

# ESME SUDRIA

École Spéciale de Mécanique et d'Électricité

PROJET : MASTER 2

Majeur : Intelligence Artificielle

---

## BIOMÉTRIE : RECONNAISSANCE DES VEINES PALMAIRES

---

Présenté par : - Kenan GONNOT  
- Fayçal MERZOUK  
- Lorenzo MARQUEZ

Jury :

**Encadrant : M.BOUBCHIR**  
**Examinateur : M.MAIDI**

**Enseignant**  
**Enseignant**

Année Universitaire : 2022

# TABLE DES MATIÈRES

## **CHAPITRE 1 : Reconnaissance des Veines Palmaires**

Introduction.....
I - La biométrie.....
1. Définition.....
2. Les différentes biométries.....
II - La Reconnaissance des veines palmaires.....
1. Définition.....
2. Histoire.....
3. Description de la modalité des veines palmaires.....
4. Cas d'utilisation.....
5. Acquisition des veines palmaires.....
6. Architecture d'un système de reconnaissance des veines palmaires.....
Conclusion.....

## **CHAPITRE 2 : Machine & Deep Learning**

Introduction.....
I - Machine Learning vs Deep Learning.....
II - Principe de Deep Learning.....
III - Les réseaux de neurones.....
1. ANN.....
2. CNN.....
3. RNN .....
IV - Les différentes architectures du Convolutional Neural Network.....
V - La comparaison.....

## **CHAPITRE 3 : Modélisation et Réalisation**

Introduction.....
I - Une première application .....
1. Les Étapes Clés.....
A. Logiciel.....
B. Librairies.....
C. Dataset.....
D. Modèles .....
2. Architecture de notre première application.....
3. Entrainement de modèles CNN.....
A. Résultat : Semestre 1.....
B. Résultat : Semestre 2.....
II - Application Web : Palm Vein Recognition.....
1. Une partie Backend.....
2. Une partie FrontEnd.....
Conclusion.....

## CHAPITRE 1 : Reconnaissance Des Veines Palmaires

# Introduction

Ce chapitre présente dans un premier temps les différentes types de biométrie. Ensuite on présentera une biométrie centrée sur la reconnaissance des veines palmaires ainsi que ses grandes lignes techniques d'authentification et ses cas d'utilisation.

## I - La Biométrie

### **1. Définition**

Qu'est-ce que la biométrie ?

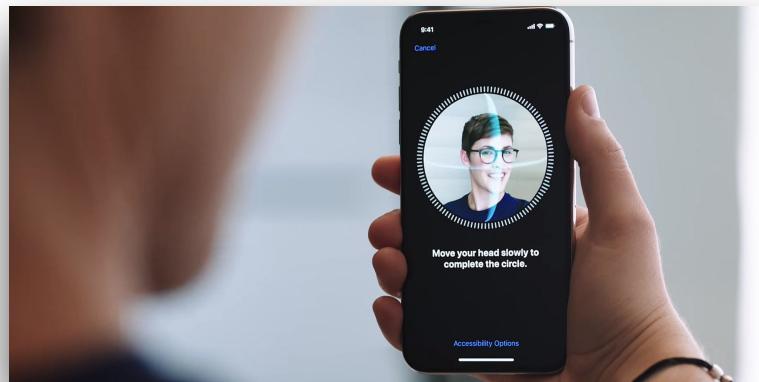
La biométrie est la science de la mesure et de l'analyse des caractéristiques physiologiques et / ou comportementales des êtres humains à des fins de reconnaissance automatique. Offrant un moyen plus pratique d'identifier un individu que les mots de passe ou les badges, les systèmes de sécurité biométriques tels que les passeports électroniques ou les appareils mobiles biométriques sont de plus en plus courants.

La biométrie peut permettre également la reconnaissance de thèmes suivants :

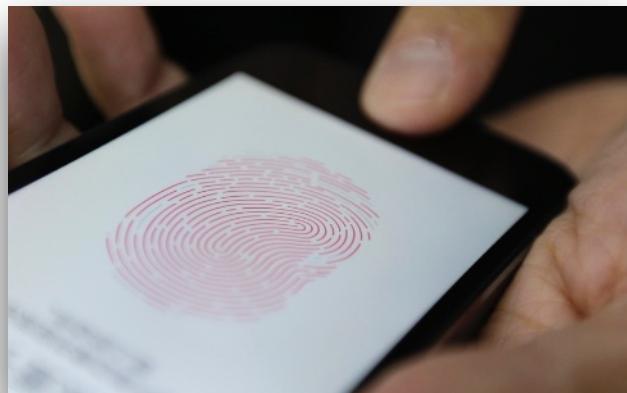
- biométrie légère (par exemple, couleur des yeux, taille, cicatrices, marques et tatouages)
- Estimation de l'âge
- Reconnaissance du sexe
- Détection des falsifications profondes
- La préservation de la vie privée pour protéger les données biométriques (c'est-à-dire la protection des modèles).

## 2. Les différentes biométries

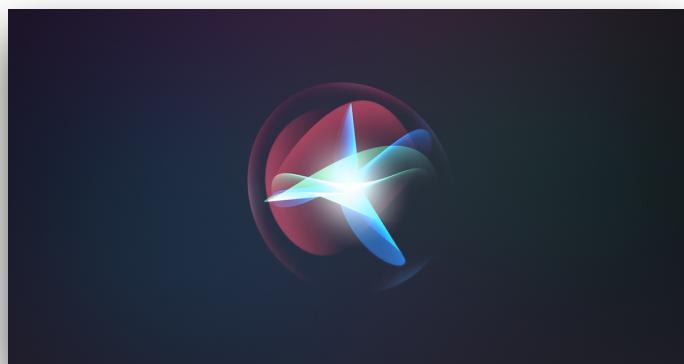
- Reconnaissance des visages



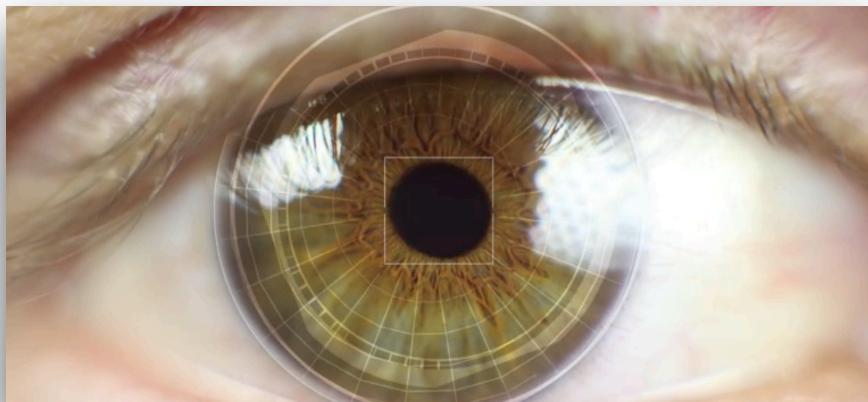
- Reconnaissance des empreintes digitales



- Reconnaissance du locuteur



- Reconnaissance de l'iris



- Reconnaissance des veines



## II - La Reconnaissance des veines palmaires

Le sujet de notre projet de 5e année porte sur la reconnaissance des veines palmaires.

### **1. Définition**

L'authentification des veines de la paume est l'une des modalités d'authentification biométrique et est classée comme une authentification biométrique physiologique. Elle utilise les veines de la paume, une image vasculaire de la paume d'une personne.

## 2. Histoire

Voici un petit résumé des acteurs principaux qui ont été les premiers à offrir cette nouvelle biométrie.

Un brevet pour l'authentification des veines de la main a été déposé en 1985 par **Joseph Rice** aux États-Unis . Le premier dispositif d'authentification des veines de la paume de main a été présenté par **Advanced Biometrics**, Inc. aux États-Unis en 1997.

Suite à cette déposition de brevet, **Hitachi**, **Fujitsu** et **Techsphere** ont lancé des produits de sécurité reposant sur la biométrie veineuse.

**Le laboratoire Fujitsu** a commencé à développer l'authentification de la veine de la paume de la main comme biométrie sans contact en 2000.

En 2003, un nouveau dispositif sans contact a été lancé par Fujitsu au Japon.

En 2004, les institutions financières japonaises, la **Banque de Tokyo-Mitsubishi**, ont été les premières à adopter la technologie de Fujitsu pour confirmer l'identité de leurs clients.

Il s'agissait de la première application majeure au Japon dans laquelle une entreprise privée a adopté l'authentification veineuse dans un service destiné au grand public. Le concept et la mise en œuvre d'un capteur sans contact de Fujitsu ont été récompensés par le prix de l'innovation technologique 2005 du Wall Street Journal pour la sécurité des réseaux.

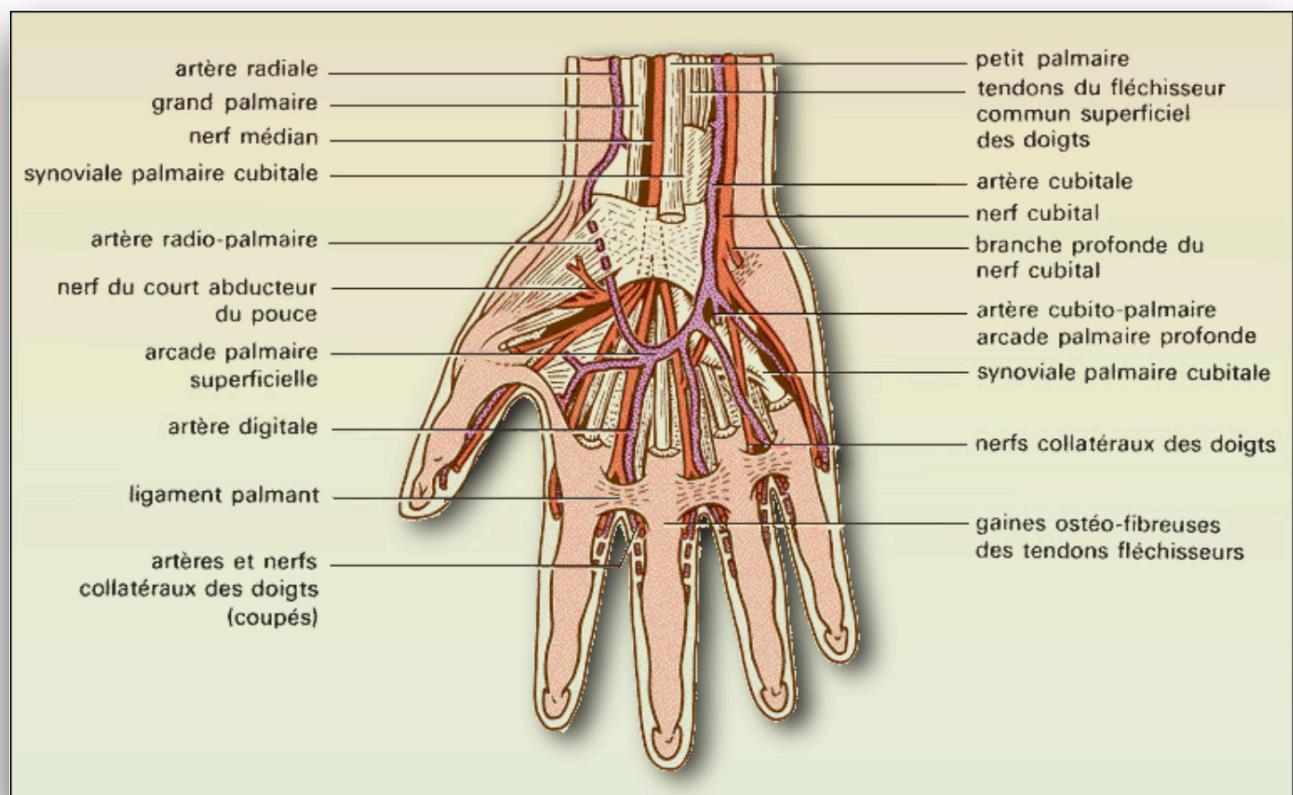
### 3. Description de la modalité des veines palmaires

La modalité d'authentification de cette biométrie est purement physiologique.

Les veines palmaires superficielle et profonde suivent le trajet des artères palmaires, traversant la paume et s'abouchant aux veines cubitale et radiale.

Comme les veines se trouvent à l'intérieur du corps humain, les images de celles-ci sont sûres et difficiles à voler ou à dupliquer.

Des détails concernant l'anatomie de la main sont indiqués sur l'image suivante :



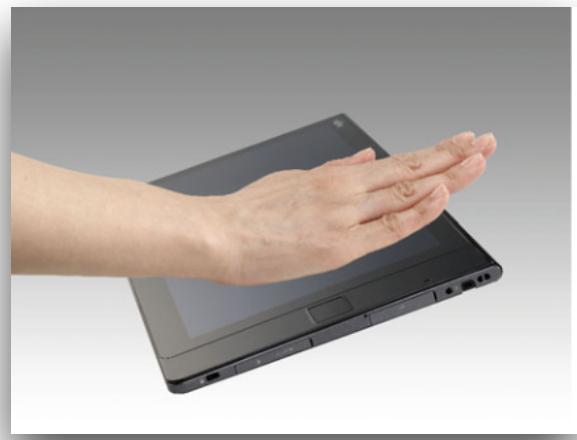
## 4. Cas d'utilisation

L'authentification par les veines de la main est utilisée dans le monde entier. Cette authentification peut permettre l'entrée d'un simple appareil jusqu'à des systèmes plus complexes comme des grands systèmes de contrôle d'accès.

### Authentification à la connexion

Les capteurs de veines de la main peuvent être intégrés dans une souris de PC. L'utilisation d'une souris comme capteur d'authentification des veines de la main offre des avantages en termes de commodité et de gain de place.

Les PC portables et des tablettes tactiles sont également équipés de l'authentification veineuse palmaire avec lesquels il est possible d'effectuer une authentification pré-boot au démarrage du BIOS.



Des centaines de milliers de salariés et d'employés les utilisent dans les grandes entreprises technologiques et les gouvernements.

## Authentification par le contrôle d'accès physique

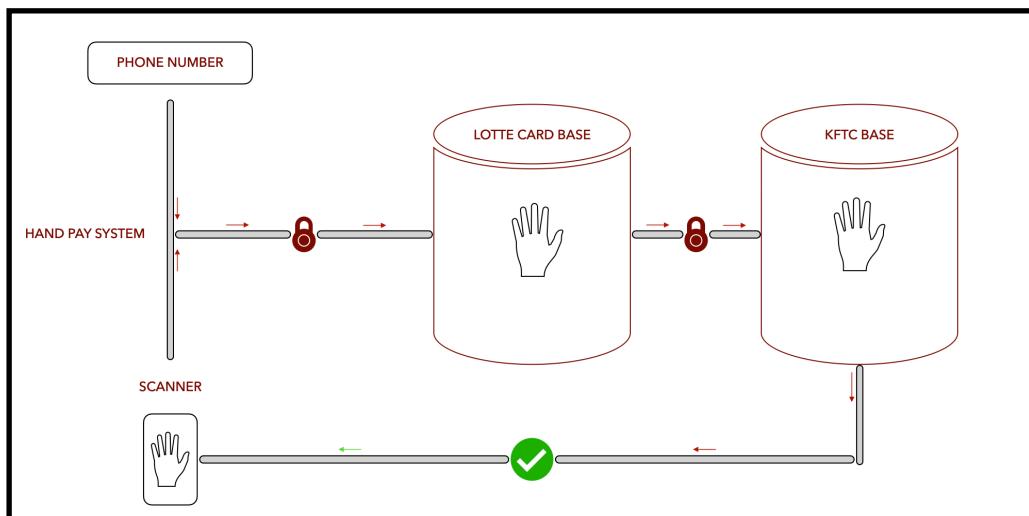
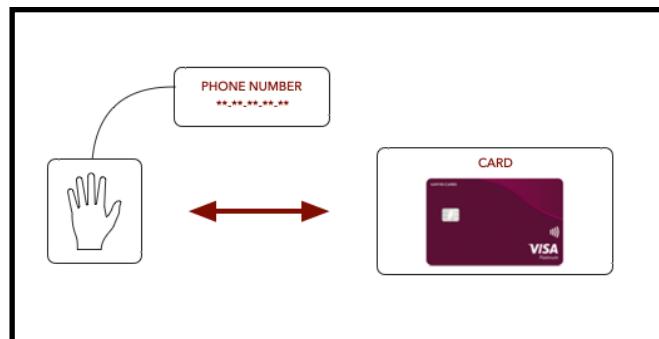
Plusieurs systèmes de contrôle d'accès physique existent dans plusieurs services dont :

### A. SYSTÈME DE PAIEMENT

Un système de paiement utilisant l'authentification des veines de la paume de la main, appelé "**Hand Pay Service**", a été introduit par la grande société coréenne de cartes de crédit **Lotte Card Co., Ltd.**

Tirant pleinement parti de la technologie d'authentification des veines de la paume de la main fièrement fournie par Fujitsu, Lotte Card a lancé le premier service de bio-paiement en Corée en mai 2016, ce qui permet aux clients de Lotte Card d'effectuer des paiements forfaits par carte de crédit même lorsqu'ils ne la portent pas sur eux.

L ' utilisation de la paume de la main des clients ainsi que leur numéro de téléphones vient échanger le processus de la carte de crédit.



Les utilisateurs du service de paiement HandPay lancé par LOTTE CARD doivent d'abord scanner et enregistrer leurs données biométriques veineuses au centre de cartes.

Les informations relatives au motif veineux scanné sont ensuite reliées aux données de la carte de crédit de la personne sur un serveur biométrique. Les informations relatives aux veines sont ensuite converties en données cryptées et sont divisées et stockées dans le centre de gestion des données distribuées sur les bio-informations du Korea Financial Telecommunications & Clearing Institute (KFTC) et dans l'environnement système de Lotte Card, afin de renforcer encore la sécurité.

Lorsque les utilisateurs enregistrés paient à la caisse, ils saisissent leur numéro de téléphone mobile dans le terminal de paiement et présentent ensuite leur paume au scanner. Les informations du motif veineux sont comparées aux informations de la carte de crédit sur le serveur d'authentification biométrique. Ces données sont ensuite comparées aux informations de la carte de crédit sur le serveur biométrique et le paiement est effectué.



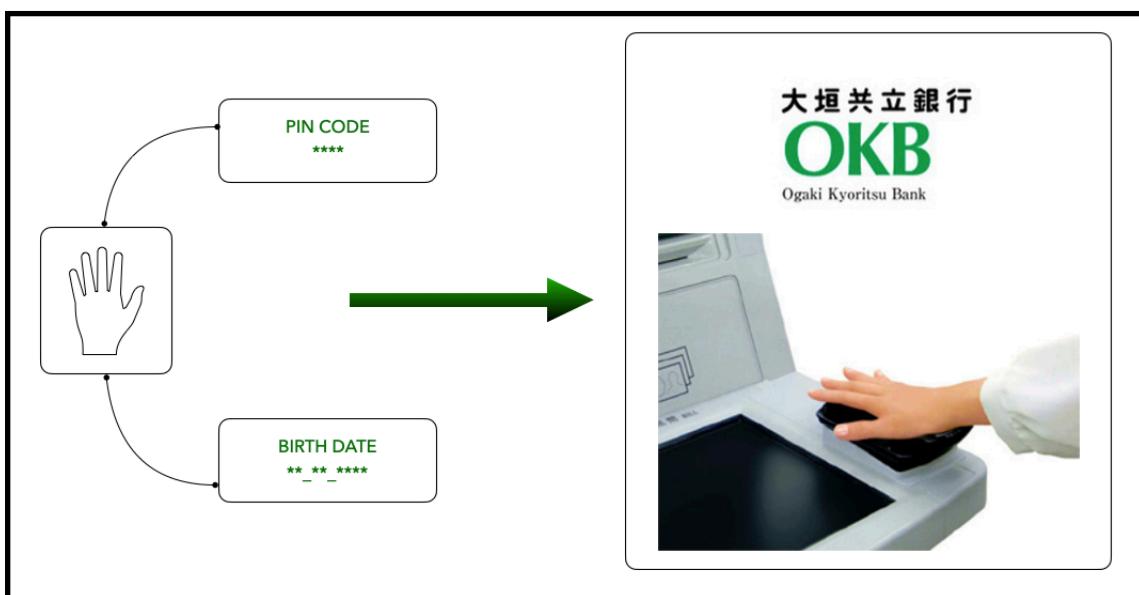
## B. SERVICE FINANCIER

En 2003, le Japon a connu un problème social important qui était basée sur l'augmentation rapide des préjudices financiers, causés par des prélèvements frauduleux sur des comptes bancaires grâce à l'utilisation de fausses cartes bancaires fabriquées à partir de cartes volées ou écrémées.

La "loi sur la protection des informations personnelles" est entrée en vigueur en mai 2005 et, en réponse, les institutions financières japonaises se sont concentrées sur les méthodes d'authentification biométrique et les cartes à puce, afin de renforcer la sécurité de l'identification personnelle.

L'authentification par les veines de la main est la forme d'authentification biométrique qui a été la plus rapidement introduite pour la confirmation des clients dans les établissements bancaires.

En 2012, **Ogaki Kyoritsu Bank Ltd.** au Japon a lancé un nouveau service de système de guichet automatique biométrique sans carte appliquant l'authentification de la veine palmaire. Grâce à ce système, les clients sont en mesure d'utiliser les services de GAB (guichet automatique de banque) pour les retraits, les dépôts et les consultations de solde sans livret ni carte de GAB.



En combinant leur date de naissance, l'authentification de la veine palmaire et leur code PIN, les clients ont accès à des services financiers qui allient sécurité et commodité.

## **C. SOIN DE SANTÉ**

L'hôpital de Sapporo de l'association Keiyu au Japon a également adopté l'authentification par les veines palmaires pour l'authentification des patients dans son système de dossiers médicaux électroniques. Les patients qui doivent subir une opération enregistrent le profil de leurs veines palmaires avant l'opération. Le jour de l'opération, le profil veineux palmaire enregistré et le profil veineux palmaire scanné du patient sont comparés, confirmant que le patient à opérer est la bonne personne. Cela permet d'éviter que le mauvais patient soit opéré, ce qui pourrait se produire si deux patients portent le même nom, par exemple.

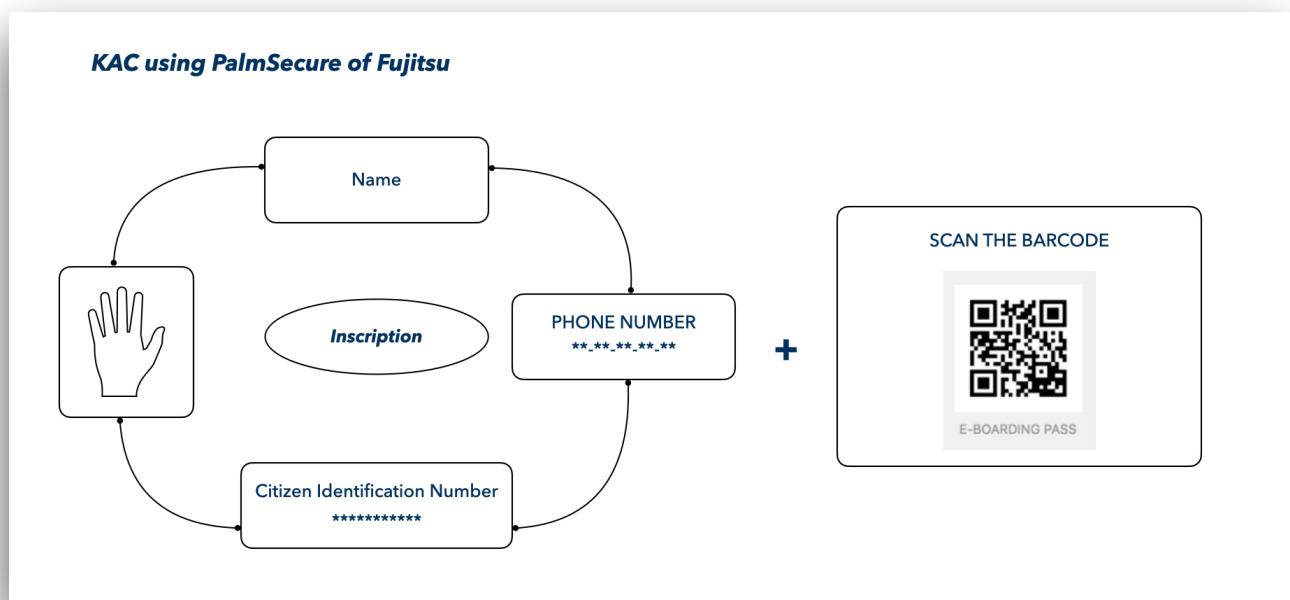
## **D. SÉCURITÉ DES AÉROPORTS**

En Corée du Sud, la Korea Airports Corporation (KAC) a déployé un système d'authentification des veines de la paume de la main dans les 14 aéroports nationaux relevant de sa compétence, afin de réduire les encombrements en identifiant les passagers à l'embarquement par authentification biométrique.



Les passagers qui n'avaient pas apporté leur carte d'identité de citoyen n'ont pas pu embarquer sur leur vol, ce qui a compromis la qualité du service à la clientèle. KAC a prêté attention à la haute précision d'identification et à la commodité de l'authentification des veines de la paume, et a donc décidé de déployer un système d'identification personnelle utilisant l'authentification des veines de la paume.

Les utilisateurs peuvent s'inscrire à l'avance auprès des dispositifs d'enregistrement installés dans les aéroports.



En associant le dessin de leur veine palmaire à leur numéro d'identification de citoyen, leur nom et leur numéro de téléphone.

Ensuite, après avoir scanné un code-barres sur leur billet, les utilisateurs peuvent confirmer leur identité en tendant la main aux nouvelles portes de confirmation d'identité installées avant les points de contrôle de sécurité. Les utilisateurs n'auront pas à porter constamment sur eux leur carte d'identité, et le système réduira les temps d'attente et permettra un traitement plus fluide dans les aéroports.

## **E. GOUVERNEMENTS ET MUNICIPALITÉS**

La ville de Naka, dans la préfecture d'Ibaraki, au Japon, a mis en place un système utilisant la technologie d'authentification des veines de la main pour la nouvelle bibliothèque publique de la ville en octobre 2006. Ce système de bibliothèque est le premier de ce type au monde. Les utilisateurs peuvent emprunter des livres à la bibliothèque en utilisant l'authentification par les veines de la main. Les usagers de la bibliothèque publique de Naka City auront le choix entre l'utilisation d'une carte d'identité avec une puce électronique intégrée ou l'utilisation du système d'authentification par veine palmaire pour la vérification de l'identité.

Les utilisateurs qui choisissent l'authentification par veine de paume pourront emprunter des documents de la bibliothèque ou utiliser sa section audiovisuelle sans utiliser de carte d'identité. Les utilisateurs commencent par saisir leur date de naissance, puis il leur suffit de suspendre leur main au-dessus du dispositif d'authentification pour que le motif de la veine de la paume soit comparé au motif préenregistré pour vérification.

## 5. Acquisition des veines palmaires

Les dessins des veines se trouvent dans le tissu sous-cutané de la paume de la main d'une personne et sont capturés à l'aide de rayons proches de l'infrarouge. Cette technologie est appelée spectroscopie et imagerie dans le proche infrarouge (NIR).

Les images des veines de la paume peuvent être capturées à l'aide de deux méthodes différentes : **la méthode par réflexion** et **la méthode par transmission**.

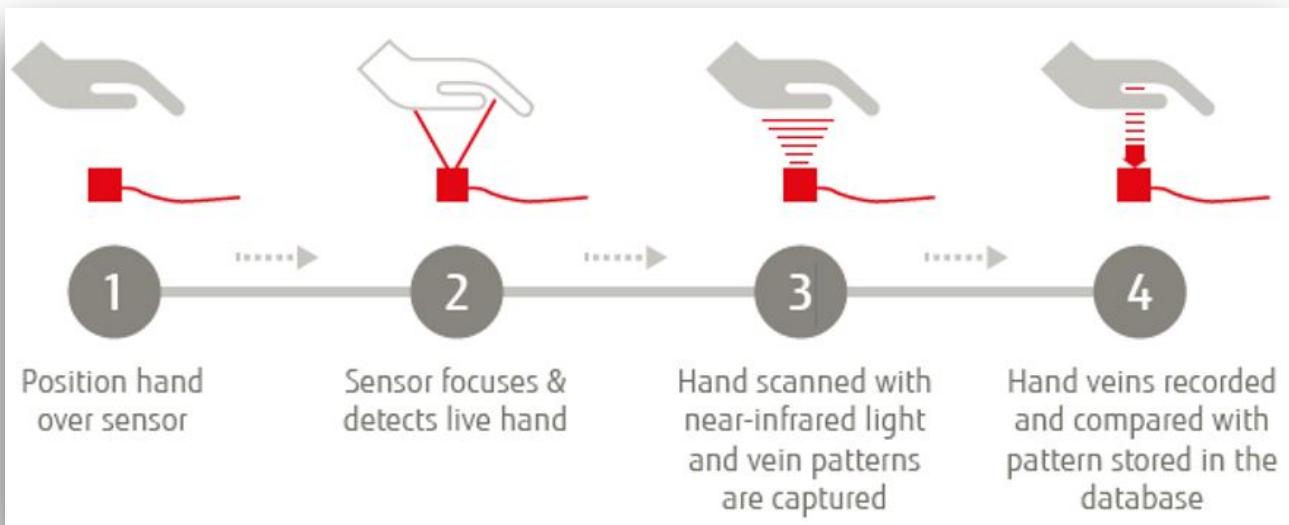
Dans la méthode de transmission, la paume de la main est éclairée depuis le dos de la main et l'image est capturée depuis le devant. Et le dispositif d'illumination et le dispositif de capture sont séparés, se faisant face à travers la paume.

Dans la méthode par réflexion, la paume de la main est éclairée de l'avant et l'image est capturée du même côté. De plus, pour cette méthode, le dispositif d'éclairage et le dispositif de capture peuvent être intégrés ensemble pour créer un dispositif plus compact, car la direction de l'éclairage est la même que la direction de la capture d'image.

Pour obtenir une image de haute qualité des veines de la paume, le processus d'imagerie doit être contrôlé de manière adéquate en raison du mouvement ou de la position de la main. En outre, l'éclairage doit être contrôlé en fonction des conditions d'éclairage ambiantes autour du capteur.

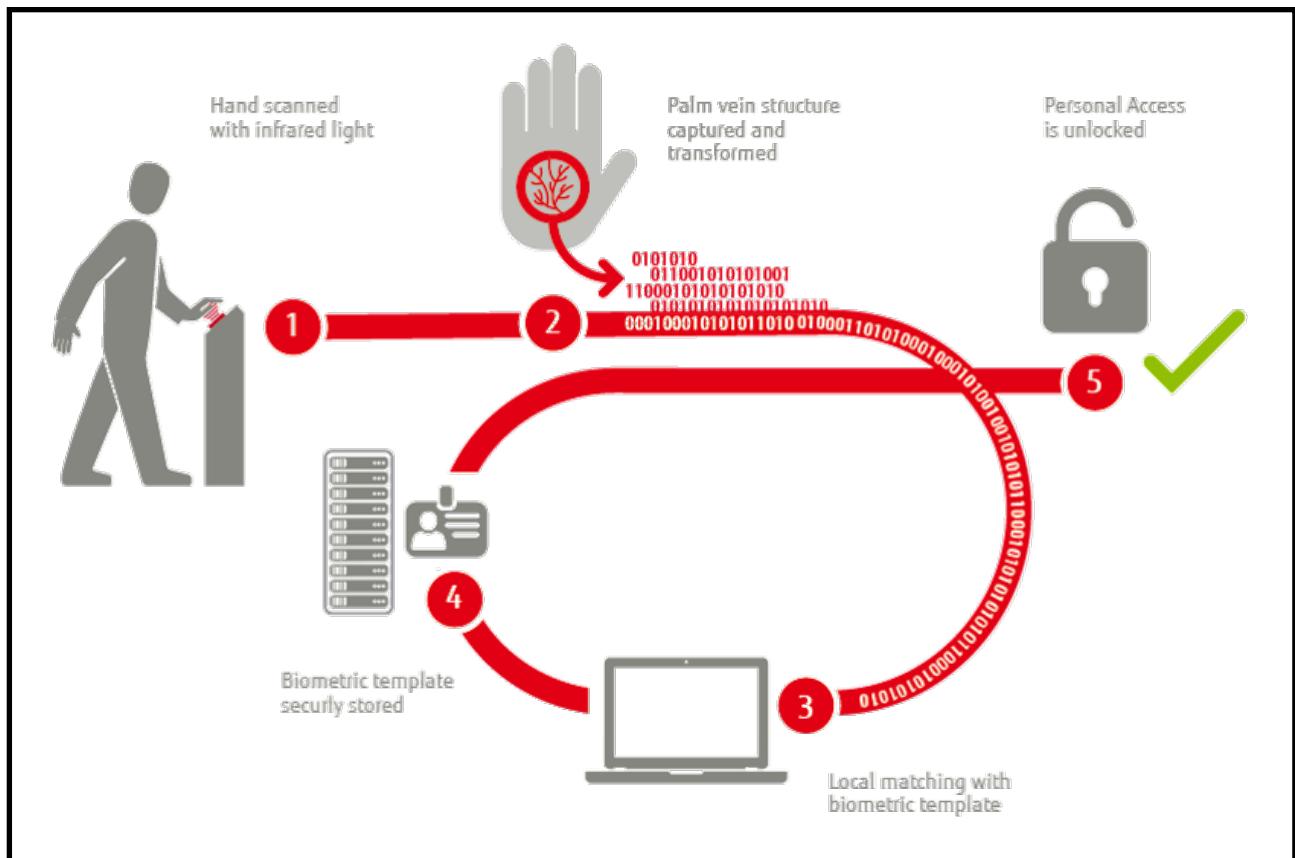


## Schéma explicatif de la détection de veines palmaires avec un capteur utilisant la méthode de réflexion



## 6. Architecture d'un système de reconnaissance des veines palmaires

Architecture d'un système de reconnaissance de veines palmaires



## Conclusion

Les caractéristiques des veines de la paume sont normalement capturées en utilisant la lumière proche infrarouge, soit par la méthode de réflexion, soit par la méthode de transmission.

Comme il s'agit d'un type d'identification biométrique sans contact, elle convient aux applications qui exigent un niveau d'hygiène élevé ou aux applications publiques.

L'authentification des veines de la paume de main a été utilisée dans diverses applications telles que les systèmes de sécurité des portes, les systèmes de gestion des connexions pour les PC, les services financiers, les services de paiement et les systèmes d'identification des patients dans les hôpitaux. Le motif veineux de la paume a une complexité bi-dimensionnelle, et comme l'image existe sous la peau, l'image acquise est très stable. Sur la base de ces avantages, nous pensons que l'authentification par veine de la paume va se généraliser.

## CHAPITRE 2 : MACHINE & DEEP LEARNING

## Introduction

Dans le cadre de notre projet, il est nécessaire de savoir comment il est possible de traiter des images de veines palmaires pour pouvoir effectuer de la reconnaissance biométrique. Des architectures existent pour répondre à ce besoin, que nous allons voir dans ce chapitre sur les modèles d'intelligence artificielle.

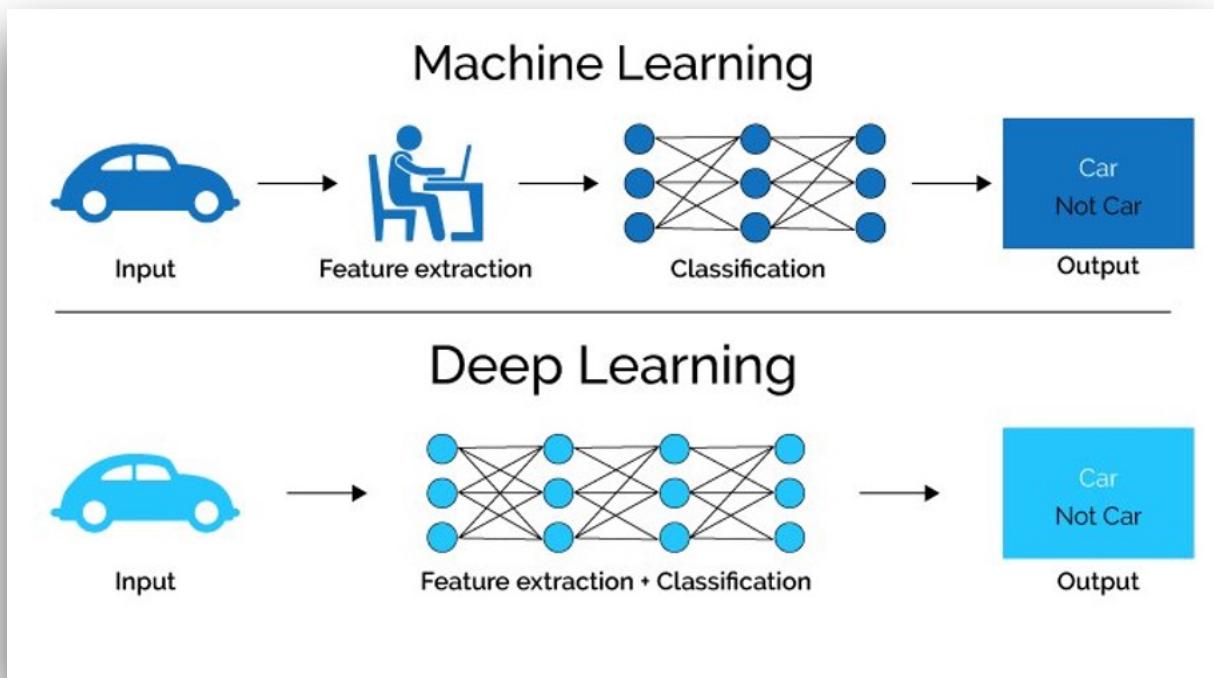
### I - Machine vs Deep Learning

L'apprentissage automatique (ou Machine Learning) est un domaine de l'intelligence artificielle visant à effectuer des opérations statistiques et mathématiques sur des données afin d'en tirer des informations importantes. A partir d'observations, un ordinateur est capable d'interpréter des résultats pour ainsi résoudre des tâches diverses, comme les problèmes biométriques (reconnaissances d'empreintes digitales, de visages...) ou encore le traitement du langage naturel (NLP).

Le Machine learning se base sur des algorithmes qui fonctionnent en 2 parties majeurs :

- L'apprentissage : la machine va analyser des données issues d'observations. Ces données seront utilisées dans un cycle d'entraînement dans le but d'établir un modèle, qui sera évalué et amélioré jusqu'à obtenir des résultats satisfaisants.
- La mise en production: une fois le modèle établi, celui-ci va être utilisé sur des nouvelles données afin de réaliser un tâche.

Dans le domaine de Machine, il faut distinguer le domaine de l'apprentissage supervisé (Supervised Learning) et celui de l'apprentissage profond (Deep Learning). Bien que répondant au même besoin d'automatisation de tâches complexes, ils diffèrent au niveau de la phase d'apprentissage.



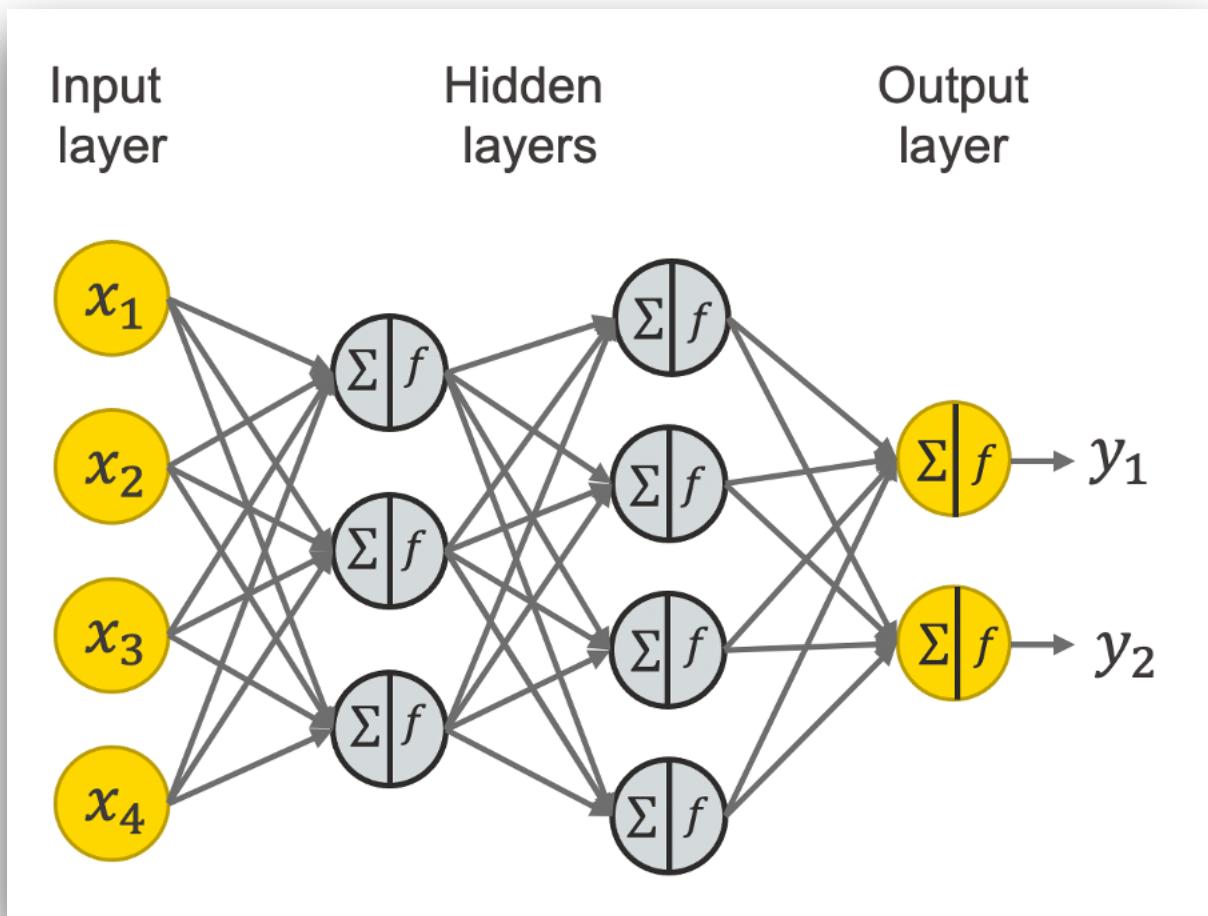
## II - Principe de Deep Learning

L'apprentissage supervisé se base sur des observations labellisées par un humain. Au contraire, le Deep Learning s'occupe lui-même de labelliser les données qui lui sont fournies. Bien que cela nécessite beaucoup plus de puissances de calculs, le processus de labellisation devient plus facilement automatisable et rapide.

### III - Réseaux de neurones

#### 1. Artificial Neural Network (ANN)

Les algorithmes de Deep Learning se basent sur des réseaux de perceptrons (neurones), aussi communément appelés Artificial Neural Network (ANN). Les modèles ANN sont basés sur le concept du **Feed-Forward Neural network (réseau de neurones à propagation avant)**. Les entrées sont uniquement traitées et transmises dans un seul sens, jusqu'à atteindre la couche de sortie.



Ces réseaux sont constitués :

- D'une couche d'entrée (Input Layer), qui correspond au paramètres de notre modèle ;
- D'une ou plusieurs couches cachées (Hidden Layers) ;
- D'une couche de sortie (Output Layer), les résultats de l'apprentissage.

Les sorties de chaque perceptrons du modèle constituent les entrées de ceux de la couche suivante. Au sein de chaque perceptron, les résultats des neurones précédents pointant sur celui-ci seront sommés, et grâce à une fonction d'activation, pouvant être transmis au neurone suivant ou non.

L'avantage de ce modèle est sa capacité d'apprendre n'importe quelle fonction non-linéaire. Grâce au paramétrage des fonctions d'activations de chaque neurone du réseau, il est possible d'introduire ces relations complexes entre les entrées et les sorties.

Dans le cadre du traitement d'images, les architectures ANN rencontrent 2 grandes limitations.

De part la nature de l'entrée, il est nécessaire de convertir une image en 2 dimensions en un vecteur unidimensionnel, ce qui augmente lourdement la quantité de paramètres à injecter dans le modèle.

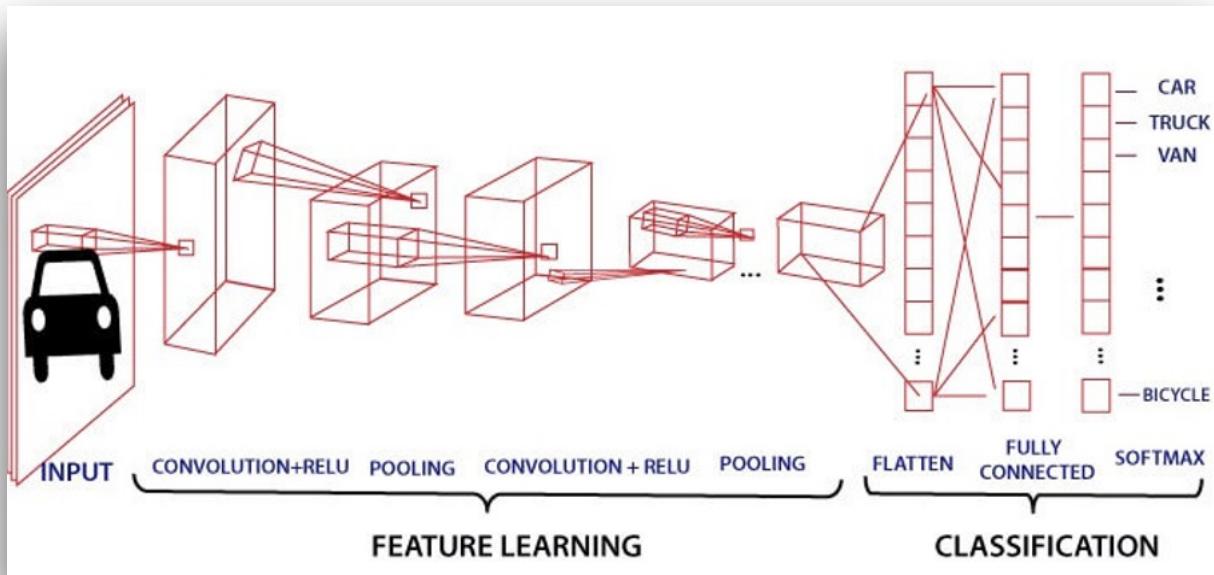
De plus, la conversion supprime l'arrangement des pixels dans une image, qui constitue une information importante dans un problème de reconnaissance.

Des solutions ont été développés pour surmonter ces problèmes, par le biais de ces grandes architectures du Deep Learning :

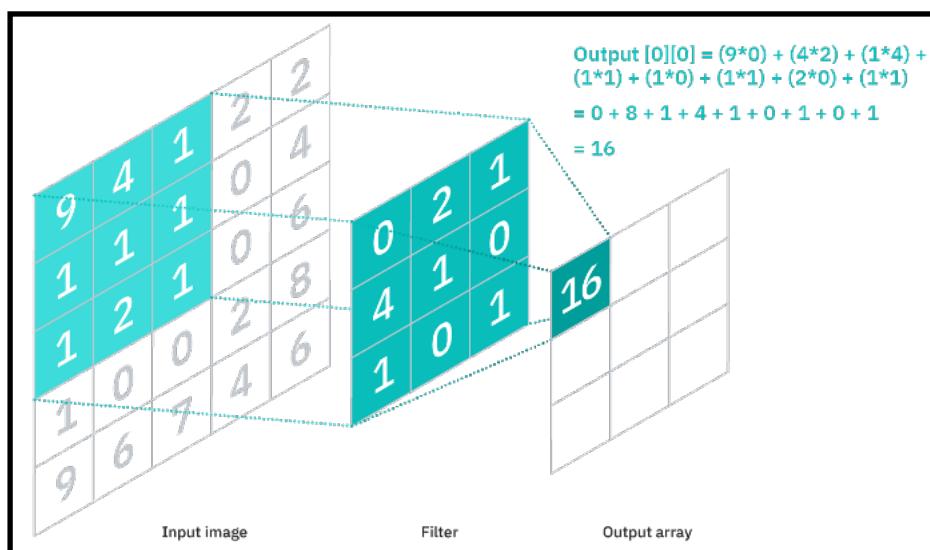
- **Convolution Neural Network (CNN)**
- **Recurrent Neural Network (RNN)**

## 2. Convolution Neural Network (CNN)

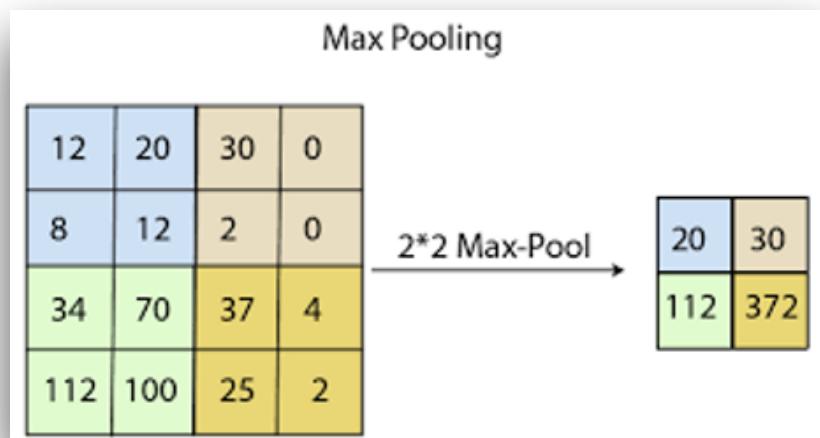
Les CNNs, ou Convolution Neural Networks, sont des modèles basés sur des opérations de convolution pour extraire des points d'intérêts d'une entrée. Ils sont utilisés principalement dans le domaine de traitement d'images.



Ces architectures se basent sur ensembles de filtres, appelés noyaux, qui seront appliqués successivement sur les différents états de l'entrée afin d'en extraire les points d'intérêts.



Après la convolution vient l'étape de Pooling. Celle-ci permet de réduire la dimension de l'échantillon d'entrée tout en conservant les éléments importants.



Ce cycle de convolution et de Pooling se poursuivra jusqu'à obtenir un vecteur unidimensionnel, qui pourra ensuite être injecté dans un ANN pour apprentissage.

### 3. Recurrent Neural Network (RNN)

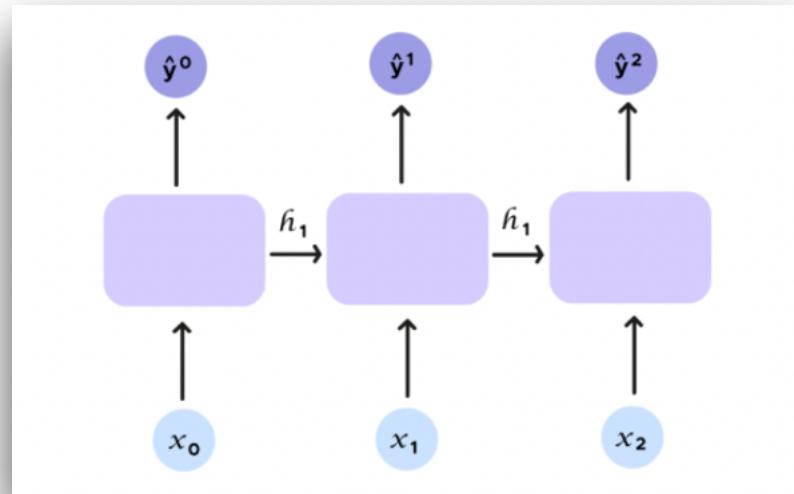
RNN (Recurrent Neural Networks) est une architecture de réseau de neurone qui utilise le principe de récurrence. Il est très utilisé dans le NLP (Natural Language Processing).

Il est également utilisé dans le domaine de la Computing Vision, mais seulement dans des problématiques lorsque le temps est un facteur nécessaire pour le modèle du réseau de neurone. Par exemple, déterminer la trajectoire d'une balle, ou bien déterminer et prédire les marches d'une personne.

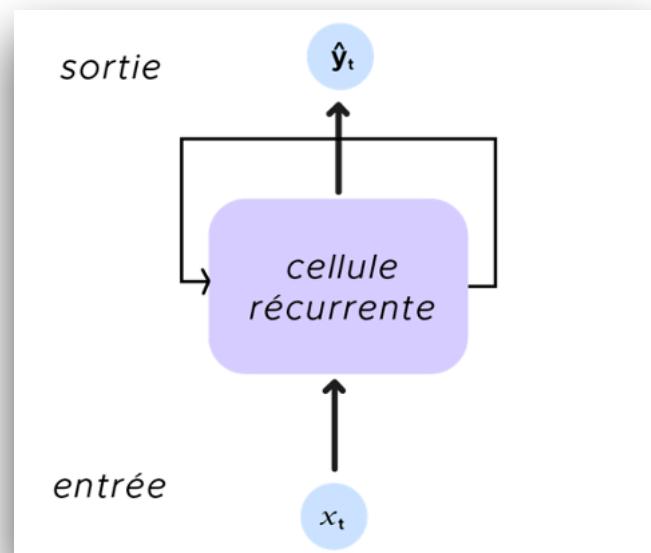
Voici des exemples de sujet utilisés par le RNN :

- Speech Recognition
- Music Generation
- Sentiment Classification
- DNA Sequence analysis
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition

Voici l'architecture d'un réseau de neurone RNN classique :

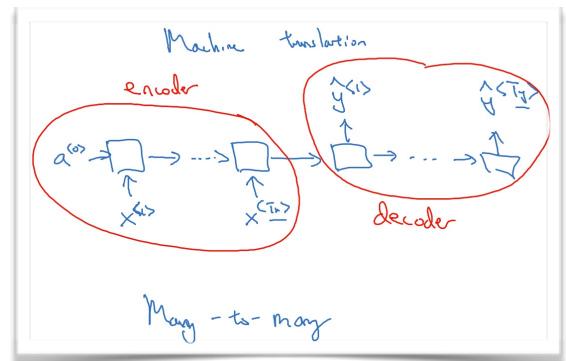
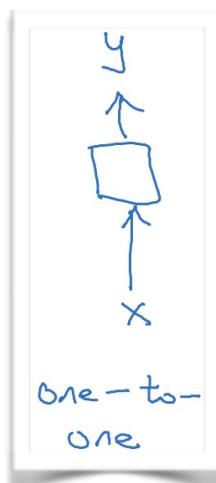
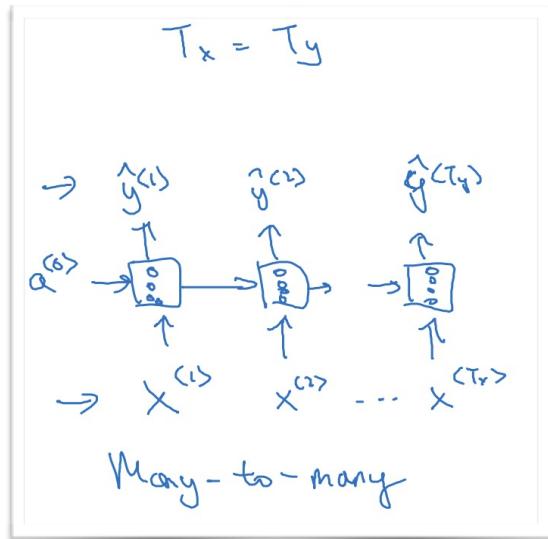
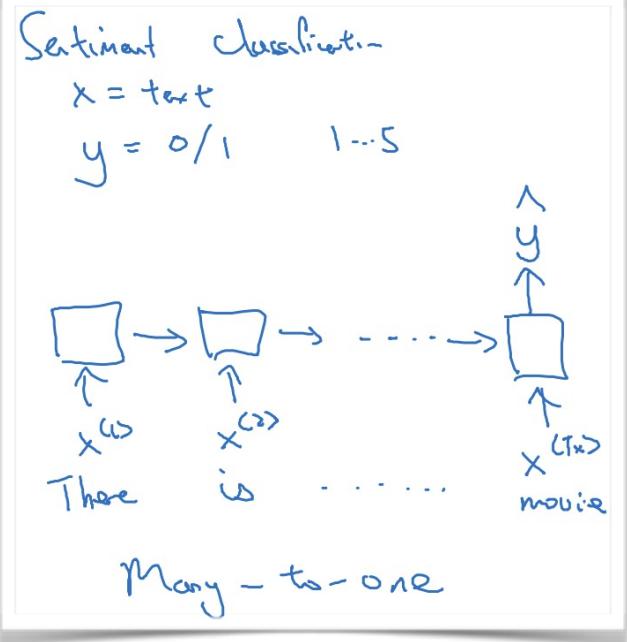
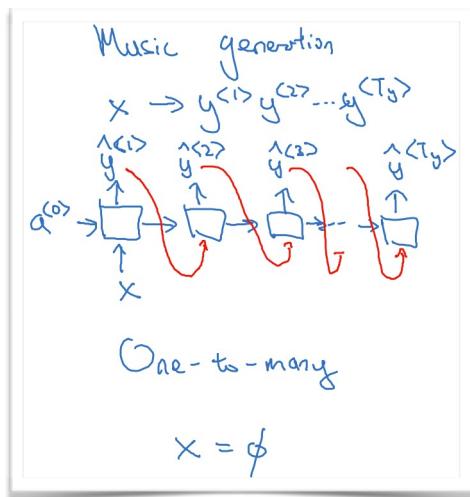


Voici l'une des cellules présentent dans l'architecture RNN

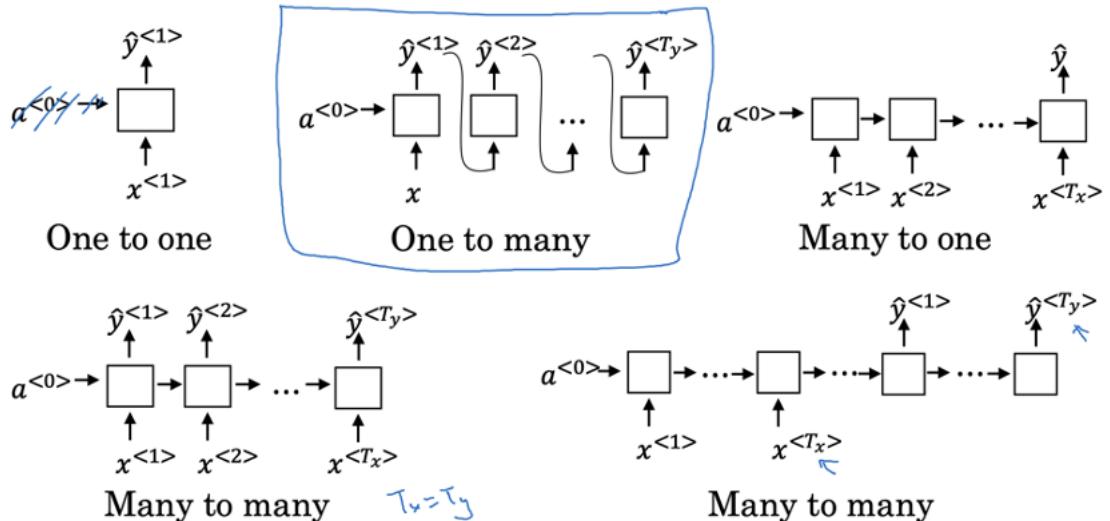


Voici des exemples d'architectures de RNN («input» -to- «output») :

- Many-to-Many
- Many-to-One
- One-to-One
- One-to-Many



## Summary of RNN types



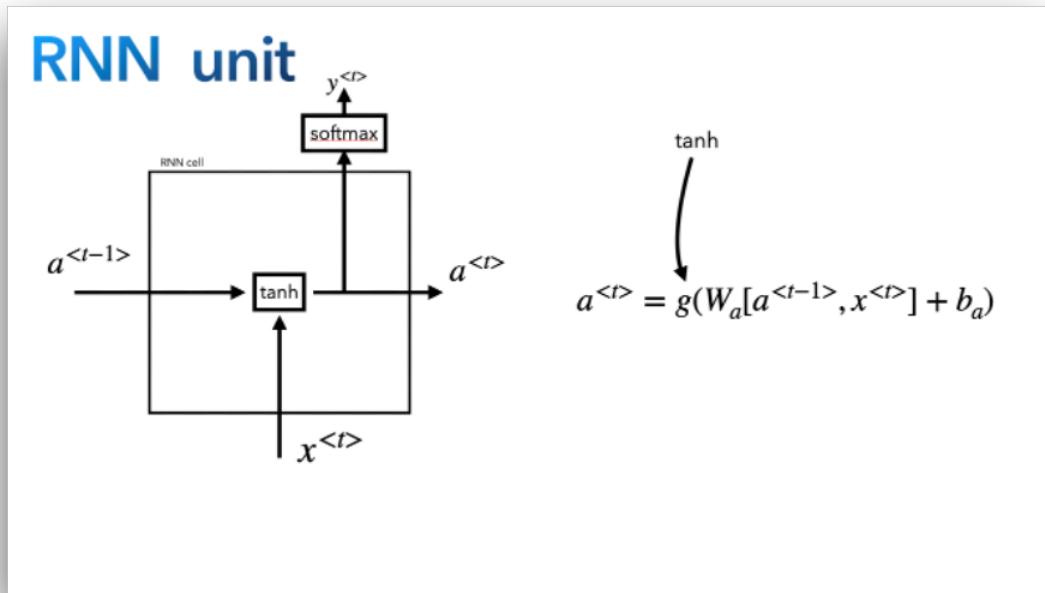
Andrew Ng

Le problème avec l'architecture du RNN c'est sa mémoire très courte. Elle n'est pas adaptée pour une séquence très longue. Pour combler ce problème, il existe plusieurs solutions :

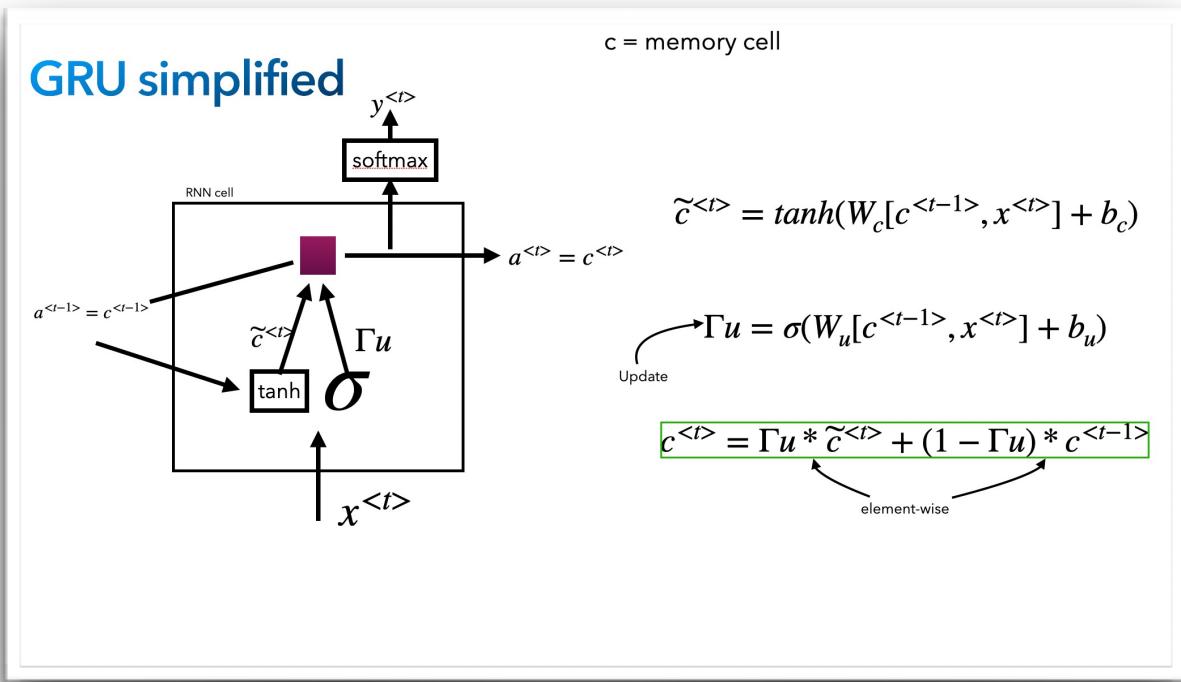
- GRU (Gated Recurrent Unit)
- LSTM (Long Short Term Memory)

Ces solutions permettent de contrôler les informations que nous gardons et transmettons à travers le temps. Ils changent l'architecture de la cellule.

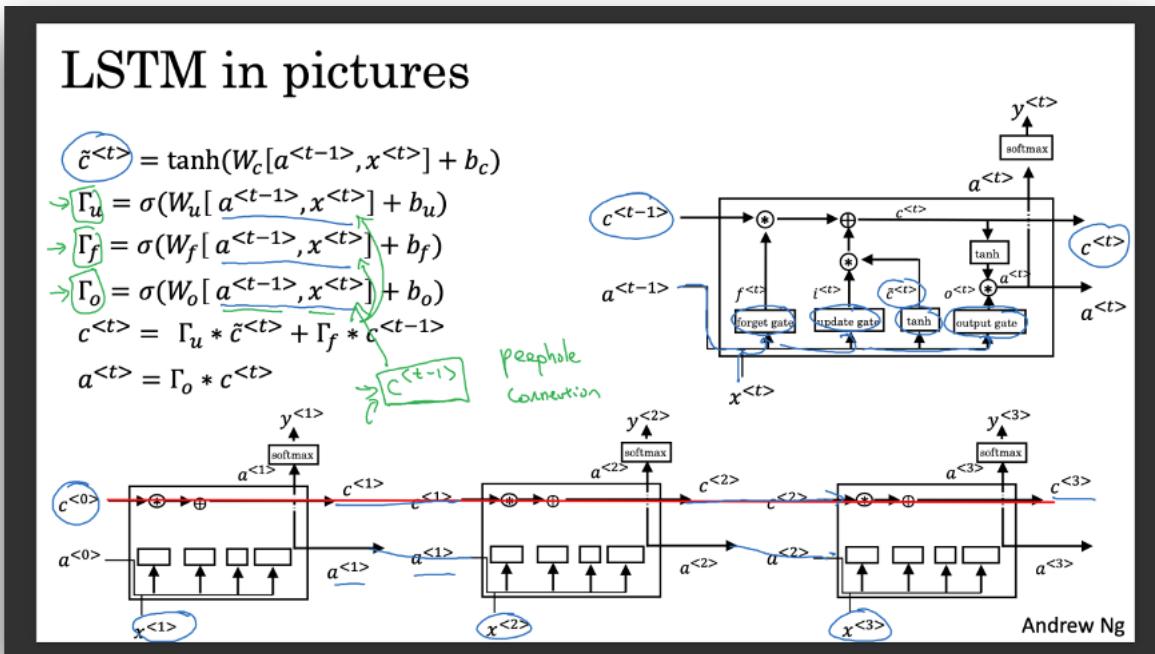
Voici le schéma d'une cellule RNN standard avec une fonction d'activation tanh.



Voici le schéma d'une cellule GRU (simplifié) :



Voici le schéma d'une cellule LSTM standard :



## Conclusion

Pour la résolution de notre problème, il est clair que l'utilisation d'un modèle de CNN est indispensable. L'extraction des informations issues des veines palmaires nécessitent de conserver un maximum d'informations que les autres architectures ne sauraient présenter, notamment l'arrangement spatial, pour pouvoir effectuer une reconnaissance biométrique précise.

## CHAPITRE 3 : Modélisation et Réalisation

## Introduction:

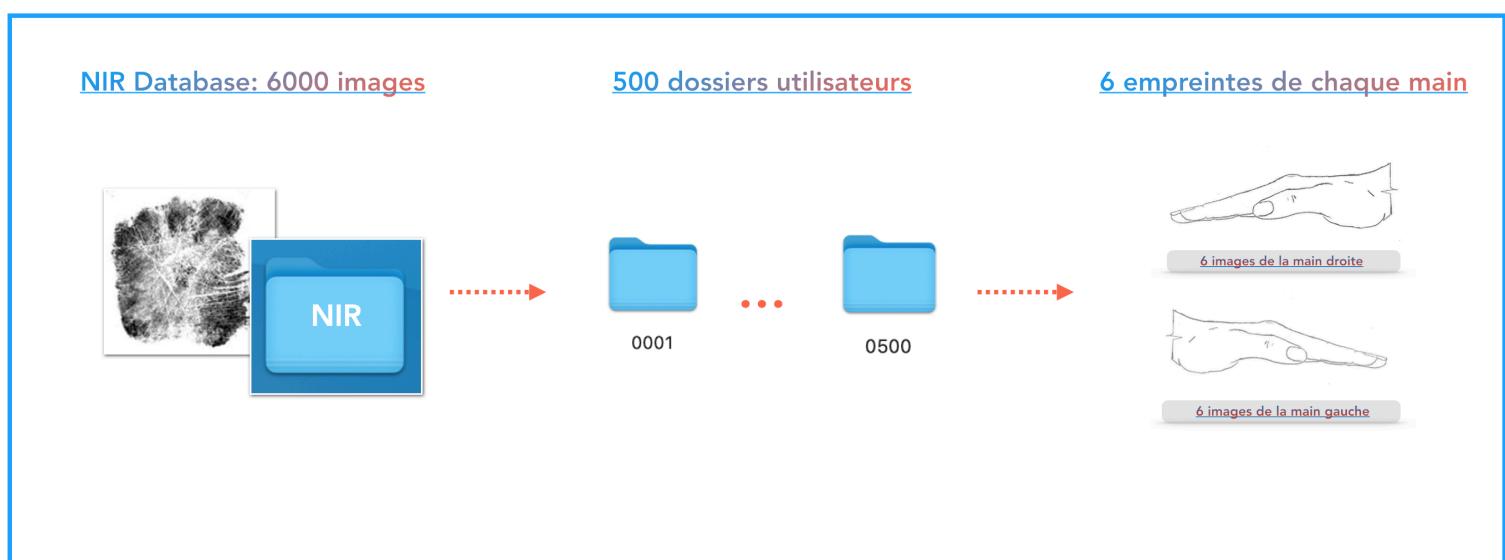
Dans ce chapitre, nous présenterons une application sur laquelle nous avons travaillé à partir d'un dossier comportant des images de veines palmaires.

### I - Une première application

Notre projet de 5e année a été effectué sous l'encadrement de notre professeur M.BOUBCHIR.

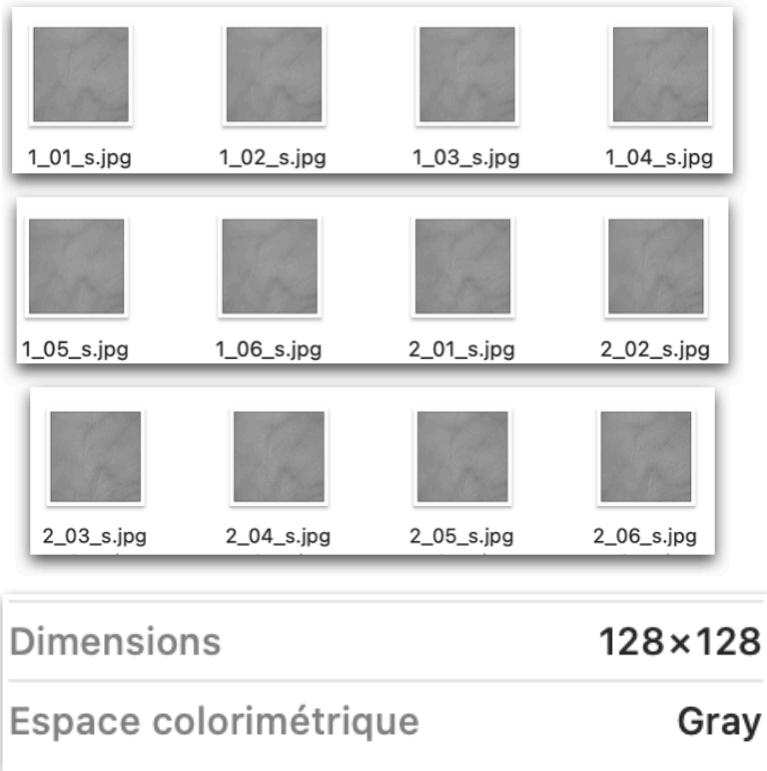
Ce projet se situe dans le domaine de la reconnaissance biométrique des veines palmaires.

500 personnes ont enregistré et scanné leurs empreintes palmaires qui ont été placé avec sécurité dans une Database. Notre encadrant M.BOUBCHIR nous a confié cette database, dans laquelle il y a 6000 images.



La Database NIR comporte 500 sous-dossiers utilisateurs. Ils ont chacun leur tour déposé leurs empreintes de paume de main.

Chaque dossier utilisateur disposent **images de dimension 128x128** et de **couleur noir et blanc** pour chaque main. La main droite et gauche sont soulignées par 1\_ et 2\_ respectivement.

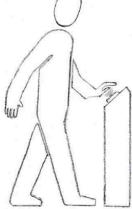


Notre première application consiste donc à entraîner des modèles CNN afin de prédire le numéro de l'utilisateur à partir d'**une image de veines palmaire**.

[500 utilisateurs ont enregistré et scanné leurs empreintes palmaires](#)

[Réception de la NIR Database: 6000 images](#)

[Entrainement de modèles CNN afin de prédire l'identité de l'utilisateur](#)



## 1 - Les étapes clés

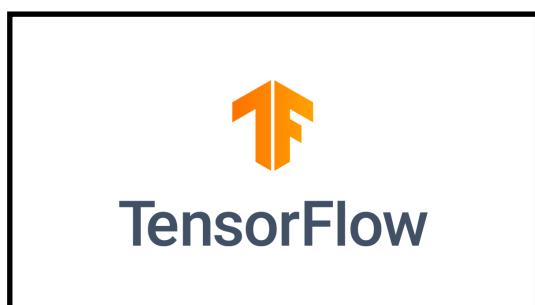
### A. Logiciel

Nous avons travaillé sur Jupyter Notebook qui est une application web originale pour la création et le partage de documents informatiques. Elle nous a offert une expérience simple, rationalisée et centrée sur notre projet de reconnaissance d'images.



### B. Librairies

En premier lieu, nous avons importé toutes les librairies qui nous seront nécessaires pour la suite notamment **keras** et **tensorflow** qui sont toutes les deux des libraires très utiles pour implémenter **un réseau de neurones convolutif (CNN)**.



```

Imports

import gc
import glob
import os
import sys
import cv2
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt
from random import randrange
from tqdm import tqdm, trange
from tqdm.notebook import tqdm_notebook

import warnings

from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split

from keras.optimizer_v2.gradient_descent import SGD
from keras.wrappers.scikit_learn import KerasClassifier
from keras.callbacks import ReduceLROnPlateau
from keras.layers import Dense, MaxPooling2D, Flatten, Conv2D, Lambda, Dropout, Leaky
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator

from mlxtend.evaluate import accuracy

from tensorflow.keras.optimizers import Adam, RMSprop, SGD, Nadam
#from tensorflow.keras.utils import to_categorical
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

```

## C. Dataset

Ensuite on charge les images en trois sous-ensembles :

### - Training set (70%)

Load images - Train(70%) Test(15%) Val(15%)

```

path_data = "data/data_palm_vein/NIR"
weight_path='saved_model/resnet50v2TL_20epochs_32batch.h5'

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    path_data,
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=1007,
    validation_split=0.3,
    subset="training",
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
)

```

.. Found 6000 files belonging to 500 classes.  
Using 4200 files for training.  
Metal device set to: Apple M1

- Testing set (15%)

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(  
    path_data,  
    labels="inferred",  
    label_mode="categorical",  
    class_names=None,  
    color_mode="rgb",  
    batch_size=32,  
    image_size=(128, 128),  
    shuffle=True,  
    seed=1007,  
    validation_split=0.15,  
    subset="validation",  
    interpolation="bilinear",  
    follow_links=False,  
    crop_to_aspect_ratio=False,  
)
```

```
Found 6000 files belonging to 500 classes.  
Using 900 files for validation.
```

- Validation set (15%)

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(  
    path_data,  
    labels="inferred",  
    label_mode="categorical",  
    class_names=None,  
    color_mode="rgb",  
    batch_size=32,  
    image_size=(128, 128),  
    shuffle=True,  
    seed=1007,  
    validation_split=0.15,  
    subset="validation",  
    interpolation="bilinear",  
    follow_links=False,  
    crop_to_aspect_ratio=False,  
)
```

```
Found 6000 files belonging to 500 classes.  
Using 900 files for validation.
```

## D. Recherche de modèles CNN

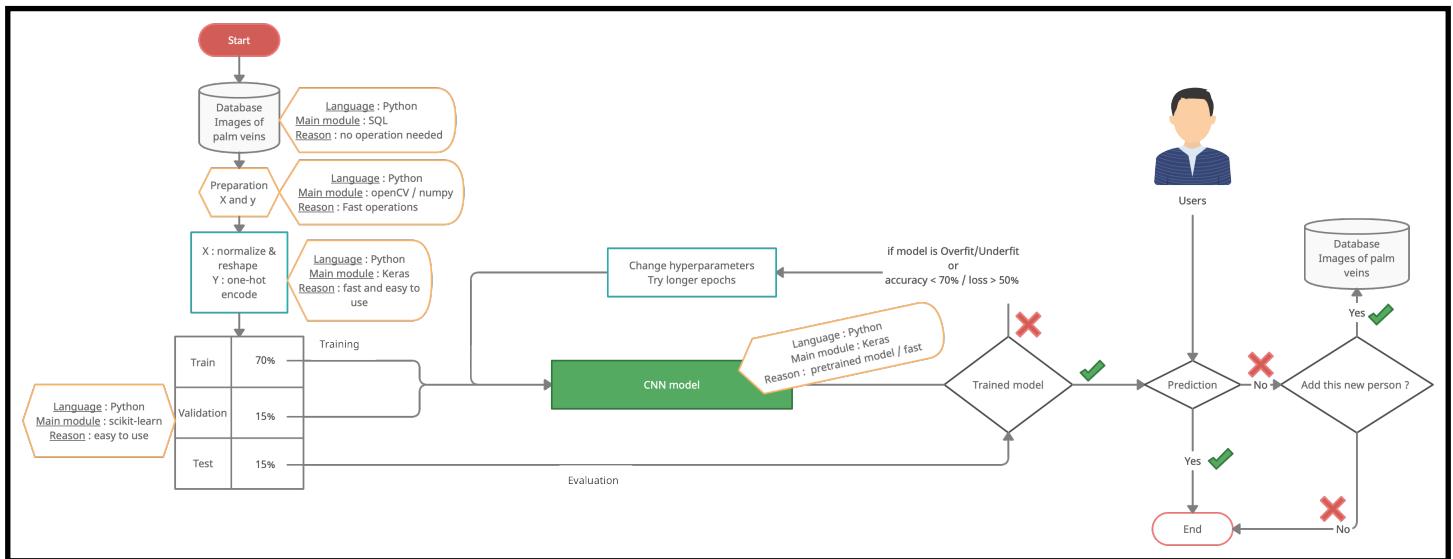
Après plusieurs recherches, nous avons trouvé des modèles CNN fourni par les services de Keras.

Nous pouvons voir dans ces différents modèles que le nombre de couches varie entre 7 et 150.

Models	Number of layers
<b>AlexNet</b>	8
<b>VGG-16</b>	16
<b>VGG-19</b>	19
<b>ResNet-50</b>	50
<b>ResNet-150</b>	150
<b>Inception/GoogLeNet</b>	22
<b>MobileNet</b>	28
<b>EfficientNet</b>	17
<b>ZFnet</b>	7

## 2 - L'architecture de notre première application

Avant de commencer à entraîner les modèles CNN, nous avons conçu l'architecture afin de bien assimiler les étapes de notre application.



## 3 - L'entraînement des modèles

### A. Résultat - Semestre 1

Nous avons entraîné les modèles présentés par Keras.

Et nous avons été confronté à plusieurs problèmes techniques.

- La valeur du « Loss » est nettement supérieur à la valeur d' « Accuracy »
- Le résultat reste négatif même pour des modèles possédant plus de 50 couches.
- Le temps d'entraînement est long ( 1000 epochs pour environ 50h d'entraînement).

Models	Loss / Accuracy
<b>VGG-16</b>	639%   0.10%
<b>VGG-19</b>	630%   0.00%
<b>ResNet-50</b>	604%   0.15%
<b>ResNet-150</b>	600%   1.16%
<b>Inception/GoogLeNet</b>	610%   0.50%
<b>Xception</b>	629%   0.11%
<b>ZFnet</b>	635%   0.00%

### Solution n°1 : Utilisation du Transfer Learning

Le Transfer Learning a connu un grand succès avec l'essor du Deep Learning.

Pour expliquer simplement cette méthode, prenons l'exemple de quelqu'un qui maîtrise la guitare et souhaite apprendre à jouer au piano. Il pourra capitaliser sur ses connaissances en musique pour apprendre à jouer un nouvel instrument. De la même manière, un modèle de reconnaissance de voiture pourra être très rapidement réadapté à la reconnaissance de camions.

L'apprentissage par transfert est une technique d'apprentissage automatique par laquelle un modèle est formé et développé pour une tâche et est ensuite réutilisé pour une deuxième tâche connexe. Il s'agit d'une situation dans laquelle ce qui a été appris dans un contexte est exploité pour améliorer l'optimisation dans un autre contexte.

Models	Sans Transfer Learning (Loss / Accuracy)	Avec Transfer Learning (Loss / Accuracy)
VGG-16	639%   0.10%	629%   0.00%
VGG-19	630%   0.00%	629%   0.00%
ResNet-50	604%   0.15%	0.95%   99.78%
ResNet-150	600%   1.16%	3.16%   99.33%
Inception/ GoogLeNet	610%   0.50%	1.57%   99.89%
Xception	629%   0.11%	107.76%   67.44%
ZFnet	635%   0.00%	X

Le Transfer Learning fait son effet sur certains de nos modèles.

3 modèles de CNN retiennent particulièrement notre attention après l'utilisation du Transfer Learning :

- **ResNet-150** : Loss = 3.16% | Accuracy = 99.33%
- **Inception / GoogleNet** : Loss = 1.57% | Accuracy = 99.89%

Et le meilleur résultat est obtenu par le modèle **ResNet-50** avec :  
Loss = 0.95% | Accuracy = 99.78%

## Entraînement du modèle ResNet-50

### Creation of the model - ResNet50 with Transfer Learning

```
from keras.applications.resnet_v2 import ResNet50V2

def resnet_model_tf(input_shape=(128, 128, 3), nombre_classes=500):
    resnet = ResNet50V2(weights='imagenet', include_top=False, input_shape=input_shape)
    resnet.tbatch_sizenable = False
    model = Sequential()
    model.add(resnet)
    model.add(Flatten())
    model.add(Dense(4096, activation='LeakyReLU'))
    model.add(Dropout(0.5))
    model.add(Dense(4096, activation='LeakyReLU'))
    model.add(Dropout(0.5))
    model.add(Dense(nombre_classes, activation='softmax'))

    print(model.summary())
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                                patience=2,
                                                verbose=1,
                                                factor=0.7,
                                                min_lr=0.0000000001)
    return model, learning_rate_reduction
```

### Evaluation of the model pretrained

```
val = model.evaluate(test_ds)
input_shape = (128, 128, 3)
epochs = 20
batch = 32

print("\n ===== Evaluation : Resnet 50 model ===== \n")
print(" With : \n")
print("Batch size      : {} | Epochs      : {}".format(batch, epochs))
print("Nombres de classes : {} | Input shape : {} \n".format(len(train_ds.class_names), input_shape))
print("\n =====\n")

print(" Results : \n")
print("Loss : {:.2f}%".format(val[0] * 100))
print("Score : {:.2f}%".format(val[1] * 100))
```

```
29/29 [=====] - 4s 121ms/step - loss: 0.0095 - accuracy: 0.9978
```

```
===== Evaluation : Resnet 50 model =====
```

```
With :
```

```
Batch size      : 32      | Epochs      : 20
Nombres de classes : 500    | Input shape : (128, 128, 3)
```

```
=====
```

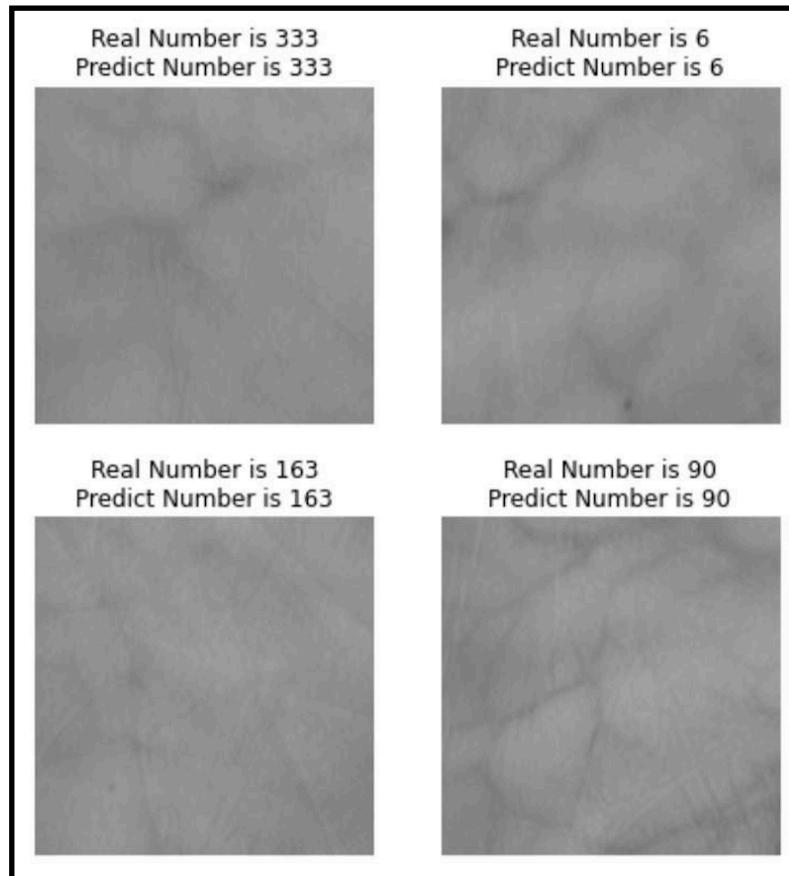
```
Results :
```

```
Loss : 0.95%
Score : 99.78%
```

**ResNet-50** a donc réussi à prédire avec une grande confiance (99.78%) que l'image donnée en entrée correspond à une image de veines palmaires d'un utilisateur.

### Plot images with prediction

```
plt.figure(figsize=(22, 24))
lass_names = data.class_names
for images, labels in data.take(1):
    for i in range(32):
        ax = plt.subplot(6, 6, i + 1)
        y_pred = model.predict(images)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title("Real Identity is {}\nPredict Identity is {}".format(np.argmax(labels[i]), y_pred[i].argmax()))
        plt.axis("off")
```



## B. Résultat - Semestre 2

Jusqu'ici les modèles CNN que nous avons utilisés étaient des modèles connus et disponibles depuis la librairie Keras. Le Transfer Learning nous a permis d'obtenir des résultats très convaincants.

Le but ultime pour ce second semestre sera d'obtenir des résultats semblables sans l'utilisation de Transfer Learning.

Ainsi , nous commencerons par travailler sur des modèles CNN plus simple (1,2, 3 couches).

### Entraînement d'un modèle à une couche

Les détails de notre modèle :

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 125, 125, 128)	2176
max_pooling2d (MaxPooling2D )	(None, 62, 62, 128)	0
dropout (Dropout)	(None, 62, 62, 128)	0
flatten (Flatten)	(None, 492032)	0
dense (Dense)	(None, 128)	62980224
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 500)	64500
<hr/>		
Total params: 63,046,900		
Trainable params: 63,046,900		
Non-trainable params: 0		

La répartition de notre data :

Train ----- 70% | 4200 photos  
Validation ----- 15% | 900 photos  
Test ----- 15% | 900 photos

```
===== Splitting data =====

X_train shape : (4200, 128, 128, 1) | y_train shape : (4200, 500)

X_val shape   : (900, 128, 128, 1) | y_val shape   : (900, 500)

X_test shape  : (900, 128, 128, 1) | y_test shape  : (900, 500)
```

```
# Compile model
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'],
)
```

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                             patience=2,
                                             verbose=1,
                                             factor=0.7,
                                             min_lr=0.0000000001)
```

Nous utilisons également un callback moniteur pour nos entraînements,  
« **ReduceLROnPlateau** ».

Cette fonction est d'une grande utilité car elle nous permet d'optimiser un hyper-paramètre, le « Learning rate », qui au cours de l'entraînement baissera la valeur de celle-ci.

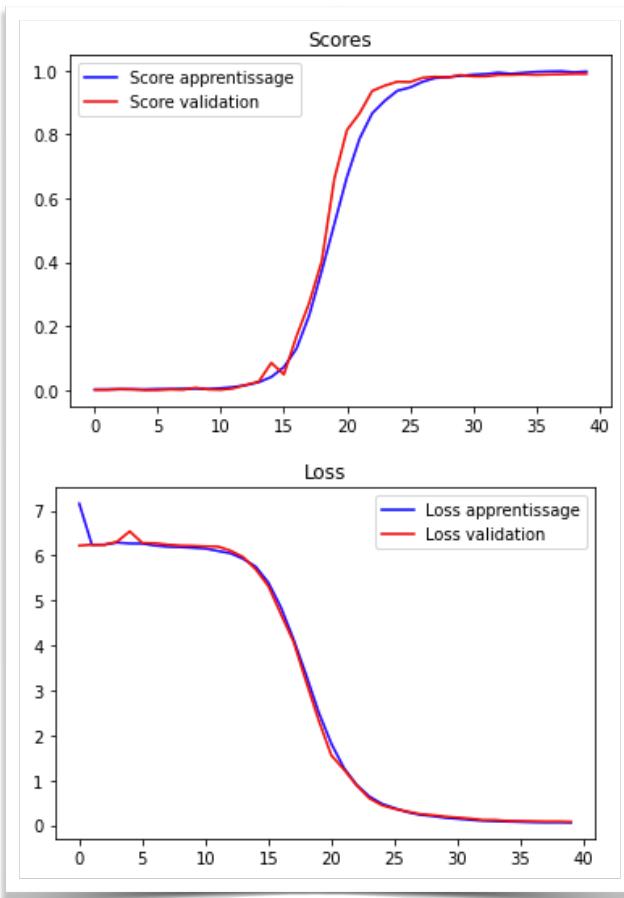
Ici, la fonction s'activera et baissera la valeur du learning rate lorsque la valeur de « val-accuracy » ne change pas pendant 2 epochs à la suite.

Nous avons donc entraîné notre modèle à une couche pendant 40 epochs avec une batch-size à 64. Contrairement à toute attente, nous avons obtenu un résultat très performant qui serait même meilleur que ceux des modèles entraînés avec transfer learning.

```
===== Evaluation : 1 layer model =====
```

Results :

Loss : 0.09  
Score : 99.11%



Comme vous pouvez le constater sur ces graphes, nous obtenons une courbe quasi-parfaite.

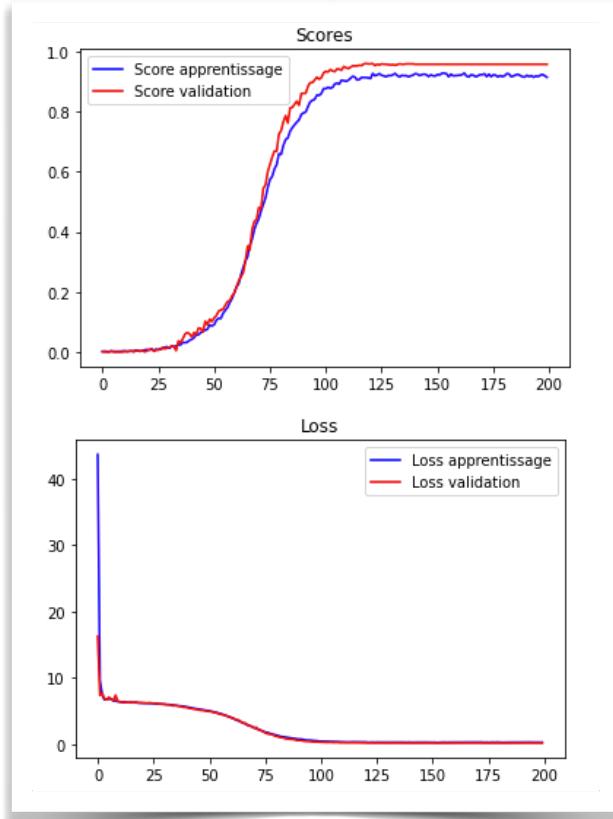
Les graphes que nous avons obtenu avec des modèles de transfer learning n'ont pas été fluides comme celle-ci.

Ces résultats ont été la surprise de notre projet. Nous obtenons des performances très impressionnant et cela avec peu de période.

Nous sommes plus que satisfaits de ces résultats, mais nous ne pouvons pas nous arrêter là. Nous avons donc décidé de continuer sur ce chemin et de continuer, peu à peu, à augmenter la profondeur de notre modèle.

## Entraînement d'un modèle à deux couches

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 122, 122, 96)	4800
max_pooling2d (MaxPooling2D)	(None, 60, 60, 96)	0
lambda (Lambda)	(None, 60, 60, 96)	0
conv2d_1 (Conv2D)	(None, 56, 56, 256)	614656
max_pooling2d_1 (MaxPooling2D)	(None, 27, 27, 256)	0
lambda_1 (Lambda)	(None, 27, 27, 256)	0
dropout (Dropout)	(None, 27, 27, 256)	0
flatten (Flatten)	(None, 186624)	0
dense (Dense)	(None, 128)	23888000
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 500)	64500
<hr/>		
Total params:	24,571,956	
Trainable params:	24,571,956	
Non-trainable params:	0	
<hr/>		



Nous avons rajouté une couche de CONV2D suivi d'un MAXPOOL.

Pour cet entraînement, nous avons gardé les mêmes configurations que pour le modèle à 1 couche, avec 200 epochs.

**Results :**

**Loss : 0.16**  
**Score : 96.44%**

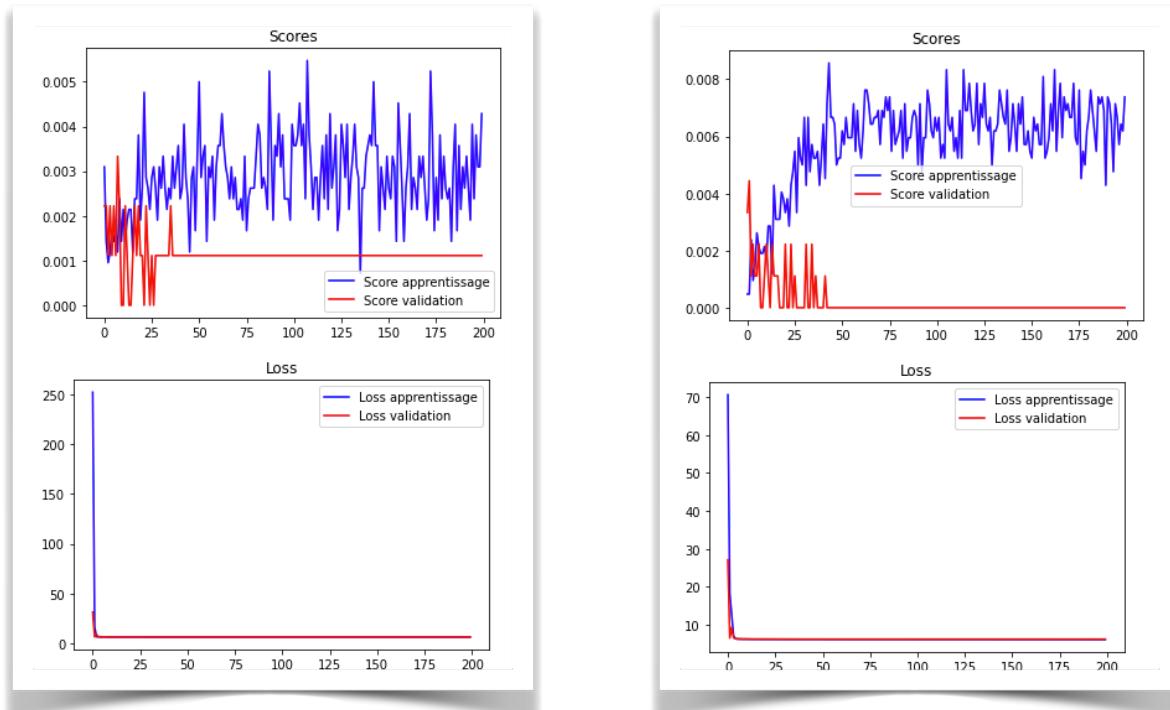
Le résultat obtenu est également satisfaisant.

Nous remarquons une baisse d'accuracy, mais nous obtenons une courbe très stable.

L'absence d'overfit nous rassure sur la base de notre modèle jusqu'à présent fonctionnel.

## Entraînement d'un modèle à trois couches

Lors de l'entraînement de notre modèle à trois couches, les résultats sont en chutes libre et nous montre des performances non-évolutives.



Nous avons essayé plusieurs combinaisons possibles lors de l'ajout d'une troisième couche. Cependant, comme vous pouvez le constater sur les graphes au-dessus, les courbes n'arrivent pas à augmenter et obtenir une forme d'évolution au cours des epochs.

## Entraînement de modèles CNN (fournis par Keras)

Le callbacks « **ReduceLROnPlateau** » permet de réduire le « **Learning Rate** » au fur et à mesure de l'entraînement. Cette fonctionnalité nous permet d'affiner et d'améliorer potentiellement les performances du modèle.

Après avoir observer l'utilité du callbacks « ReduceLROnPlateau », nous allons retravaillé sur les modèles CNN du semestre 1 accompagnée de cette fonction.

Les informations à savoir sur les entraînements de nos modèles :

- Répartition du data :

- ▶ Train ----- 70% - 4200 images
- ▶ Validation ----- 15% - 900 images
- ▶ Test ----- 15% - 900 images

- Input shape : (128, 128, 1)

- Optimizer : « nadam »

- Epochs : 60~200

- Batch size : 64

- Callbacks : ReduceLROnPlateau

Modèles	Accuracy	Loss	Training duration	params
1 couche	99.11%	0.09	~20 minutes	63 millions
2 couches	96.44%	0.16	36min - 200epochs	~24 millions
3 couches	~ 0.11%	~ 6.24	~24 minutes	33~77 millions
ZFNet	~ 0%	~ 6.30	~150 minutes	28~237 millions
ResNet 50	99.22%	0.15	64 minutes	58 millions
ResNet 152	96.11%	1.87	100 minutes	75 millions
EfficientNetB0	99.22%	0.19	34 minutes	14 millions
Xception	99.89%	0.01	40 minutes	37 millions
VGG19	~ 0%	~ 6.32	140 minutes	30 millions
InceptionResNet	99.67%	0.01	107 minutes	62 millions
DenseNet201	99.00%	0.09	160 minutes	51 millions
NASNetLarge	X	X	X	X

Pour le cas du NASNetLarge, nous avons pas réussi à le faire entraîner par soucis de manque de mémoire RAM 8go. Nous avons donc du faire attention sur le nombre de paramètres entraînable pour nos entraînement.

Nous avons utilisé un callbacks « **ReduceLROnPlateau** » qui permet de réduire le « Learning Rate » au fur et à mesure de l'entraînement. Cette fonctionnalité nous permet d'affiner et d'améliorer potentiellement les performances du modèle. C'est une fonction de recherche d'hyper-paramètre qui s'est avéré très utile tout au long du projet.

À titre d'information, les entraînements de chaque modèle présent dans ce tableau s'est effectué sans overfitting ce qui nous a empêché de perdre du temps lors de la comparaison entre tous les modèles. Excepté dans le cas du ZFNet dont on a également lancé un entraînement avec plus de 10 000 epochs mais qui s'est terminé sur un échec avec le même résultat que dans le tableau ci-dessus.

Parmi les 6 modèles suivants :

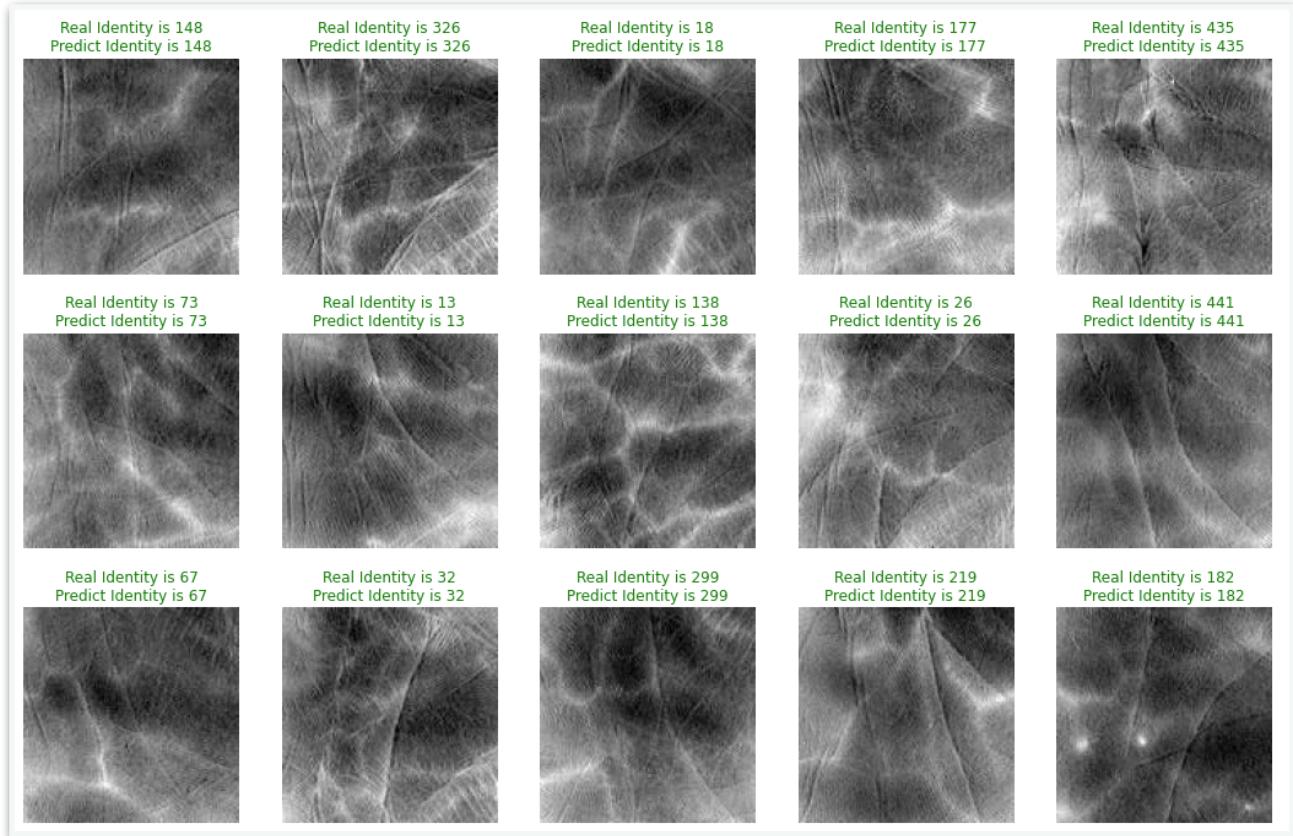
- ResNet50
- ResNet152
- EfficientNetB0
- Xception
- InceptionResNet
- DenseNet201

Nous obtenons la moyenne suivante :

- Loss = 0.39%
- Accuracy = 98.85%
- Training duration = 84 minutes

Avec comme meilleur modèle : **Xception**

- Loss = 0.01%
- Accuracy = 99.89%
- Training duration = 40 minutes



## II - Application Web : Palm Vein Recognition

Les résultats précédents ont permis de réaliser la dernière partie de notre projet : le développement d'une application Web.

L'objectif de cette application sera d'identifier un utilisateur après qu'il ait chargé l'image de son empreinte palmaire.

### 1 - Une partie Backend

Dans cette partie, nous allons vous montrer essentiellement comment déployer une application Web.

#### Docker et Flask



Docker est une plate-forme logicielle , Flask est un micro framework open-source de développement web. Ces deux outils vont nous permettre de concevoir, tester et déployer notre application rapidement.

## i. Création d'un fichier python et d'un fichier html

Tout d'abord, nous allons créer un fichier python « **app.py** » ainsi que d'un fichier html pour la mise en page « **home.html** ».

Tous les fichiers « **.html** » doivent être mises dans un dossier que vous devez créer qui doit s'appeler « **templates** ». Flask va lire les fichiers html en cherchant dans ce dossier.

```
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5
6  @app.route('/')
7  def home():
8      return render_template('home.html')
9
10
11 if __name__ == "__main__":
12     app.run(host="0.0.0.0", debug=True)
13
```

**app.py**

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Flask Docker</title>
6  </head>
7  <body>
8      <h1>Hello, Flask Docker!</h1>
9      <p>
10         This is the default home page for the Flask Docker application.
11     </p>
12
13 </body>
14 </html>
```

**home.html**

Nous allons également créer un fichier « **requirements.txt** ». Ce fichier servira à installer des frameworks dont on aura besoin ici. Ce fichier sera lancé à chaque build du docker-compose. Pour l'instant, on a besoin que de Flask.

```
1 | Flask==2.0.2
2 |
```

## requirements.txt

### ii. Notre image Docker

Maintenant, nous allons créer deux fichiers en lien avec docker : « Dockerfile » et « docker-compose.yml ».

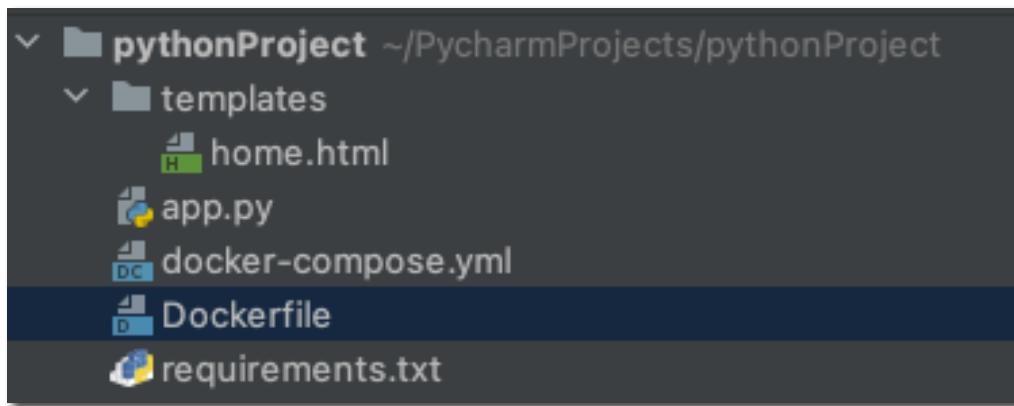
```
1 | version: '3'
2 |
3 |> services:
4 |>   web:
5 |>     build:
6 |>       context: .
7 |>       dockerfile: Dockerfile
8 |>     ports:
9 |>       - "5000:5000"
10|>    expose:
11|>      - "5000"
12|>    volumes:
13|>      - ./code
```

## docker-compose.yml

```
1 ► FROM python:3.8-alpine
2 | └─
3 |   ADD . /code
4 |   WORKDIR /code
5 |
6 | RUN pip install -r requirements.txt
7 |
8 | COPY . .
9 |
10| CMD [ "python", "app.py" ]
11|
12| EXPOSE 5000
13|
```

## Dockerfile

Voici à quoi doit ressembler votre arbre hiérarchique:

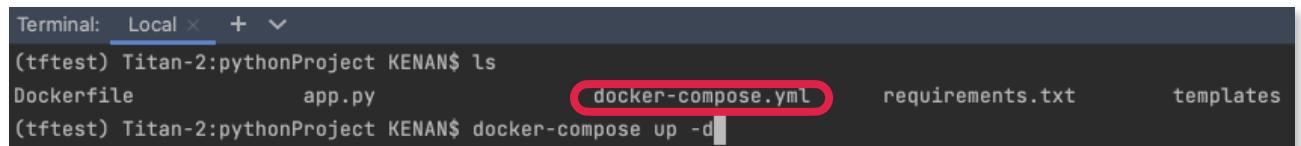


### **iii. Démarrer Docker**

Maintenant que tous les fichiers nécessaires pour le démarrage ont été créés, nous pouvons donc lancer notre application.

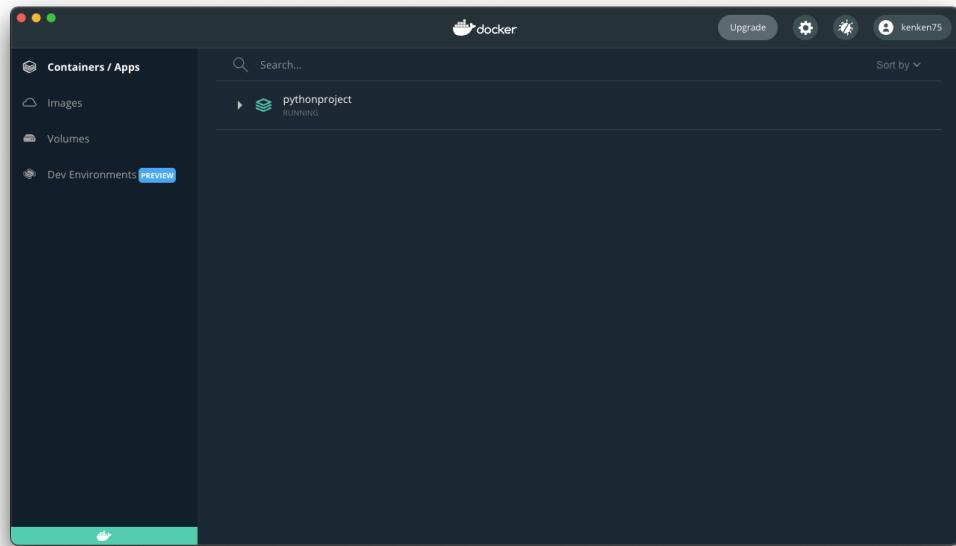
Vous devez écrire, dans votre terminal, la commande suivante :

**docker-compose up -d**



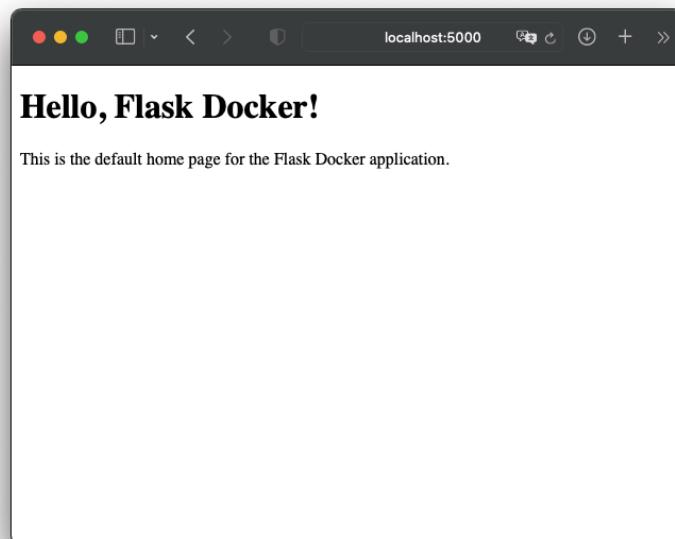
```
Terminal: Local + ▾
(tftest) Titan-2:pythonProject KENAN$ ls
Dockerfile           app.py          docker-compose.yml      requirements.txt      templates
(tftest) Titan-2:pythonProject KENAN$ docker-compose up -d
```

Vous devez vous trouver dans le même dossier où se trouve le fichier « docker-compose.yml ». Cette commande va lancer ce fichier ce qui va démarrer notre application.



Si dans votre Docker Desktop, vous voyez cette icône, c'est que votre application s'est démarrer avec succès. Cependant, ceci ne veut pas dire que votre application va fonctionner comme vous l'entendez.

Maintenant vous pouvez aller sur : localhost:5000/



#### **iv. Déploiement du modèle entraîné**

Maintenant que nous avons créé les bases, nous allons maintenant déployer notre modèle CNN de la partie « entraînement de nouveau modèle sans transfer learning » dans cette application.

Pour le choix du modèle, nous allons prendre **le modèle à 1 couche** car il est l'un des modèles plus légers et plus performants que nous avons.

Nos modèles, dont on estime qu'ils sont très bons, sont sauvegardés dans nos bases de données. Lorsque nous sauvegardons nos modèles, nous sauvegardons d'une part l'architecture du modèle (.json) ainsi que de leurs poids (.h5). À partir de ces fichiers, nous pouvons récupérer nos modèles avec leur connaissances intactes. Ces fichiers sont stockés dans le dossier « saved\_models ».

C'est dans le fichier « app.py » que nous allons faire toutes nos opérations de l'application web.

```
16 # Load the trained model
17 model = model_from_json(model_architecture_path)
18 model.load_weights(model_weights_path)
19 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Voici les trois lignes qui permettent de charger nos modèles pré-entraînés.

Le « model\_from\_json » est une fonction de Keras.models qui permet de charger l'architecture du modèle.

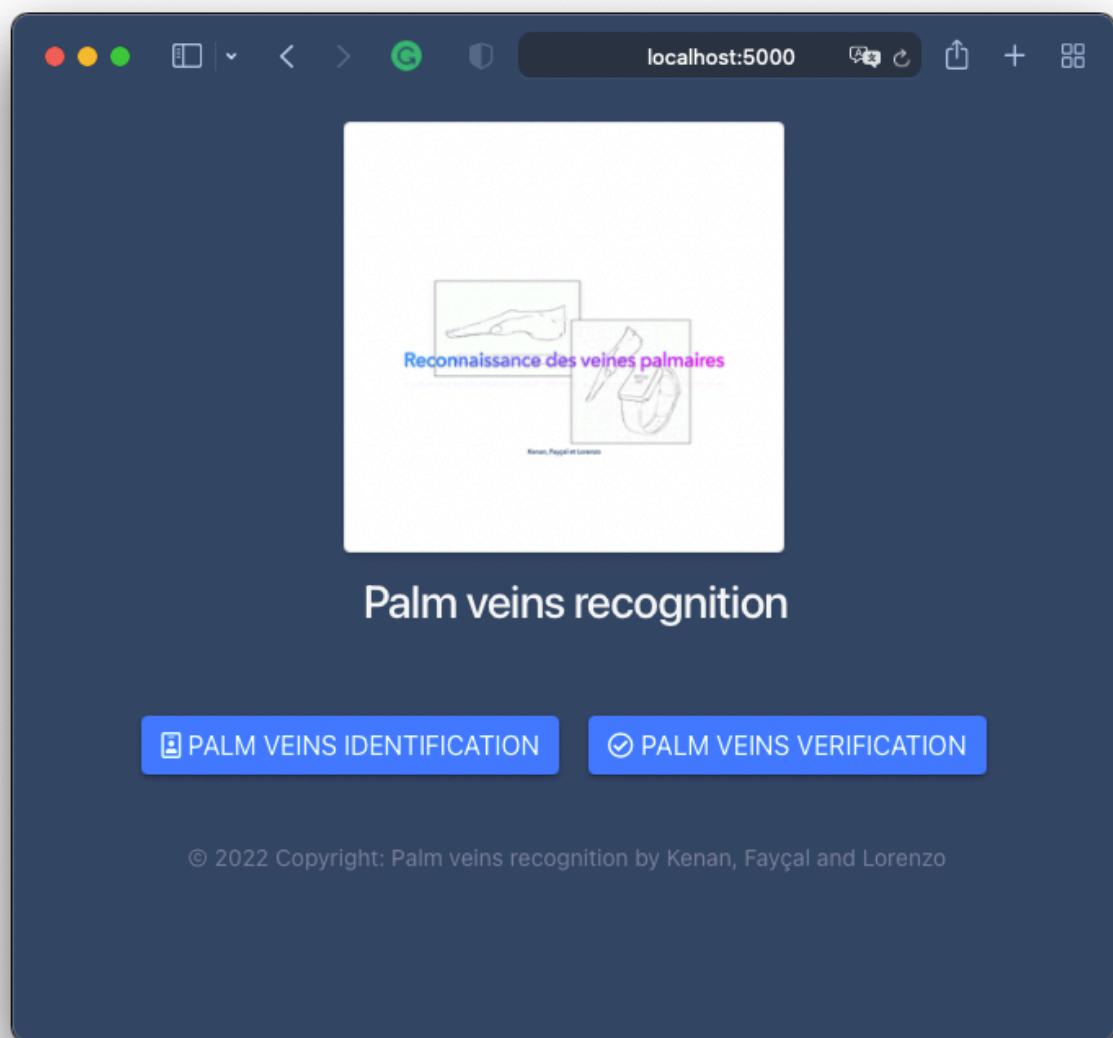
Et le « model.load\_weights » permet donc de charger les poids de ce modèle. Et après avoir compilé, le modèle devient donc apte à être utilisé pour notre application web.

Cette facilité à déployer un modèle CNN est surtout dû au fait que nous avons choisi Flask, une application web codé en Python.

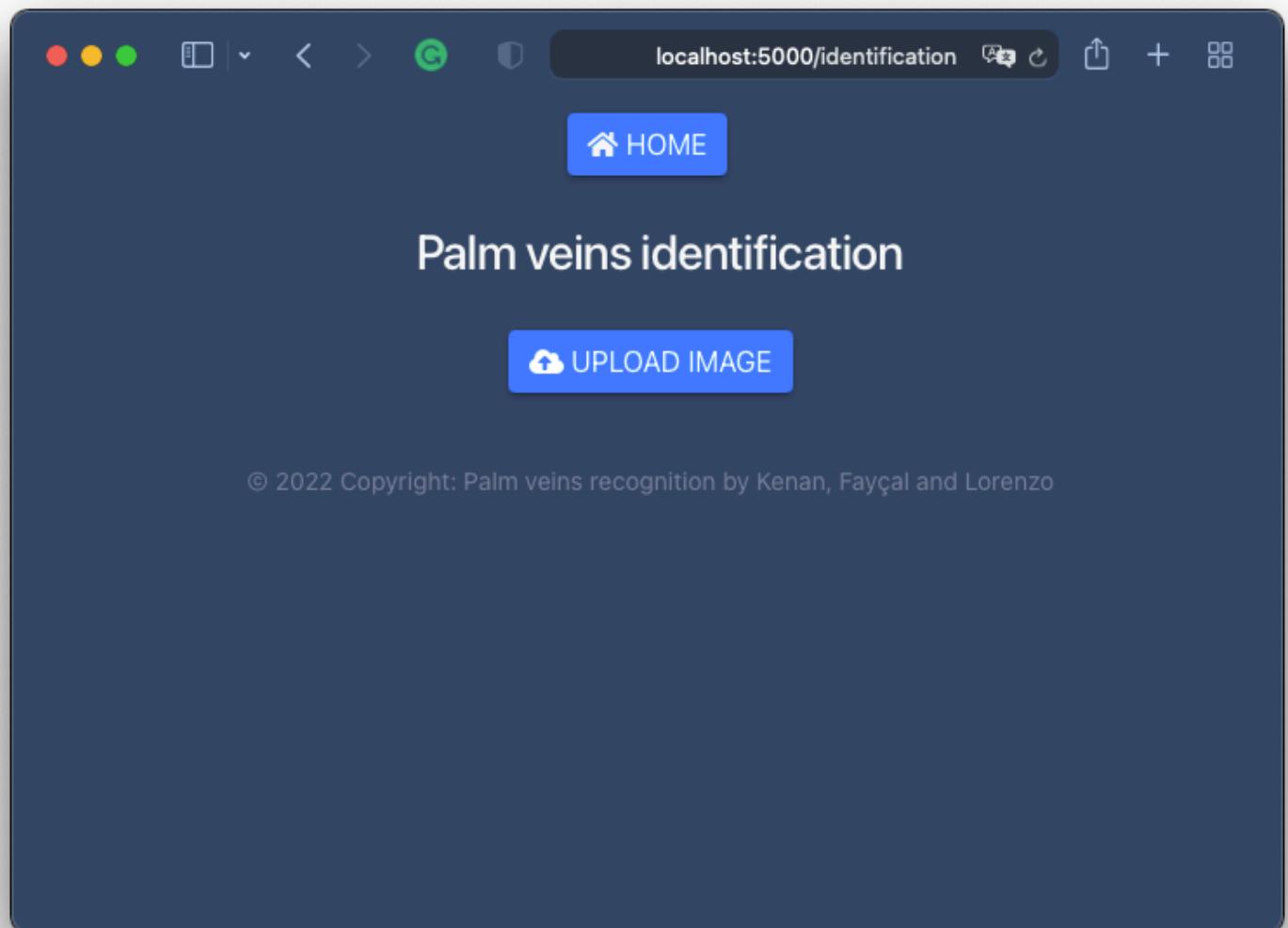
## 2 - Une partie Frontend

Voici les différentes interfaces de notre application développé avec HTML.

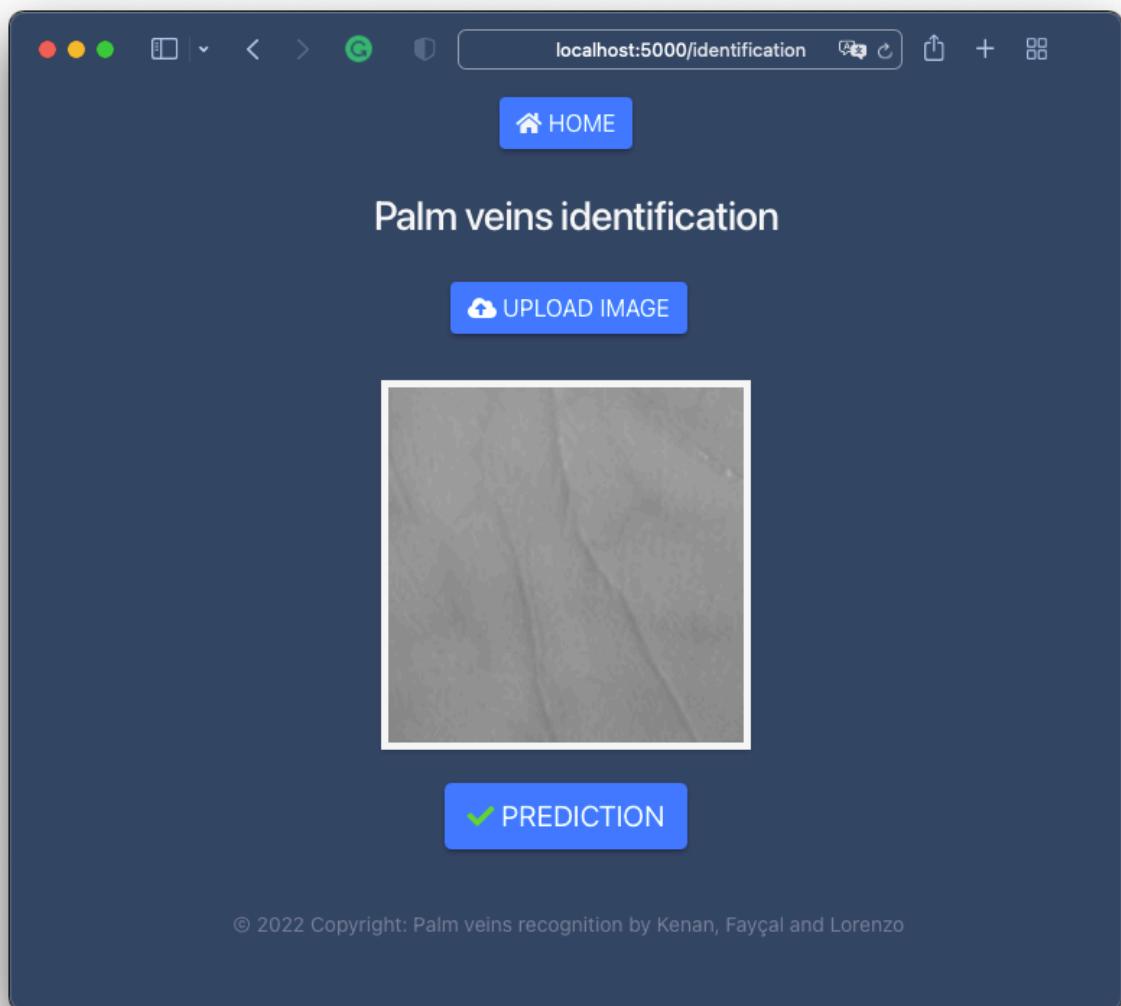
### Interface 1 : Palm Vein Identification



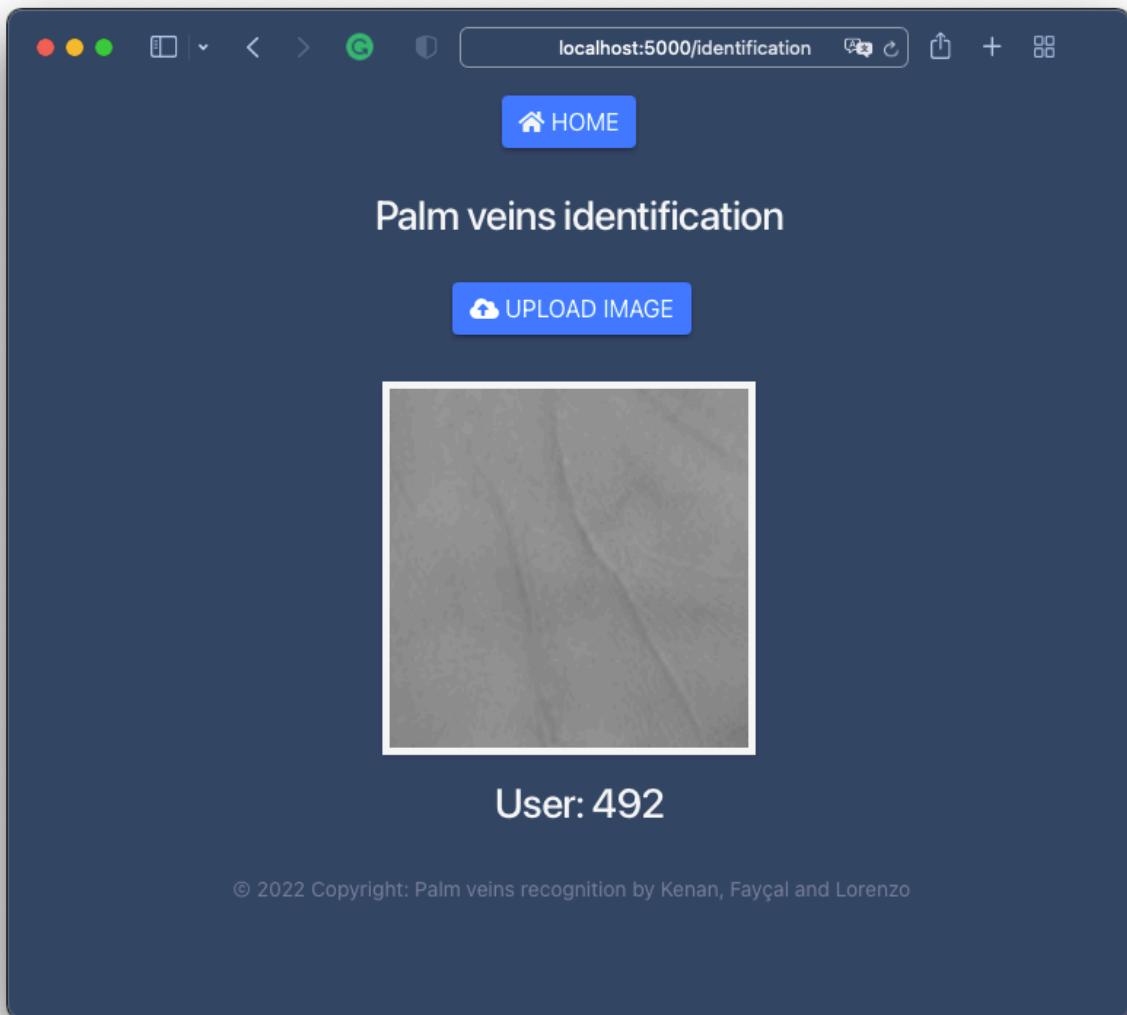
## Interface 2 : Upload your palm vein



### Interface 3 : Prediction identification



## Interface 4 : Identified user



## Conclusion:

Dans ce chapitre, nous avons présenté les étapes clés de notre première application. L'entraînement de plusieurs modèles CNN nous permet de juger que le Transfer Learning est d'une grande utilité.

Cependant, nous voulions obtenir des résultats positifs sans l'utilisation du Transfer Learning.

La rencontre que nous avons faites avec le callback « ReduceLROnPlateau » était parfaite puisque nous avons pu entraîné des modèles simples (1,2 couches) et d'autres modèles tel que Xception avec une accuracy de 99.89%.

Et pour terminer, nous avons déployé une application Web permettant l'identification d'utilisateur par la reconnaissance d'images de veines palmaires.

## Conclusion générale

Ce projet nous a permis de mieux comprendre comment la technologie derrière la reconnaissance biométrique des veines palmaires fonctionne dans son architecture globale, ainsi que la place de l'intelligence artificielle dans celle-ci.

Nous avons aussi vu comment effectuer des recherches dans le domaine des modèles de réseaux de neurones, comment les comparer et comment les paramétrer.

Les résultats que nous avons obtenus ont permis la création d'une application en vue d'une démonstration.