GROUPE 11

PROJET ANNUEL ESGI 2020
MACHINE LEARNING

Application de reconnaissance de marque de voiture

ESGI
école supérieure de génie informatique

INTRODUCTION

- **Rappel du sujet :** « Implémenter et utiliser des modèles et algorithmes simples relatifs au Machine Learning, combiner le tout dans un cas pratique réel »

- **Enjeux du projet :** commenter et analyser les résultats obtenus et porter un regard critique sur les outils.

SOMMAIRE

- Rappel de la problématique applicative choisie
- Ambiguïté et complexité du sujet
- Technologies utilisées
- Dataset : Génération, analyse des données et préparation des données
- Entraînement des modèles
- Démonstration live
- Difficultés

RAPPEL DE LA PROBLÉMATIQUE

Car recognition app

- Différencier des voitures de différentes marques à partir d'une photo
- Usage business et public

COMPLEXITÉ DU SUJET

- Enormément de marque de voitures, de modèles, de séries, de couleurs
- Pas de dataset préexistant de photos de voitures étiquetées par marque
- Angles de prises de photos multiples

COMPLEXITÉ DU SUJET

COMPLEXITÉ DU SUJET

TECHNOLOGIES UTILISÉES

Machine learning


Rust Programming Language


Python 3.8


Jupyter Notebook

Tensorflow 2.2


Keras

Application


Angular 7

Flask 1.1

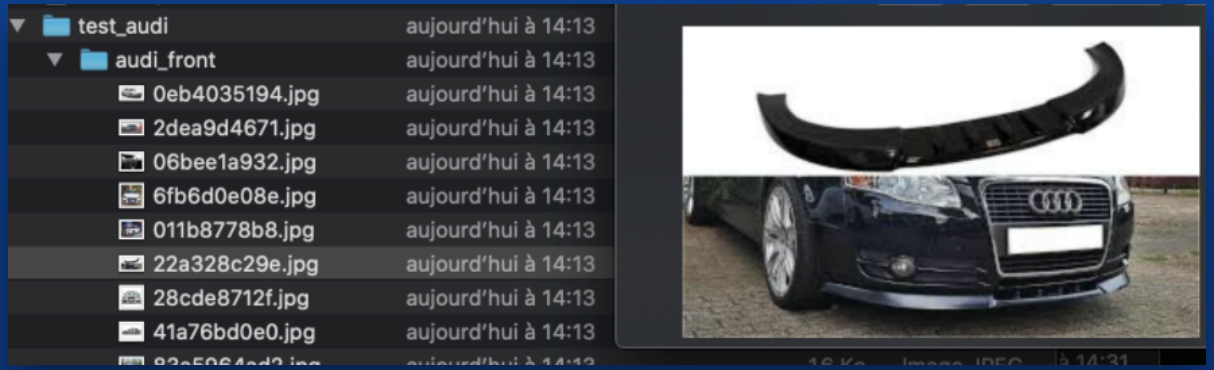

Git


SCSS

DATASET

Génération du dataset

Première approche : scraping Google Image (« audi front »)



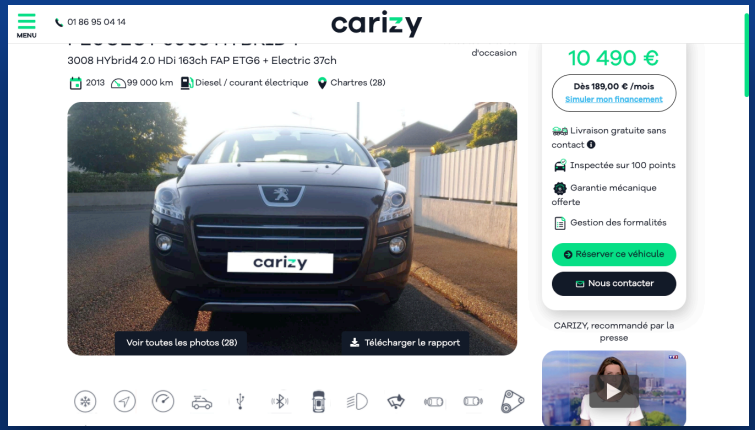
Résultats de mauvaise qualité

Beaucoup de tri

DATASET

Génération du dataset

Seconde approche: scraping CDN d'un site d'annonce automobile (https://carizy.com)



Aucun tri

Formalisation

DATASET

Génération du dataset

```
#
# Define a car data class that retrieve the useful data
#

class CarData:
    def __init__(self, car_id, url, img):
        self.car_id = car_id
        self.url = url
        self.img = img

    def __repr__(self):
        return str(self.car_id) + " : " + self.url + ' + ' + self.img

#
# Request the search api of the website with good params
#

def send_algolia_request(page):
    url = 'https://65quub6n61-dsn.algolia.net/1/indexes/*/queries?x-algolia-
agent=Algolia%20for%20vanilla%20JavaScript%20(lite)%203.18.1%3Binstantsearch.js%201.8.11&x-algolia-application-
id=65QUUB6N61&x-algolia-api-key=9ce4fd3eb414de32d0985b9b5a394448'
    data = {'requests': [{'indexName' : 'CarAlgoliaIndex_prod', 'params':
'query=&hitsPerPage=21&maxValuesPerFacet=100&page='+ str(page)
+'&facets=%5B%22make%22%2C%22model%22%2C%22body%22%2C%22registartionDate%22%2C%22color%22%2C%22gear%22%2C%22fuel%22
%2C%22price%22%2C%22monthly%22%2C%22kilometers%22%2C%22district%22%2C%22districtName%22%2C%22city%22%2C%22seats%22%
2C%22isScrapPremium%22%5D&tagFilters='}]}

    p = requests.post(url, json=data)
    return p.json()
```

DATASET

Génération du dataset

```
#
# map data to keep only useful data
#

def map_data(_car_data, hits):
    for e in hits:
        if e['make'] in _car_data:
            _car_data[e['make']].append(CarData(e['carId'], e['lien'], e['photo']))
        else:
            _car_data[e['make']] = [CarData(e['carId'], e['lien'], e['photo'])]

#
# Make all the request with a interval to avoid api blocking
#

car_data = {}
i = 0
while i < 48:
    x = send_algolia_request(i)
    map_data(car_data, x['results'][0]['hits'])
    i += 1
    time.sleep(1)
```

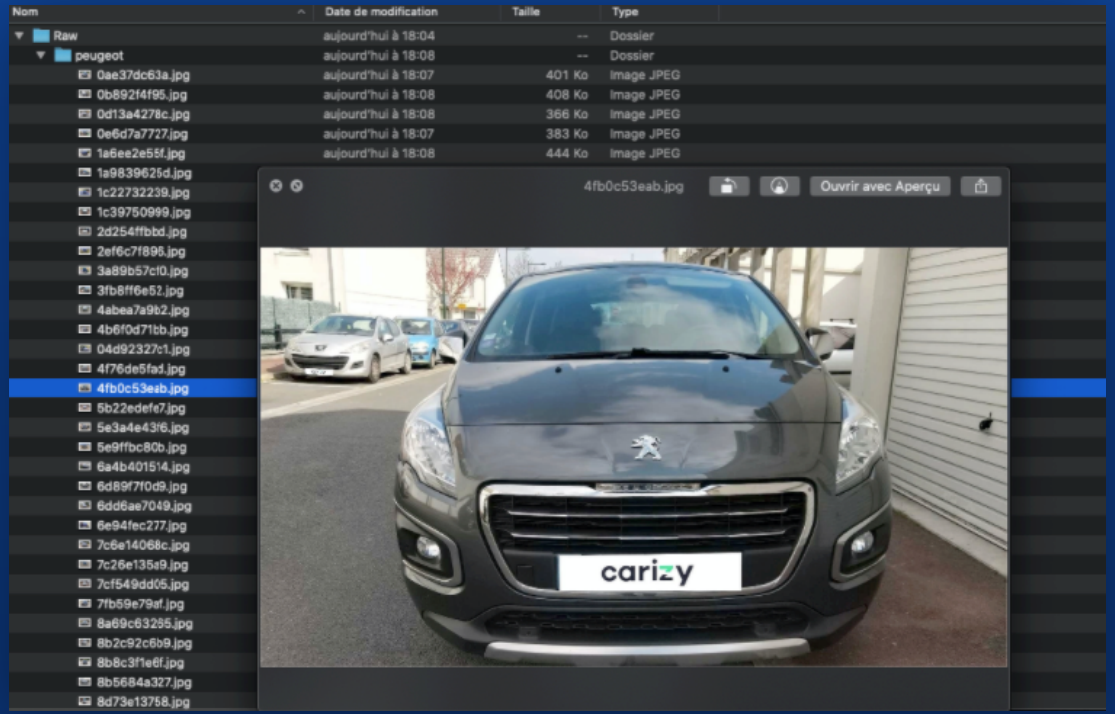
DATASET

Génération du dataset

```
#
# map data to keep only useful data
#

def map_data(_car_data, hits):
    for e in hits:
        if e['make'] in _car_data:
            _car_data[e['make']].append(CarData(e['carId'], e['lien'], e['photo']))
        else:
            _car_data[e['make']] = [CarData(e['carId'], e['lien'], e['photo'])]

#
# Make all the request with a interval to avoid api blocking
#

car_data = {}
i = 0
while i < 48:
    x = send_algolia_request(i)
    map_data(car_data, x['results'][0]['hits'])
    i += 1
    time.sleep(1)
```

DATASET

Génération du dataset



DATASET

Analyse de la donnée

Images de dimension : **1080** x **1920**

Colorimétrie : **RGB**

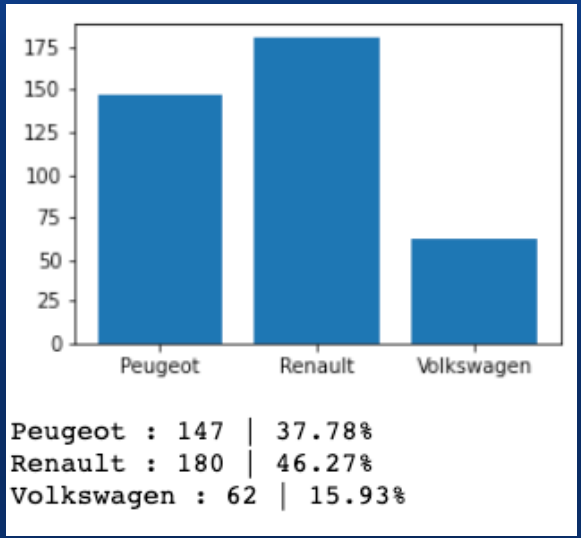
Vecteur de taille 1080 x 1920 x 3 = **6220800**

RGB valeur maximum : **255**

RGB valeur minimum : **0**

DATASET

Analyse de la donnée



Dataset déséquilibré

Nombres d'exemples limité

Les modèles doivent dépasser 46% de fiabilité

20% de set de validation

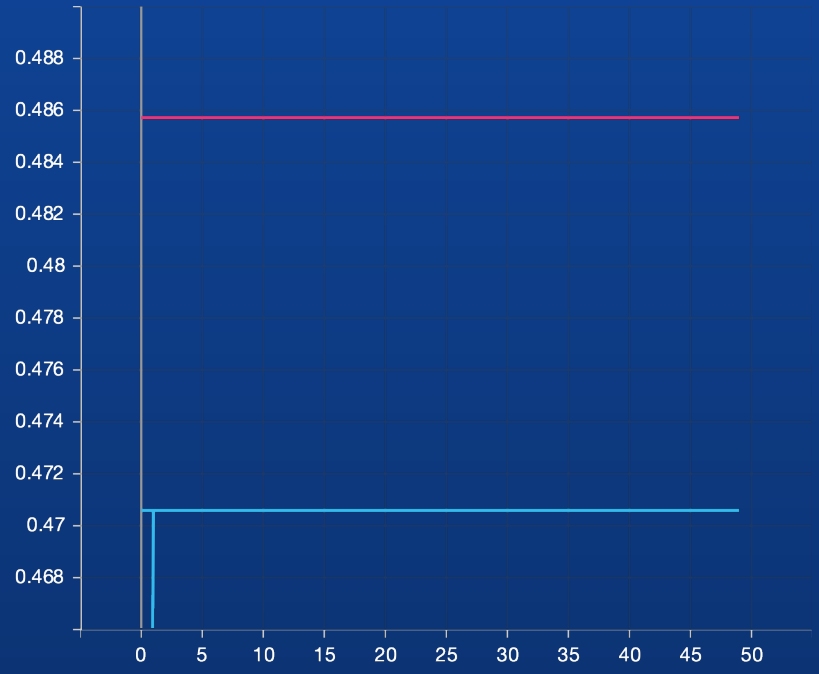
DATASET

Analyse de la donnée

```
train_dir = "../../Dataset/Train"
validation_dir = "../../Dataset/Validation"
target_width = 128
target_height = 128
color_mode = 'rgb'
target_size = (target_width, target_height)
image_generator = ImageDataGenerator(rescale=1./255)

# Set train data
train_data = image_generator.flow_from_directory(
        train_dir,
        target_size=target_size,
        color_mode=color_mode,
    )
#set Validation data
validation_data = image_generator.flow_from_directory(
    validation_dir,
    target_size=target_size,
    color_mode=color_mode
)

train_data.batch_size = train_data.n
(x, y) = train_data.next()
train_true_label = np.argmax(y, axis=1)
```

ENTRAÎNEMENT DES MODÈLES

Classification linéaire



ENTRAÎNEMENT DES MODÈLES

Classification linéaire

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	0	110	0
Renault	0	136	0
Volkswagen	0	43	0

Matrice de confusion **train**

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	0	27	0
Renault	0	34	0
Volkswagen	0	9	0

Matrice de confusion **validation**

ENTRAÎNEMENT DES MODÈLES

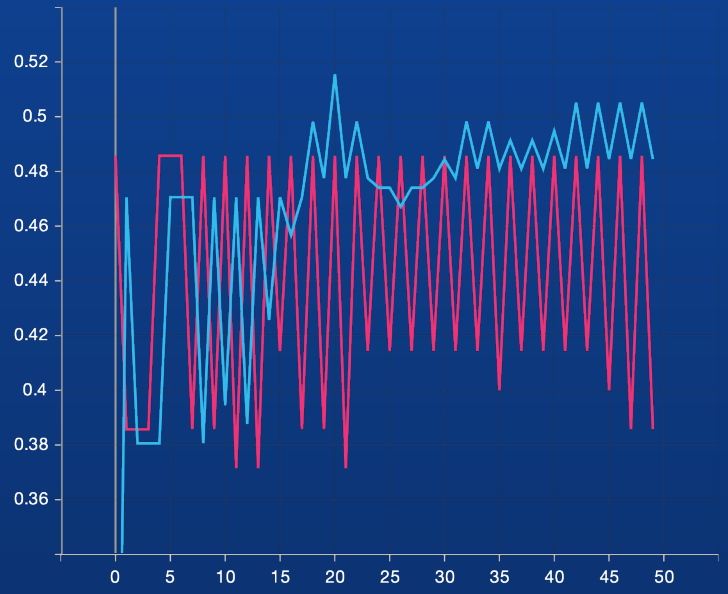
validation

train

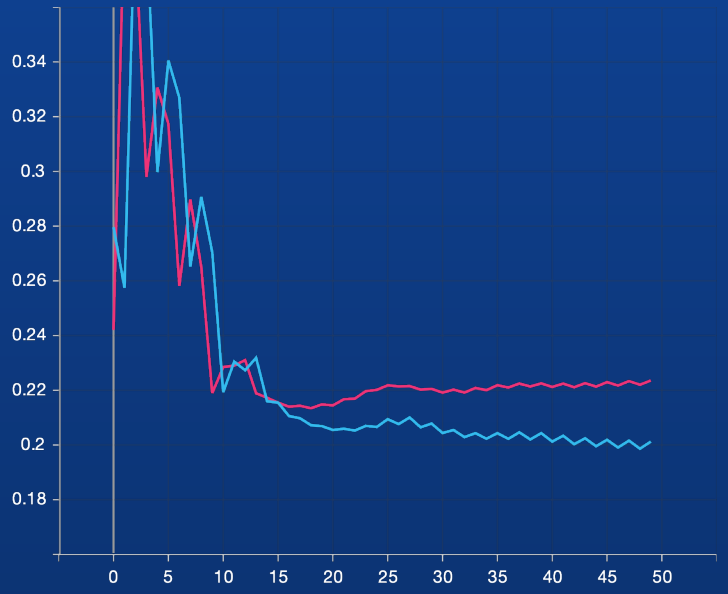
Perceptron Multi Couche

RGB – SGD – Mean Squared Error

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

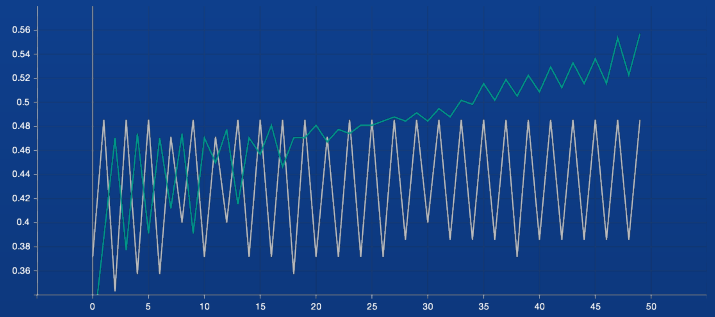
validation

train

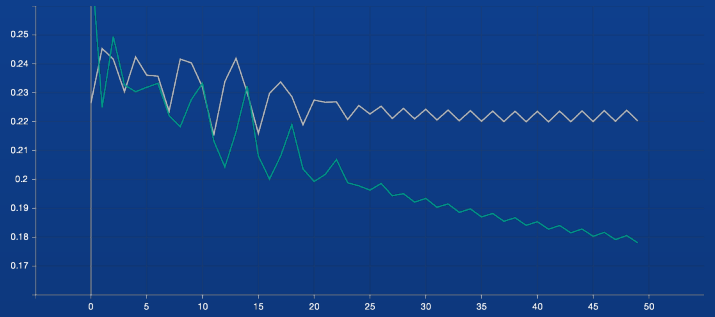
Perceptron Multi Couche

Greyscale – SGD – Mean Squared Error

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

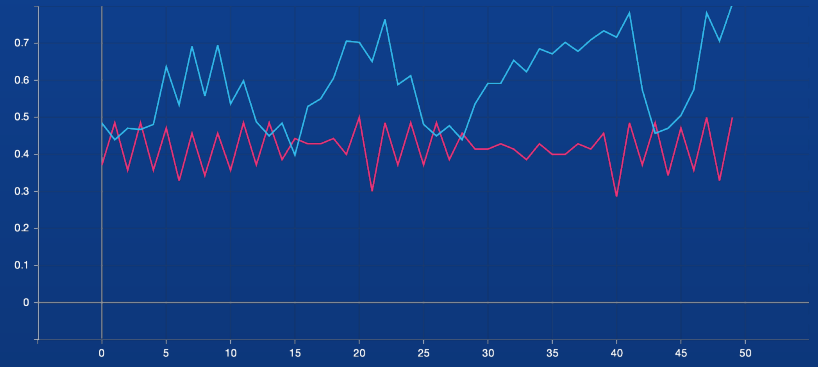
validation

train

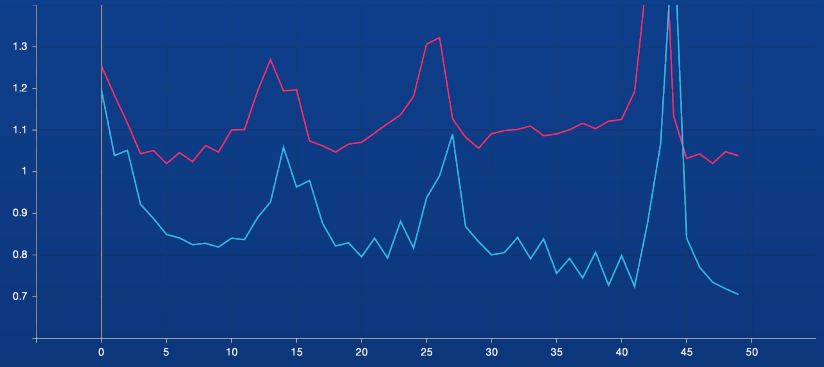
Perceptron Multi Couche

Greyscale – SGD – Categorical crossentropy – batch 32

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

Perceptron Multi Couche

Data augmentation

```
train_dir = "../../Dataset/Train"
validation_dir = "../../Dataset/Validation"
target_width = 128
target_height = 128
color_mode = 'grayscale'
target_size = (target_width, target_height)
image_generator = ImageDataGenerator(rescale=1./255,
                                     zoom_range=0.30,#Zooming
                                     horizontal_flip=True,
                                     rotation_range=10,
                                     width_shift_range=0.2, #Horizontal shifting
                                     height_shift_range=0.2)#vertical shifting

# Set train data
train_data = image_generator.flow_from_directory(
        train_dir,
        target_size=target_size,
        color_mode=color_mode,
        #classes=['peugeot', 'renault', 'volkswagen']
    )
#set Validation data
validation_data = image_generator.flow_from_directory(
    validation_dir,
    target_size=target_size,
    color_mode=color_mode
    #classes=['peugeot', 'renault', 'volkswagen']
)
```

Dataset pas assez performant

Peu d'exemples par rapport au nombre de paramètres d'entrées des réseaux neuronaux

Augmentation du jeu de données

Zoom

Flip horizontal

Rotation

Décalage vertical et horizontal des pixels

ENTRAÎNEMENT DES MODÈLES

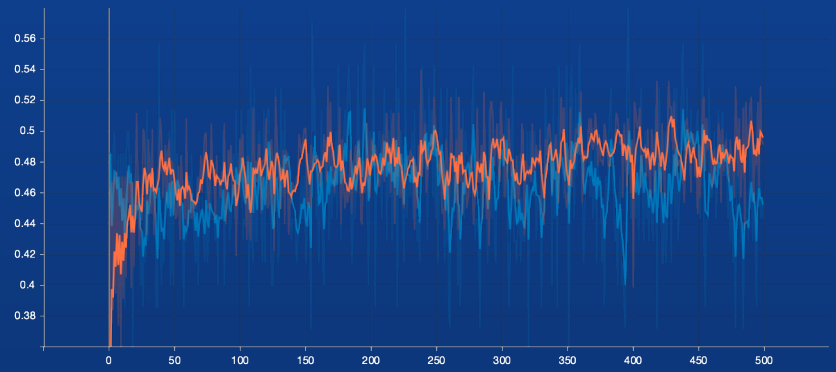
validation

train

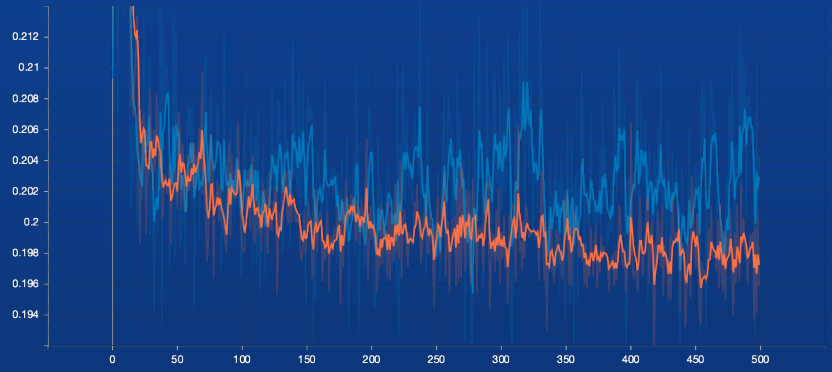
Perceptron Multi Couche

Augmented – SGD – Mean Squared Error

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

Perceptron Multi Couche

Modèle final

```
mlp_model = Sequential()
mlp_model.add(Flatten(input_shape=(128,128, 1)))
mlp_model.add(Dense(512, activation='relu'))
mlp_model.add(Dense(512, activation='relu'))
mlp_model.add(Dense(512, activation='relu'))
mlp_model.add(Dense(3, activation='softmax'))

# save the best model according to validation data set accuracy
checkpoint = ModelCheckpoint('../../App/backend/models/mlp_model.h5', monitor='val_accuracy', verbose=1,
save_best_only=True, mode='max')

# compile the model
mlp_model.compile(optimizer='SGD', loss=tensorflow.keras.losses.mean_squared_error, metrics=['accuracy'])
mlp_model.summary()
```

ENTRAÎNEMENT DES MODÈLES

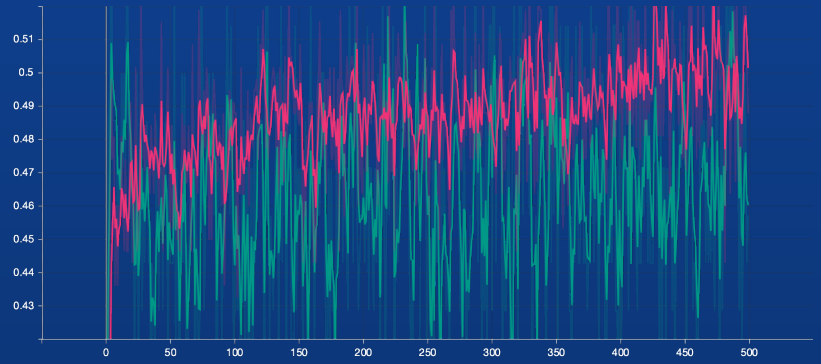
validation

train

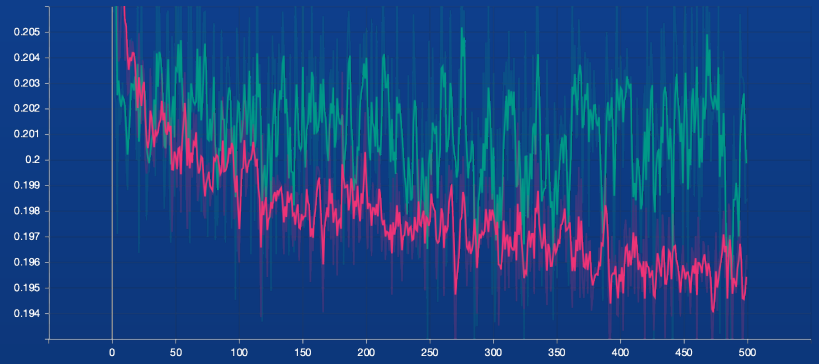
Perceptron Multi Couche

Augmented – relu activation – triple 512 SGD – Mean Squared Error

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

Perceptron Multi Couches

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	8	102	0
Renault	9	127	0
Volkswagen	1	42	0

Train

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	1	26	0
Renault	0	34	0
Volkswagen	1	8	0

Validation

ENTRAÎNEMENT DES MODÈLES

Support Vector Machine

```
svm_model = Sequential()

svm_model.add(Flatten(input_shape=(128,128, 1)))
svm_model.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.01)))# Kernel Regulizer is common
Support Vector Machine Implementation in Keras
svm_model.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
svm_model.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
svm_model.add(Dense(3, activation='softmax'))

# save the best model according to validation data set accuracy
checkpoint = ModelCheckpoint('../../App/backend/models/svm_model.h5', monitor='val_accuracy', verbose=1,
save_best_only=True, mode='max')

# compile the model
svm_model.compile(optimizer='SGD', loss=tensorflow.keras.losses.mean_squared_error, metrics=
['categorical_accuracy', 'accuracy'])
svm_model.summary()
```

ENTRAÎNEMENT DES MODÈLES

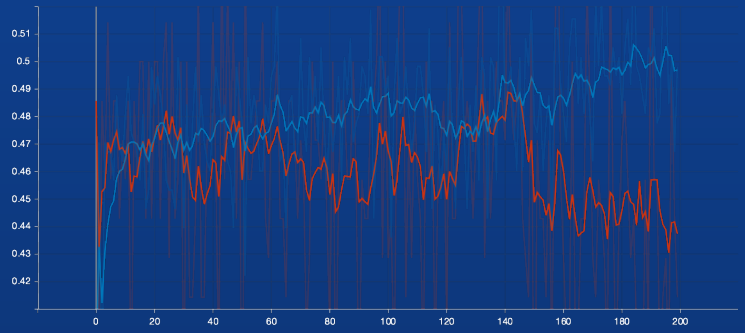
validation

train

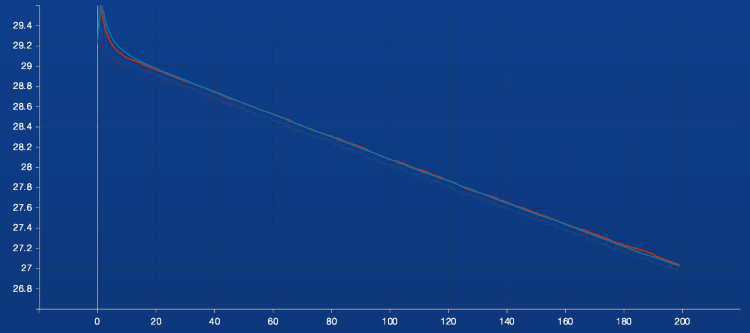
Support Vector Machine

Augmented – relu activation – triple 1024x512x128 SGD – Categorical cross entropy

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

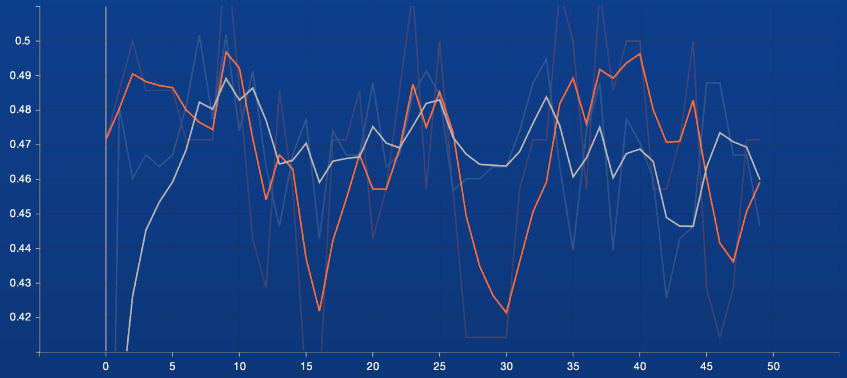
validation

train

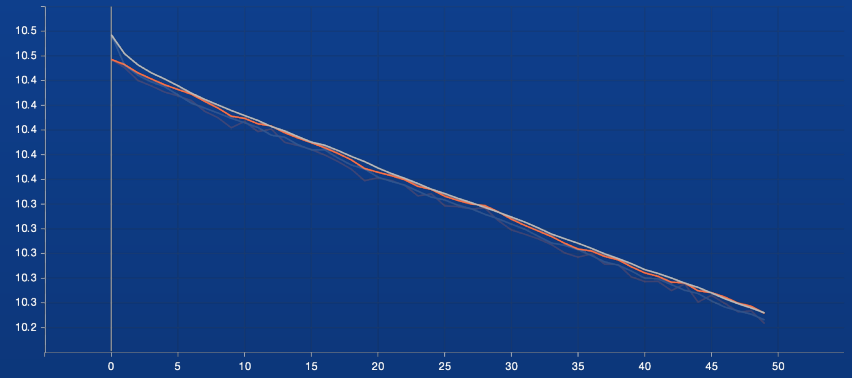
Support Vector Machine

Augmented – relu activation – triple 512 SGD – Mean Squared Error

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

Support Vector Machine

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	25	85	0
Renault	15	121	0
Volkswagen	5	42	0

Train

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	5	22	0
Renault	5	29	0
Volkswagen	2	5	0

Validation

ENTRAÎNEMENT DES MODÈLES

Convolutional Neural network

```
opt = SGD(lr=0.01)

cnn_model = Sequential()
cnn_model.add(Conv2D(filters = 8, kernel_size = (3,3),padding = 'Same', activation ='relu',input_shape=
((128,128,1))))
cnn_model.add(MaxPool2D())

cnn_model.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
cnn_model.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
cnn_model.add(MaxPool2D())


cnn_model.add(Flatten())
cnn_model.add(Dense(3))
cnn_model.add(Activation('softmax'))

# save the best model according to validation data set accuracy
checkpoint = ModelCheckpoint('../../App/backend/models/cnn_model.h5', monitor='val_accuracy', verbose=1,
save_best_only=True, mode='max')

# compile the model
cnn_model.compile(optimizer=opt, loss=tensorflow.keras.losses.categorical_crossentropy, metrics=['accuracy'])
cnn_model.summary()
```

ENTRAÎNEMENT DES MODÈLES

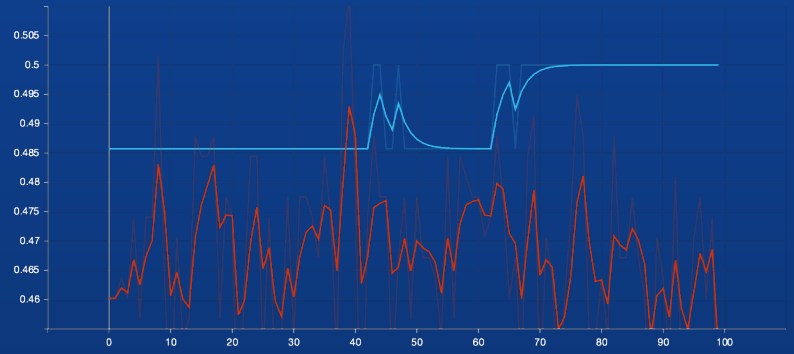
validation

train

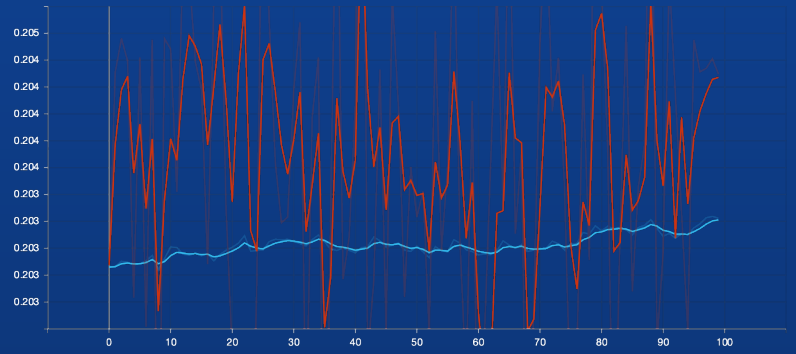
Convolutional Neural Network

Augmented – SGD – activation relu – learning rate 0.001 – Triple Conv

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

validation

train

Convolutional Neural Network

Augmented – SGD – activation relu – learning rate 0.1 – Triple Conv

Accuracy

Loss

ENTRAÎNEMENT DES MODÈLES

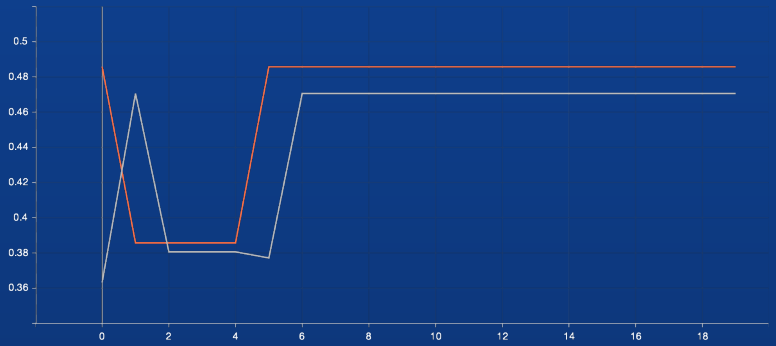
validation

train

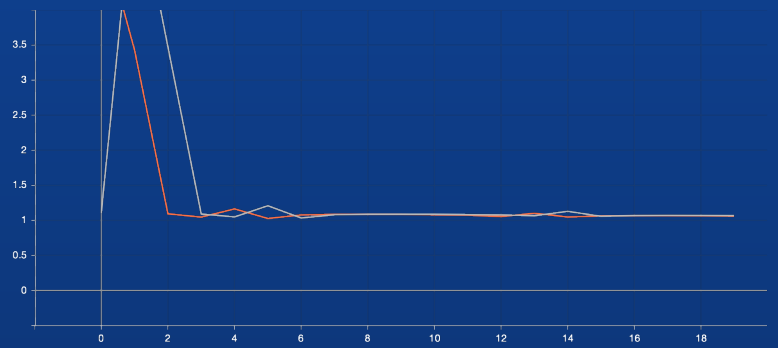
Convolutional Neural Network

Augmented – Adam

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

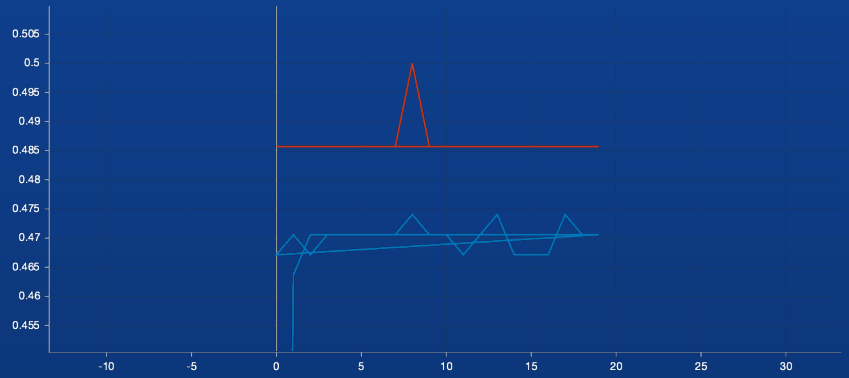
validation

train

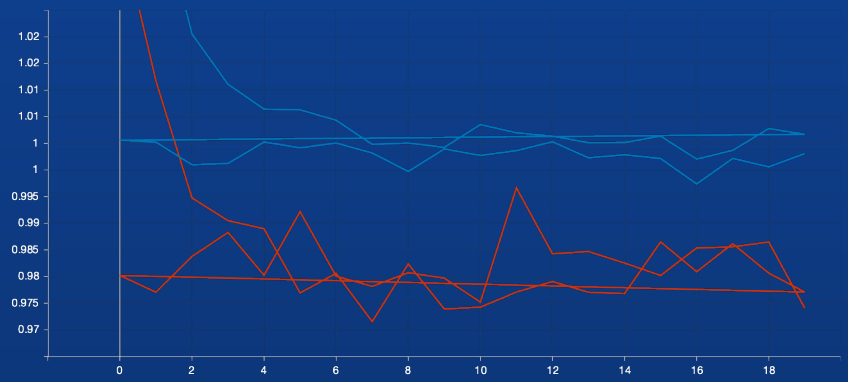
Convolutional Neural Network

Augmented – SGD – activation relu – Categorical cross entropy– learning rate 0.01 – Triple Conv

Accuracy



Loss



ENTRAÎNEMENT DES MODÈLES

Support Vector Machine

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	1	109	0
Renault	1	135	0
Volkswagen	0	43	0

Train

True / predicted	Peugeot	Renault	Volkswagen
Peugeot	0	27	0
Renault	0	34	0
Volkswagen	1	8	0

Validation

ENTRAÎNEMENT DES MODÈLES

46 %

Modèle linéaire

53 %

Support Vector Machine

55 %

Perceptron Multi Couches

50 %

Convolutional Neural Network

Car recognition app

Démonstration live

Client lourd

DIFFICULTÉS

- Problématique applicative complexe & Temps de calcul long sur MAC
- Distanciel très complexe pour les cours et la gestion du groupe => forte baisse de motivation
- Data pas assez normalisée (trop de modèles, finitions, variations)
- Overfitting / Underfitting / Hyper paramètres
- Pas le temps d'intégrer les algorithmes dans une librairies RUST
- Gestion du projet et groupe absent

Bilan

- Évolution sur les problématiques de travail en groupe et autonomie
- Montée en compétence sur Tensorflow et Keras
- Montée en compétence sur la démarche d'analyse de la donnée et d'entraînement des modèles
- Construction d'une application de A à Z
- Manque de temps sur l'implémentation des algorithmes dans une bibliothèque

Merci

pour votre attention

Damien SMAGGHE
3A IABD 1

Damien.smagghe@hotmail.com.fr
Tel : (+33) 6 72 05 70 17
Paris - Rennes