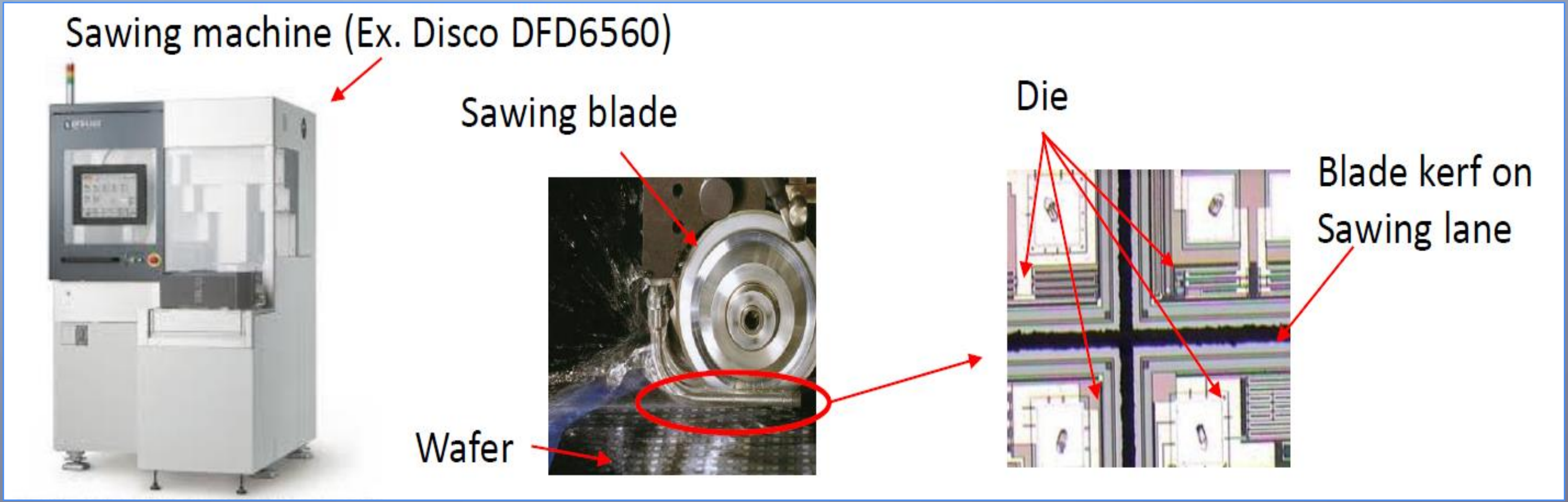
APRIL 22, 2020

USE CASE

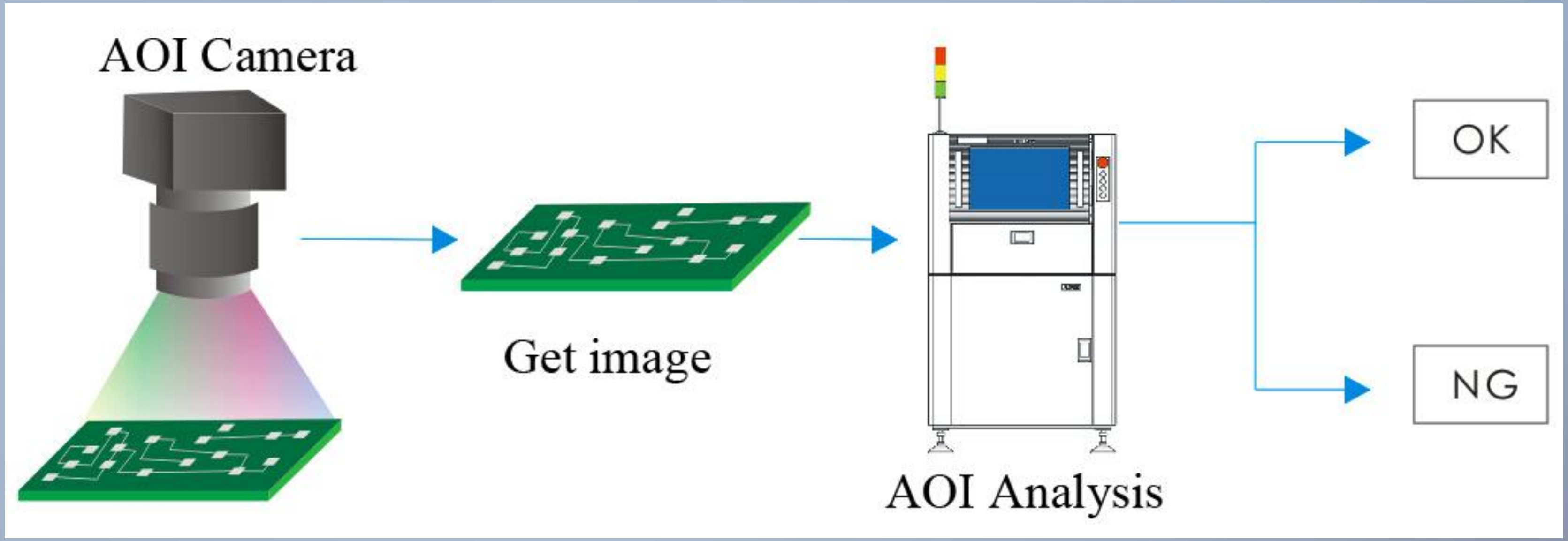
- Context: Automotive Semiconductor Industry
- Plant type: Assembly and Test Site (4 sites targeted: Kuala Lumpur, Kaohsiung, Bangkok, Tianjin)
- Project type: Machine Learning Project on Big Data
- Scope: Automatic Optical Inspection (AOI) after sawing
- Goal: picture classifier for defects due to sawing process
- Model pilot: because of a non-disclosure constraint, picture classifier is presented with public data

SAWING PROCESS

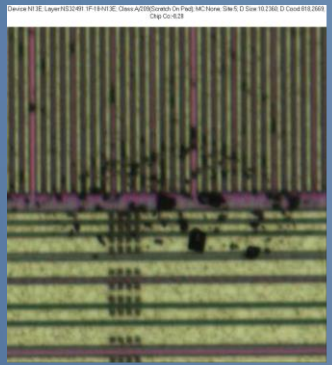
- Die sawing process generates defects on dies (chipping)
- Damaged parts have to be detected and removed from assembly and test lines

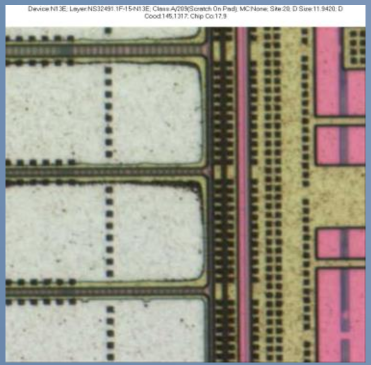


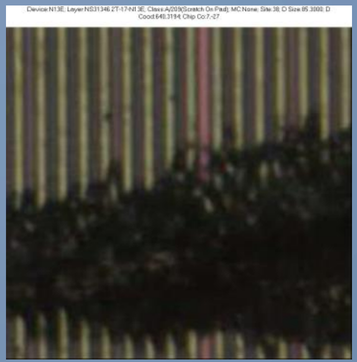
AUTOMATIC OPTICAL INSPECTION

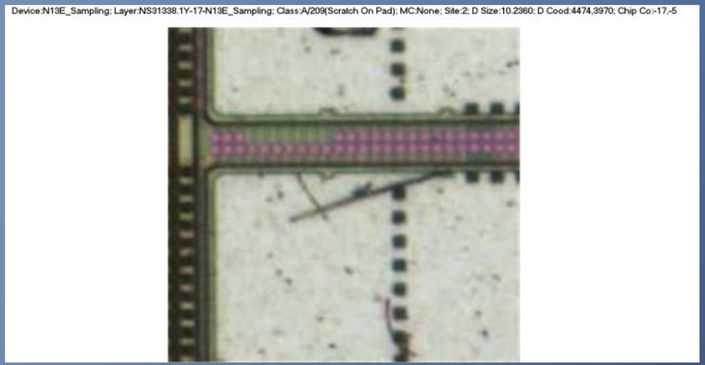


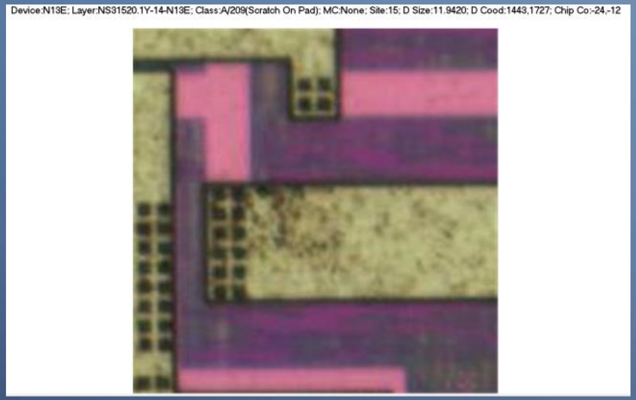
DEFECT PICTURES

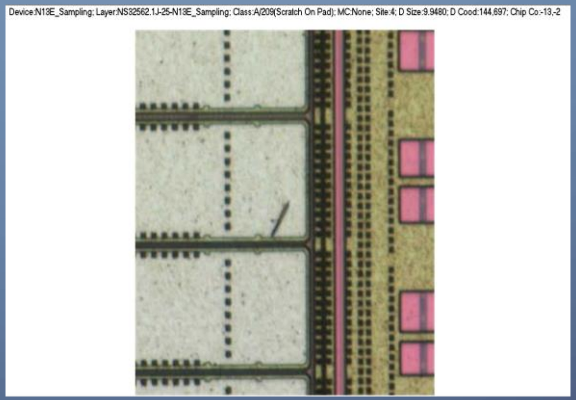


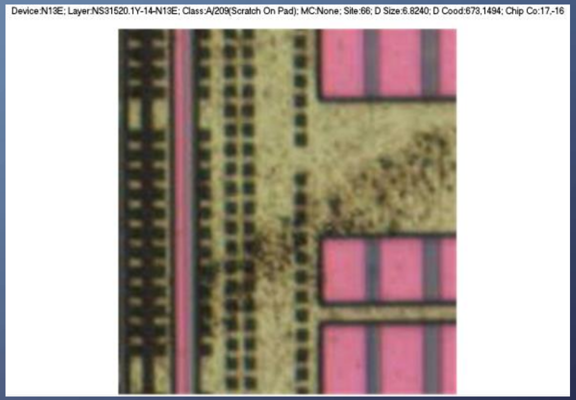


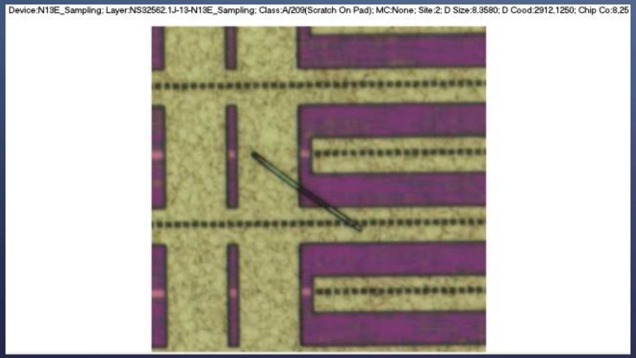


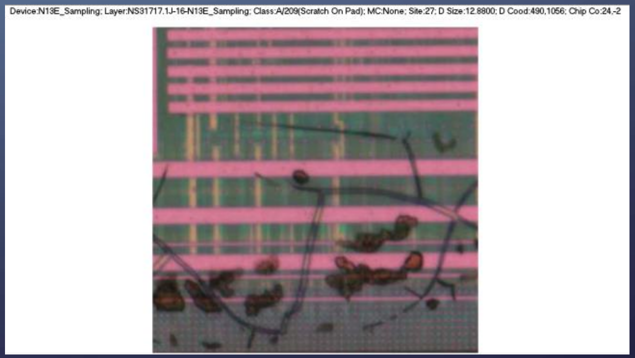






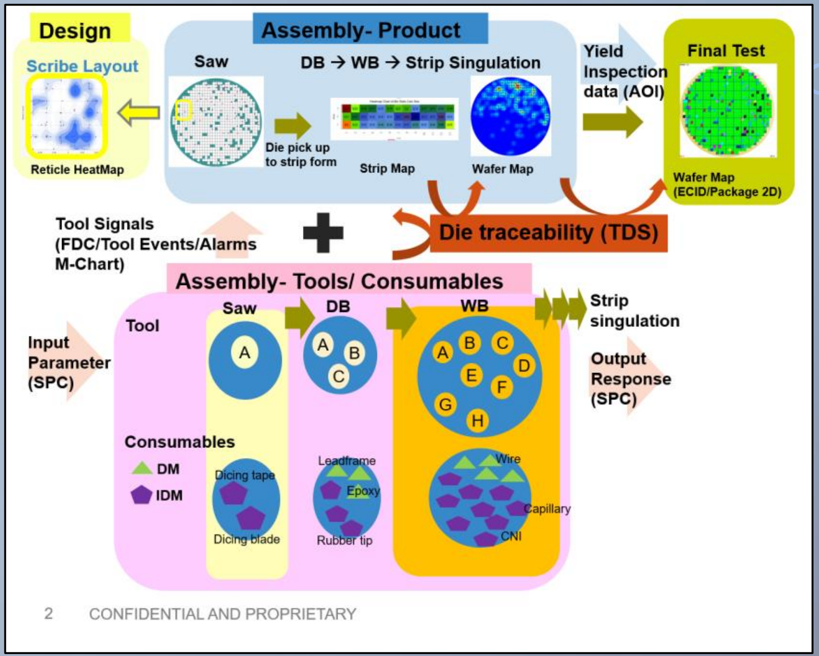


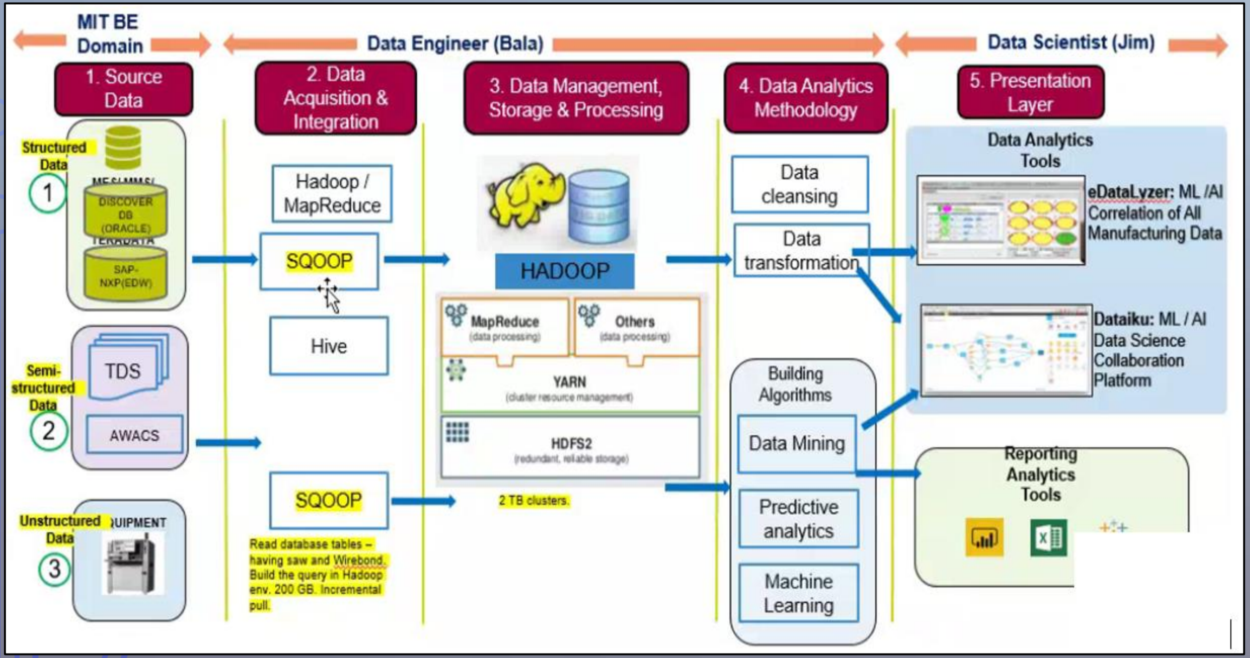


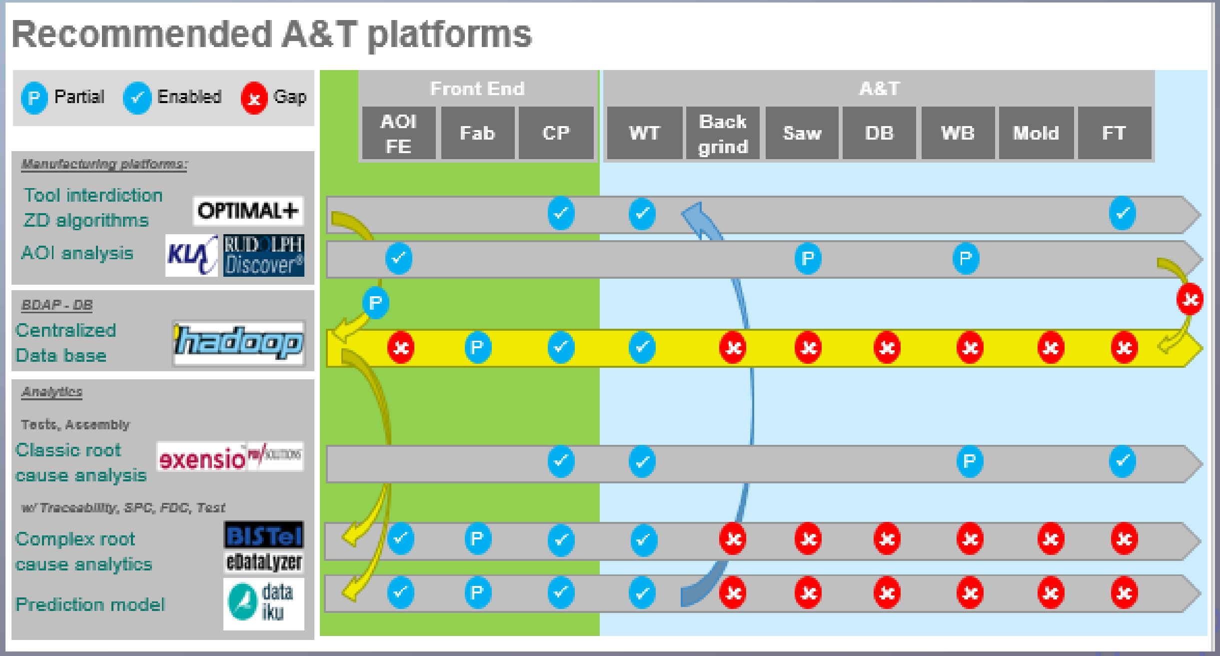


PROJECT ARCHITECTURE OVERVIEW

- Data type, data sources
- Databases, big data environment (Hadoop)
- Data Analytic platforms and tools

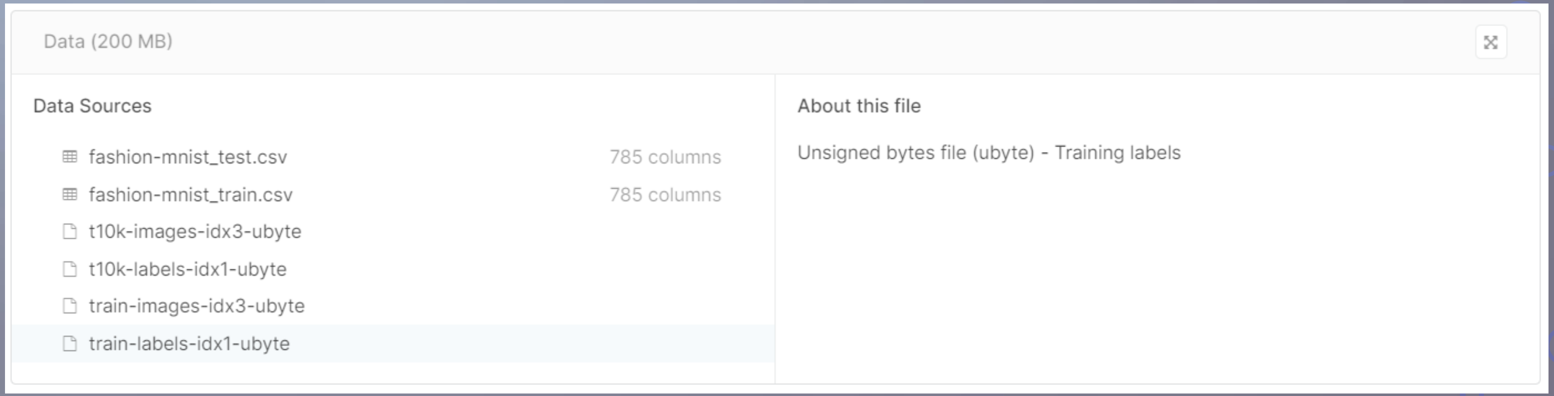






PILOT

- Data used: because of non-disclosure constraint, defect pictures are replaced by public pictures
- Data source: https://www.kaggle.com/zalando-research/fashionmnist#train-labels-idx1-ubyte
- Library: Keras

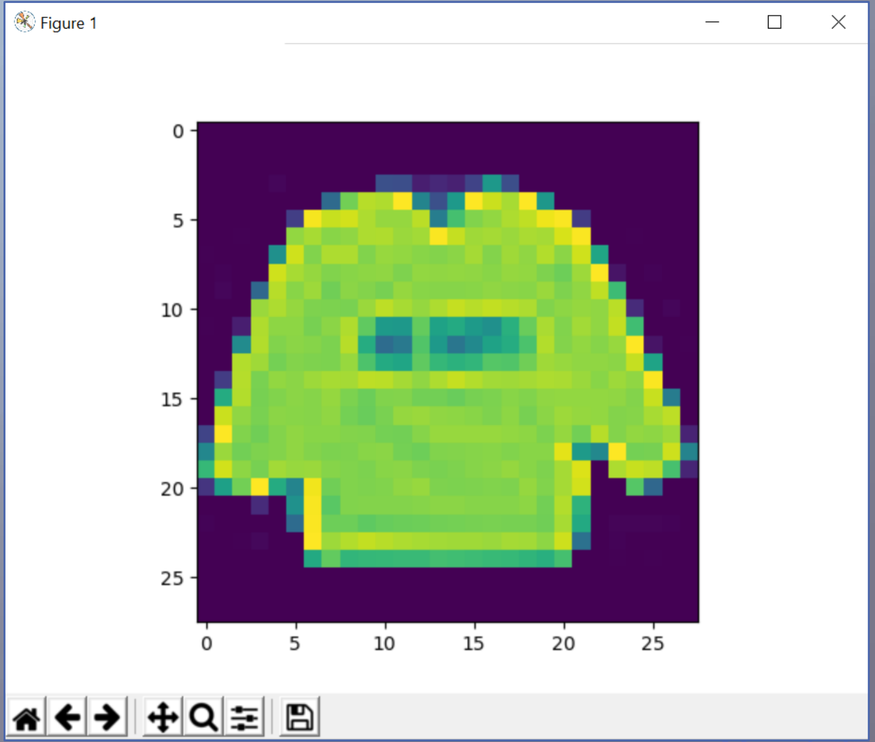


DATA PREPARATION

- What are the data for this project ?
- Features used: picture pixels

```
import pandas as pnd
import numpy as np
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split

#Définition de la longueur et de la largeur de l'image
LONGUEUR_IMAGE = 28
LARGEUR_IMAGE = 28

#Chargement des données d'entrainement
observations_entrainement = pnd.read_csv('C:/Users/nxa13794/PycharmProjects/Picture classifier/datas/zalando/fashion-mnist_train.csv')

#On ne garde que les feature "pixels"
X = np.array(observations_entrainement.iloc[:, 1:])

from matplotlib import pyplot as plt
premiereImage=X[0]
premiereImage=premiereImage.reshape([LONGUEUR_IMAGE, LARGEUR_IMAGE]);
plt.imshow(premiereImage)
plt.show()
```

Picture Label	Picture (French)	Picture (English)
0	T-shirt	T-shirt
1	Pantalon	Trousers
2	Pull	Pull
3	Robe	Dress
4	Manteau	Coat
5	Sandales	Sandals
6	Chemise	Shirt
7	Baskets	Sneakers
8	Sac	Bag
9	Bottes	Boots



TRAINING/VALIDATION DATA PREPARATION

- Training Data: 80%
- Validation Data: 20%

```
#On crée un tableau de catégories à l'aide du module Keras
y = to_categorical(np.array(observations_entrainement.iloc[:, 0]))

#Répartition des données d'entrainement en données d'apprentissage et donnée de validation
#80% de donnée d'apprentissage et 20% de donnée de validation
X_apprentissage, X_validation, y_apprentissage, y_validation = train_test_split(X, y, test_size=0.2, random_state=13)



# On redimensionne les images au format 28*28 et on réalise un scaling sur les données des pixels
X_apprentissage = X_apprentissage.reshape(X_apprentissage.shape[0], LARGEUR_IMAGE, LONGUEUR_IMAGE, 1)
X_apprentissage = X_apprentissage.astype('float32')
X_apprentissage /= 255

# On fait la même chose avec les données de validation
X_validation = X_validation.reshape(X_validation.shape[0], LARGEUR_IMAGE, LONGUEUR_IMAGE, 1)
X_validation = X_validation.astype('float32')
X_validation /= 255
```

TEST DATA PREPARATION

- Training Data: 80%
- Validation Data: 20%

```
#Preparation des données de test
observations_test = pnd.read_csv('C:/Users/nxa13794/PycharmProjects/Picture classifier/datas/zalando/fashion-mnist_test.csv')

X_test = np.array(observations_test.iloc[:, 1:])
y_test = to_categorical(np.array(observations_test.iloc[:, 0]))

X_test = X_test.reshape(X_test.shape[0], LARGEUR_IMAGE, LONGUEUR_IMAGE, 1)
X_test = X_test.astype('float32')
X_test /= 255
```

CONVOLUTION MODEL #1

```python
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D

#On spécifie les dimensions de l'image d'entree
dimentionImage = (LARGEUR_IMAGE, LONGUEUR_IMAGE, 1)
```

```python
#On crée le réseau de neurones couche par couche
reseauNeurone1Convolution = Sequential()

#1- Ajout de la couche de convolution comportant
#  Couche cachée de 32 neurones
#  Un filtre de 3x3 (Kernel) parourant l'image
#  Une fonction d'activation de type ReLU (Rectified Linear Activation)
#  Une image d'entrée de 28px * 28 px
reseauNeurone1Convolution.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=dimentionImage))

#2- Définition de la fonction de pooling avec un filtre de 2px sur 2 px
reseauNeurone1Convolution.add(MaxPooling2D(pool_size=(2, 2)))

#3- Ajout d'une fonction d'ignorance
reseauNeurone1Convolution.add(Dropout(0.2))

#5 - On transforme en une seule ligne
reseauNeurone1Convolution.add(Flatten())

#6 - Ajout d'un reseau de neuronne composé de 128 neurones avec une fonction d'activation de type Relu
reseauNeurone1Convolution.add(Dense(128, activation='relu'))

#7 - Ajout d'un reseau de neuronne composé de 10 neurones avec une fonction d'activation de type softmax
reseauNeurone1Convolution.add(Dense(10, activation='softmax'))
```

COMPILING, TRAINING AND TEST

```
#8 - Compilation du modèle
import keras
reseauNeurone1Convolution.compile(loss=keras.losses.categorical_crossentropy,
                                  optimizer=keras.optimizers.Adam(),
                                   metrics=['accuracy'])



#9 - Apprentissage
historique_apprentissage  = reseauNeurone1Convolution.fit(X_apprentissage, y_apprentissage,
           batch_size=256,
           epochs=10,
           verbose=1,
           validation_data=(X_validation, y_validation))


#10 - Evaluation du modèle
evaluation = reseauNeurone1Convolution.evaluate(X_test, y_test, verbose=0)
print('Erreur :', evaluation[0])
print('Précision:', evaluation[1])
```

```
Erreur : 0.2601269866406918
Précision: 0.9045000076293945
```

Model evaluation

CONVOLUTION MODEL #2

- Same model features
- Images added by automatic generation (ImageDataGenerator in Keras module)
- Model saved in modele.h5

```python
from keras.preprocessing.image import ImageDataGenerator
generateur_images = ImageDataGenerator(rotation_range=8,
                          width_shift_range=0.08,
                          shear_range=0.3,
                          height_shift_range=0.08,
                          zoom_range=0.08)


nouvelles_images_apprentissage = generateur_images.flow(X_apprentissage, y_apprentissage, batch_size=256)
nouvelles_images_validation = generateur_images.flow(X_validation, y_validation, batch_size=256)


#10 - Apprentissage
historique_apprentissage = reseauNeurone1Convolution.fit_generator(nouvelles_images_apprentissage,
                                             steps_per_epoch=48000//256,
                                             epochs=50,
                                             validation_data=nouvelles_images_validation,
                                             validation_steps=12000//256,
                                             use_multiprocessing=False,
                                             verbose=1 )
```

```python
# serialize model to JSON
model_json = reseauNeurone1Convolution.to_json()
with open("modele/modele.json", "w") as json_file:
    json_file.write(model_json)

# serialize weights to HDF5
reseauNeurone1Convolution.save_weights("modele/modele.h5")
```

```
Erreur : 0.24349224190711974
Précision: 0.9128000140190125
```

CONVOLUTION MODEL #4

- 4 layers

```
reseauNeurones4Convolution = Sequential()
reseauNeurones4Convolution.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=dimentionImage))
reseauNeurones4Convolution.add(BatchNormalization())

reseauNeurones4Convolution.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
reseauNeurones4Convolution.add(BatchNormalization())
reseauNeurones4Convolution.add(MaxPooling2D(pool_size=(2, 2)))
reseauNeurones4Convolution.add(Dropout(0.25))

reseauNeurones4Convolution.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
reseauNeurones4Convolution.add(BatchNormalization())
reseauNeurones4Convolution.add(Dropout(0.25))

reseauNeurones4Convolution.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
reseauNeurones4Convolution.add(BatchNormalization())
reseauNeurones4Convolution.add(MaxPooling2D(pool_size=(2, 2)))
reseauNeurones4Convolution.add(Dropout(0.25))
```

Model evaluation:
Erreur: 0.19347
Précision: 0.9302

Cost for this precision is time needed to train the model

```
reseauNeurones4Convolution.add(Flatten())

reseauNeurones4Convolution.add(Dense(512, activation='relu'))
reseauNeurones4Convolution.add(BatchNormalization())
reseauNeurones4Convolution.add(Dropout(0.5))

reseauNeurones4Convolution.add(Dense(128, activation='relu'))
reseauNeurones4Convolution.add(BatchNormalization())
reseauNeurones4Convolution.add(Dropout(0.5))

reseauNeurones4Convolution.add(Dense(10, activation='softmax'))
```

PICTURE CLASSIFIER

```
#Récuperation des pixels
pixels = list(nouvelleImage.getdata())

#Normalisation des pixels
tableau = [(255 - x) * 1.0 / 255.0 for x in pixels]

import numpy as np
#Transformation du tableau en tableau numpy
img = np.array(tableau)

#On transforme le tableau linéaire en image 28x20
image_test = img.reshape(1, 28, 28, 1)

prediction = modele.predict_classes(image_test)
print()
print("Selon moi l'image est : "+classes[prediction[0]])
print()
```

MODEL RUNNING ON NEW IMAGES

- Picture #1

```
Selon moi l'image est : Une baskets
```

7	Baskets	Sneakers

```
Un T-shirt/haut: 0.004183905548416078%
Un pantalon: 0.0032025134714785963%
Un pull: 0.0010191218279942404%
Une robe: 0.0022330257706926204%
Un manteau: 0.0022629059458267875%
Une sandale: 10.583294183015823%
Une chemise: 0.00271360495389672%
Une baskets: 89.34122920036316%
Un sac: 0.016235857037827373%
Une botte de cheville: 0.04363248881418258%
```

- Picture #2

```
Selon moi l'image est : Un sac
```

8	Sac	Bag

```
Un T-shirt/haut: 0.0035013799788430333%
Un pantalon: 0.0003859966000163695%
Un pull: 0.001080885067494819%
Une robe: 0.003076469511142932%
Un manteau: 0.006730201857862994%
Une sandale: 0.004474401066545397%
Une chemise: 0.0031596871849615127%
Une baskets: 0.0013574815056927036%
Un sac: 99.97579455375671%
Une botte de cheville: 0.00044567723307409324%
```

- Picture #3

```
Selon moi l'image est : Un manteau
```

4	Manteau	Coat

```
Un T-shirt/haut: 0.34571674186736345%
Un pantalon: 0.006158505129860714%
Un pull: 3.25436107814312%
Une robe: 0.06413143128156662%
Un manteau: 93.86534690856934%
Une sandale: 0.005009111919207498%
Une chemise: 2.351035736501217%
Une baskets: 0.0015677407645853236%
Un sac: 0.10374020785093307%
Une botte de cheville: 0.002930087794084102%
```

CONCLUSION

- Opportunities to improve model including more images and using more layers
- But, cost in term of time needed in training or in term of material if we want to decrease time needed

THANK YOU FOR YOUR ATTENTION