INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE

KHONG
THANH DUY

Rapport CONFIDENTIEL
(cocher la case correspondante)

☒ NON
☐ **OUI**

Si oui est coché, signature du tuteur entreprise obligatoire

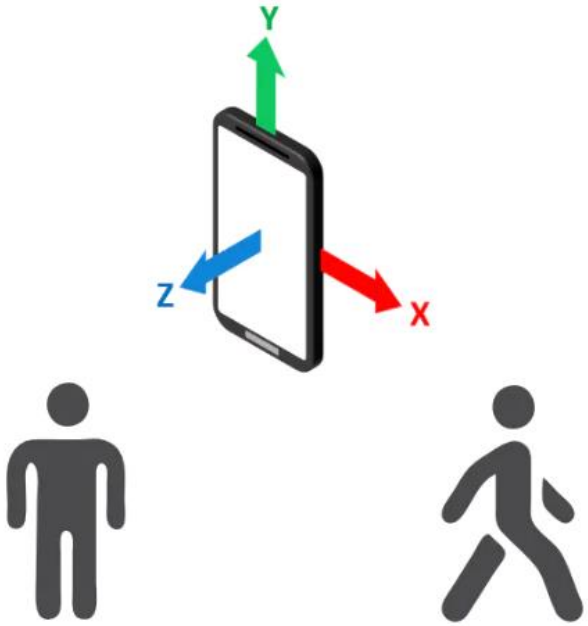
4A GSI

Année Universitaire :

2021-2022

Application de Machine Learning dans la secteur de reconnaissance de l'activité humaines à base de capteurs corporels

RAPPORT DE STAGE



Tuteur Laboratoire :

HO Thi Thao Nguyen

Assistante de lecture

Enseignant référent :

JULIEN Mille
Lecturer

AALBORG UNIVERSITE
Aalborg 9220, Denmark

Remerciement

Je tiens à remercier toutes les personnes dans l’qui m’ont aidée et ont rendu mon environnement de travail très confortable et agréable pendant mon stage.

Tout d'abord, j'aimerais adresser mes remerciements à mon tuteur de stage Madame Thi Thao Nguyen HO, pour m'avoir intégré rapidement au sein de l'entreprise et pour ses conseils techniques tout le long du stage.

Je remercie également Monsieur Peter, le Président – Directeur Général de l'université de Aalborg, pour m'avoir permis de réaliser ce stage.

Je tiens aussi à remercier mon tuteur de l'école Monsieur Julien Mille et le Service Relation International qui ont réalisé ma convention et ont servi d'intermédiaire entre l'INSA Centre Val de Loire et le département de Computer Science de l’université de Aalborg

Résumé

Le stage s'est déroulé en ligne, le stagiaire a effectué le stage par le biais d'appels hebdomadaires pour échanger des connaissances, des tâches et des résultats de travail.

La tâche principale du stage de 3 mois est de : construire un modèle simple qui permet d'identifier 6 activités simples sur la base d'un ensemble de données disponibles.

Les résultats du stage ont répondu aux exigences énoncées au départ, le modèle fonctionne bien avec de nombreuses donnes différentes et peut être intégré sur des applications mobiles pour une application pratique.

Mots clés: Machine Learning, Human Activity Recognition, Wearable-sensor, Neural network, TensorFlow, Keras, CNN, LSTM, Conv1D, Optimizer, Confusion-matrix.

Abstract

The internship was conducted online, the intern carried out the internship through weekly calls to exchange knowledge, tasks and work results.

The main task of the 3-months internship is to: build a simple model that identifies 6 simple activities based on a set of available data.

The results of the course met the requirements stated at the start, the model works well with many different dataset and can be integrated on mobile applications for practical application.

Mots clés: Machine Learning, Human Activity Recognition, Wearable-sensor, Neural network , TensorFlow,Keras,CNN,LSTM,Conv1D,,Optimizer,Confusion-matrix .

Table des matières

Introduction

Au cours des dernières années, les méthodes traditionnelles de reconnaissance de formes ont fait de grands progrès. Cependant, ces méthodes reposent fortement sur l'extraction manuelle des caractéristiques, ce qui peut entraver les performances du modèle de généralisation. Avec la popularité et le succès croissants des méthodes d'apprentissage en profondeur, l'utilisation de ces techniques pour reconnaître les actions humaines dans des scénarios informatiques mobiles et portables a attiré une large attention. Dans le domaine de la santé humaine, le Machine Learning joue un rôle très important dans le suivi de l'état du patient. Dans ce rapport, je présenterai la base théorique et le processus de construction d'un modèle CNN et LSTM pour réaliser une fonction de reconnaissance de l'activité humaine de base basée sur les données disponibles. Résultats obtenus avec une grande précision de 96,49% avec le modèle CNN et 96.56 % avec le modèle CNN -LSTM.

En tant qu'étudiant étudiant en ACAD, c'est une excellente occasion pour moi d'appliquer ce que j'ai appris à l'école avec mes nouvelles connaissances en python et en apprentissage automatique. Cette connaissance sera une excellente base pour moi pour entrer en 5e année et pour le processus de travail dans ce domaine au plus tard

Développement

1. Présentation de l'entreprise / Cadre du travail :

1.1 Histoire :

L'Université d'Aalborg (AAU) est une grande université publique Danoise avec des campus à Aalborg, Esbjerg et Copenhague fondée en 1974. L'université décerne des diplômes de licence, de maîtrise et de doctorat dans une grande variété de matières en sciences humaines, sciences sociales, technologie de l'information, design, ingénierie, sciences exactes et médecine.

1.2 Géographie

Le stage s'est déroulé au sein du département de Computer Science.

Adresse : Selma Lagerlof Vej 300, Aalborg, Danemark

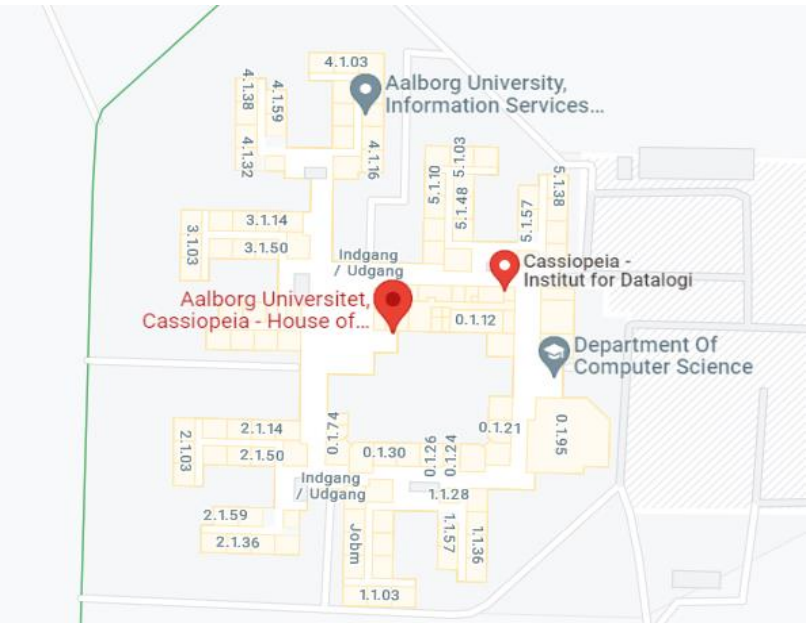


Figure 1: *Emplacement géographique de l'Université de Aalborg*

2. Objectif du stage / Cahier des charges :

2.1 Objectif du stage

La mission principale du stage est d'effectuer 2 opérations majeures :

- Construire des modèles d'apprentissage automatique pour prédire les activités humaines à partir de données précédemment obtenues
- Évaluation de la précision et amélioration du modèle

2.2 Cahier des charges

Le cahier des charges de ma mission est détaillé dans le diagramme de Gantt disponible en Annexe n°1 : Diagramme de Gantt associé au cahier des charges de la mission du stage.

3. Présentation les problématiques et les solutions :

3.1 Problématique :

La reconnaissance de l'activité humaine est l'une des principales applications du Machine Learning. Dans ce projet, nous allons construire un modèle et l'entraîner à reconnaître les actions humaines sous une forme supervisée (données disponibles avec des étiquettes).

Le problématique posé :

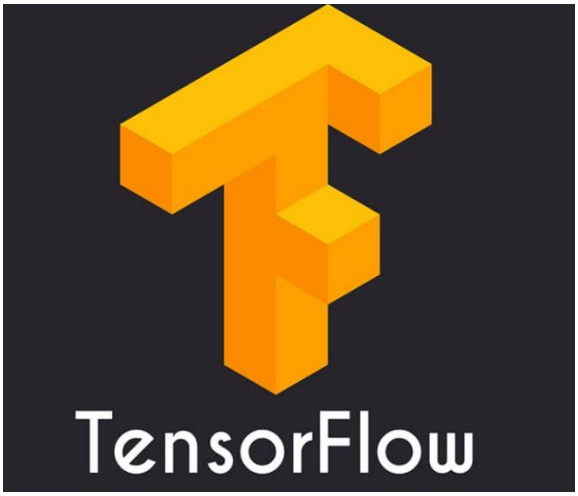
- Le stagiaire doit être doté d'un certain nombre de connaissances en mathématiques et en programmation, être capable de construire des idées et de simuler des modèles, comprendre parfaitement la structure du modèle.
- Le modèle doit avoir une précision élevée (> 90 %) pour pouvoir être appliqué dans la pratique, par exemple, une application de surveillance de la santé humaine.

3.2 Choix technologies :

La nature de l'apprentissage automatique est un défi extrêmement difficile, nécessitant une connaissance approfondie des mathématiques et de la programmation pour pouvoir créer un modèle d'IA à partir de zéro. Bien sûr, un étudiant de 4e année comme moi ne serait pas capable de faire ça. Par conséquent, dans ce projet, nous utiliserons les API disponibles.

- **Tensorflow**

TensorFlow est une bibliothèque de logiciels gratuite et open source pour l'apprentissage automatique et l'intelligence artificielle. Il peut être utilisé dans une gamme de tâches, mais met particulièrement l'accent sur la formation et l'inférence des réseaux de neurones profonds.



TensorFlow a été développé par l'équipe Google Brain pour une utilisation interne de Google dans la recherche et la production. La version initiale a été publiée sous la licence Apache 2.0 en 2015. Google a publié la version mise à jour de TensorFlow, nommée TensorFlow 2.0, en septembre 2019.

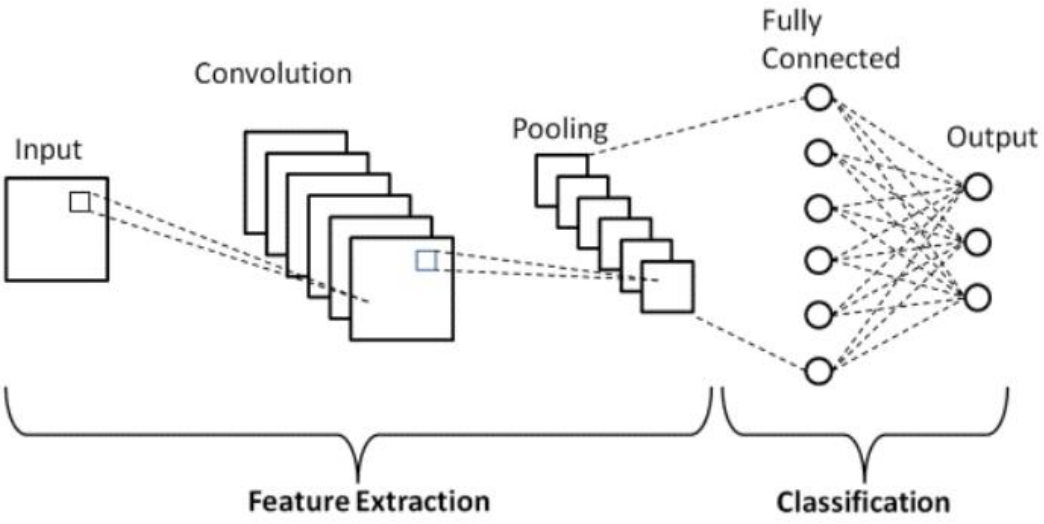
TensorFlow peut être utilisé dans une grande variété de langages de programmation, notamment Python, ainsi que Javascript, C++ et Java.Cette flexibilité se prête à une gamme d'applications dans de nombreux secteurs différents.

La version utilisée dans ce projet est la 2.9.0

3.3 Sélection d'algorithme

- **CNN**

CNN (Convolutional Neural Network en Anglais) ou '' Réseau Neuronal Convolutif '' en Français est une forme spéciale du réseau neuronal artificiel. Il s'agit d'une structure particulière d'un réseau de neurones artificiels spécialement conçu pour l'apprentissage automatique et le traitement d'images ou de données audio. Dans notre cas, nous l'utiliserons pour gérer le changement de coordonnées d'un appareil mobile sur le corps.

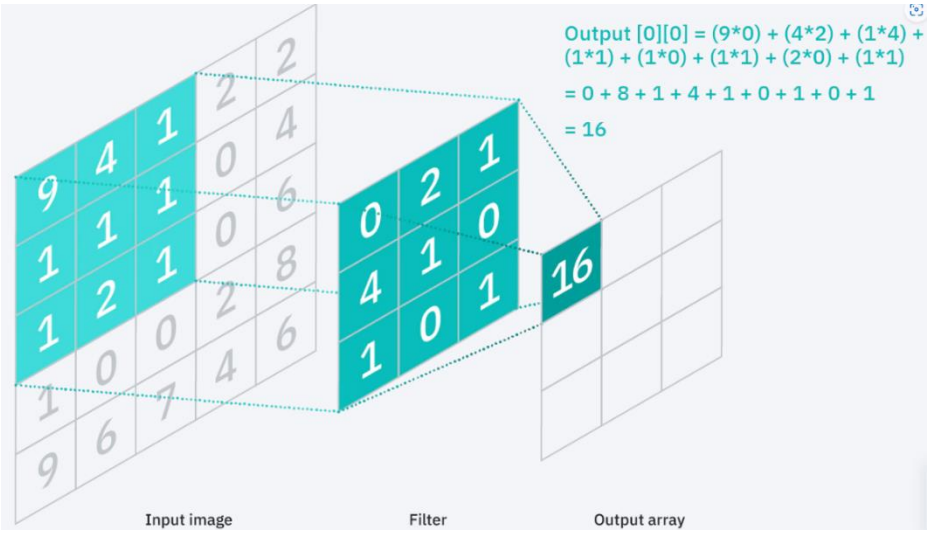


Le fonctionnement du réseau CNN est simulé sur la base du réseau neuronal du cerveau humain. Le CNN se compose de plusieurs couches.

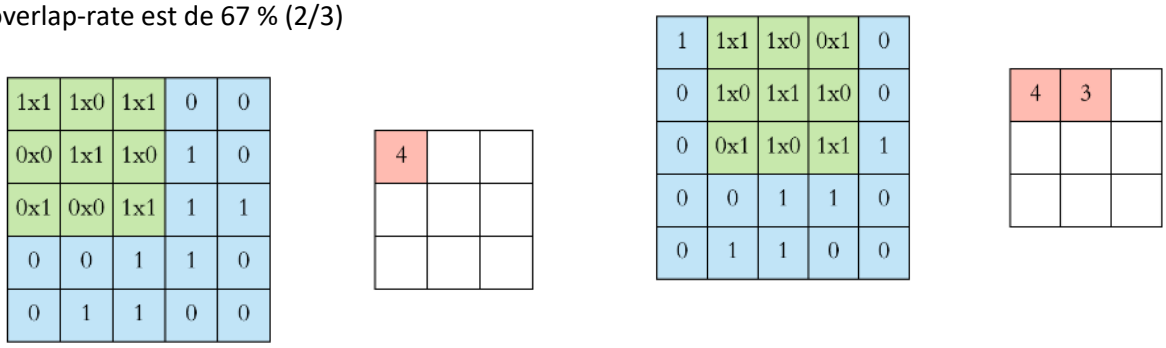
Il existe trois types de couches qui composent le CNN, à savoir les couches convolutionnelles, les couches de regroupement et les couches entièrement connectées (FC). Lorsque ces couches sont empilées, une architecture CNN sera formée. En plus de ces trois couches, il existe deux paramètres plus importants qui sont la couche de décrochage et la fonction d'activation qui sont définis ci-dessous.

- **Couche de convolutionnelles (Convolution layer)**

Cette couche est la première couche utilisée pour extraire les différentes caractéristiques des donnes d'entrée. Dans cette couche, l'opération mathématique de convolution est effectuée entre l'image d'entrée et un filtre d'une taille particulière MxM. En faisant glisser le filtre sur l'image d'entrée, le produit scalaire est pris entre le filtre et les parties de l'image d'entrée par rapport à la taille du filtre (MxM).



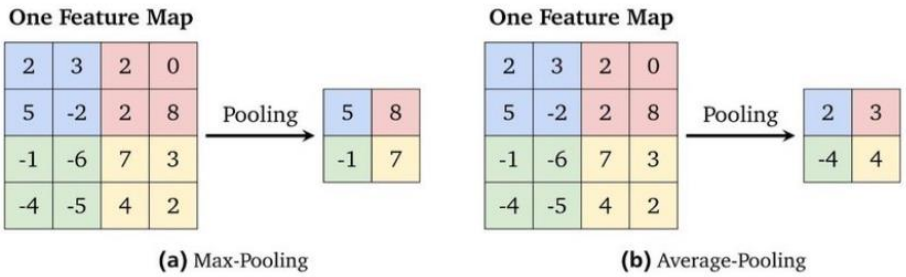
Le but de la couche de convolution est de capturer toutes les caractéristiques de données, de sorte que le filtre glissera avec une foulée plus petite que la taille du kernel. Bien sûr , il y aura duplication de données supplémentaires . Ce rapport est appelé overlap-rate. Dans l'exemple ci-dessous, overlap-rate est de 67 % (2/3)



- **Couche de regroupement (Pooling-Layer)**

Dans la plupart des cas, une couche convolutive est suivie d'une couche de regroupement. L'objectif principal de cette couche est de réduire la taille de la carte d'entités convoluées afin de réduire les coûts de calcul. Ceci est effectué en diminuant les connexions entre les couches et fonctionne indépendamment sur chaque carte d'entités. Selon la méthode utilisée, il existe plusieurs types d'opérations de regroupement. Il résume essentiellement les caractéristiques générées par une couche de convolution.

Dans Max Pooling, le plus grand élément est extrait de la carte des fonctionnalités. La Average-Pooling calcule la moyenne des éléments dans une section de taille prédéfinie. La somme totale des éléments de la section prédéfinie est calculée dans Sum Pooling. La couche de regroupement sert généralement de pont entre la couche convolutive et la couche FC.



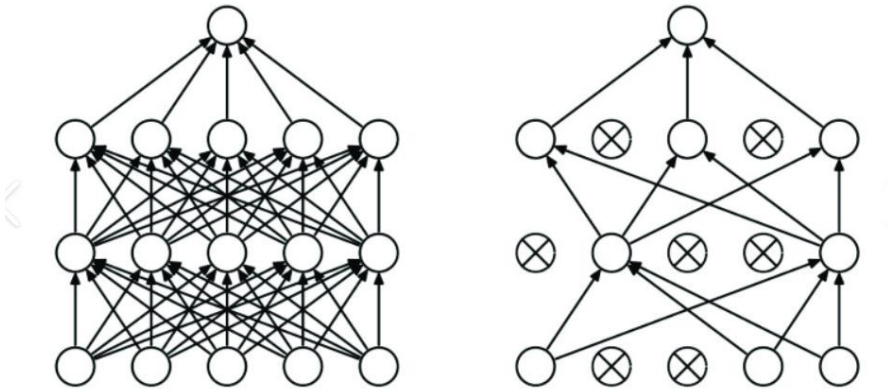
Ce modèle CNN généralise les caractéristiques extraites par la couche de convolution et aide les réseaux à reconnaître les caractéristiques indépendamment. Grâce à cela, les calculs sont également réduits dans un réseau.

- **Couche de Décrochage (Dropout Layer)**

Avant de se connecter au réseau, nous devons faire une dernière étape, le décrochage. Une couche d'Décrochage prend la sortie des activations de la couche précédente et définit de manière aléatoire une certaine fraction (taux d'abandon) des activations à 0, en les annulant ou en les « abandonnant

». Il s'agit d'une technique de régularisation courante utilisée pour éviter le surajustement dans les réseaux de neurones.

Le surajustement se produit lorsqu'un modèle particulier fonctionne si bien sur les données d'entraînement, ce qui a un impact négatif sur les performances du modèle lorsqu'il est utilisé sur de nouvelles données.



(a) Standard Neural Network (b) Neural Net with Dropout

- **Couche d'activation :**

L'un des paramètres les plus importants du modèle CNN est la fonction d'activation. Ils sont utilisés pour apprendre et approximer tout type de relation continue et complexe entre les variables du réseau. En termes simples, il décide quelles informations du modèle doivent être déclenchées dans le sens direct et lesquelles ne doivent pas se déclencher à la fin du réseau.

Il ajoute de la non-linéarité au réseau. Il existe plusieurs fonctions d'activation couramment utilisées telles que les fonctions ReLU, Softmax, tanH et Sigmoïde. Chacune de ces fonctions à un usage spécifique. Pour un modèle CNN de classification binaire, les fonctions sigmoïdes et softmax sont préférées et pour une classification multi-classes, nous utilisons généralement softmax. En termes simples, les fonctions d'activation dans un modèle CNN déterminent si un neurone doit être activé ou non. Il décide si l'apport au travail est important ou non à prédire à l'aide d'opérations mathématiques.

Dans ce projet, nous utiliserons l'activation reLu car c'est la meilleure activation

- **La couche entièrement connectée**

La couche entièrement connectée (FC) comprend les poids et les biais ainsi que les neurones et est utilisée pour connecter les neurones entre deux couches différentes. Ces couches sont généralement placées avant la couche de sortie et forment les dernières couches d'une architecture CNN.

Dans ce cas, l'donnes entrée des couches précédentes est aplatie et transmise à la couche FC. Le vecteur aplati subit ensuite quelques couches FC supplémentaires où les opérations des fonctions mathématiques ont généralement lieu. À ce stade, le processus de classification commence à avoir lieu. La raison pour laquelle deux couches sont connectées est que deux couches entièrement connectées fonctionneront mieux qu'une seule couche connectée.

4. Mise en œuvre du projet

4.1 Bibliothèques utilisées

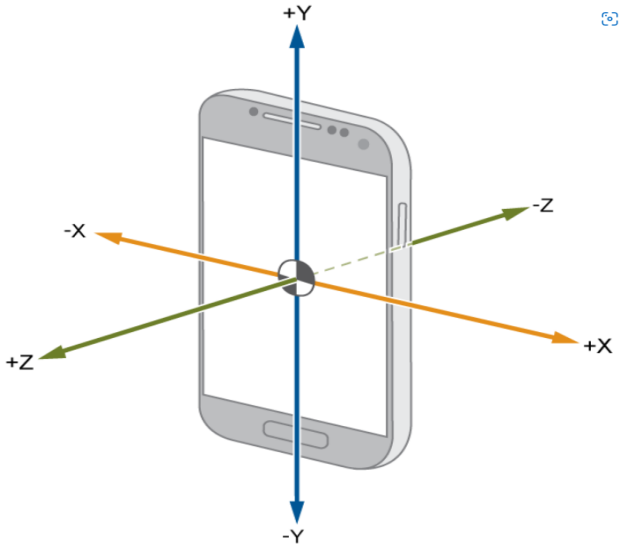
En plus des bibliothèques courantes en python telles que numpy , pandas , ... Dans ce projet , je utilisons la bibliothèque tensorflow et ses sous - bibliothèques . Ce sont des bibliothèques principalement axées sur le Machine Learning, fournissant des outils pour construire et optimiser des modèles de manière simple et facile

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from sklearn import metrics
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout, Conv1D, MaxPooling1D,GlobalAveragePooli
from keras import optimizers
import keras
from keras.utils import np_utils
```

***Figure 2** : Des bibliothèques et outils utilisés dans le projet*

4.2 Base de données

L'ensemble de données se compose de signaux provenant d'un smartphone porté par 36 individus effectuant 6 activités différentes en utilisant des smartphones dans leur poches. L'accéléromètre et le gyroscope à l'intérieur du téléphone enregistreront la variation des coordonnées x, y, z du téléphone pendant leur mouvement. Les données sont enregistrées à une fréquence de 20 Hz (20 valeurs par seconde).



Les activités réalisées sont listées ci-dessous avec leurs codes correspondants.

- Descendre d'escalier (Downstair)
- Monter l'escalier (Upstair)
- Faire du jogging (Jogging)
- S'asseoir (Sitting)
- Se lever (Standing)
- Marcher (Walking)

Après avoir converti les données au format DataFrame, nous obtenons une table de données avec 6 colonnes. Nous utiliserons les données de trois colonnes «x-axis », «y-axis », et «z-axis », comme données pour entrainer le modèle, la ligne "activity" sera utilisée comme étiquette pour entrainer.

```
33,Jogging,49105962326000,-0.6946377,12.680544,0.50395286
33,Jogging,49106062271000,5.012288,11.264028,0.95342433
33,Jogging,49106112167000,4.903325,10.882658,-0.08172209
33,Jogging,49106222305000,-0.61291564,18.496431,3.0237172
33,Jogging,49106332290000,-1.1849703,12.108489,7.205164
33,Jogging,49106442306000,1.3756552,-2.4925237,-6.510526
33,Jogging,49106542312000,-0.61291564,10.56939,5.706926
33,Jogging,49106652389000,-0.50395286,13.947236,7.0553403
33,Jogging,49106762313000,-8.430995,11.413852,5.134871
```

user	activity	timestamp	x-axis	y-axis	z-axis
33	Jogging	49105962326000	-0.694638	12.680544	0.503953
33	Jogging	49106062271000	5.012288	11.264028	0.953424
33	Jogging	49106112167000	4.903325	10.882658	-0.081722
33	Jogging	49106222305000	-0.612916	18.496431	3.023717
33	Jogging	49106332290000	-1.184970	12.108489	7.205164

Figue 3 *: Données brutes et données après conversion au format Dataframe*

Le graphique ci-dessous illustre le nombre d'échantillons dans notre données par activité :

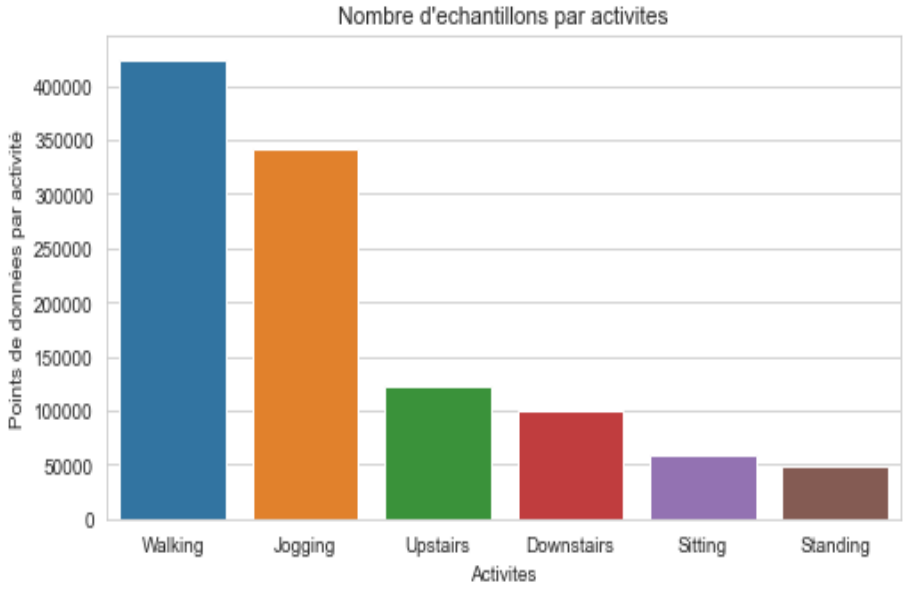
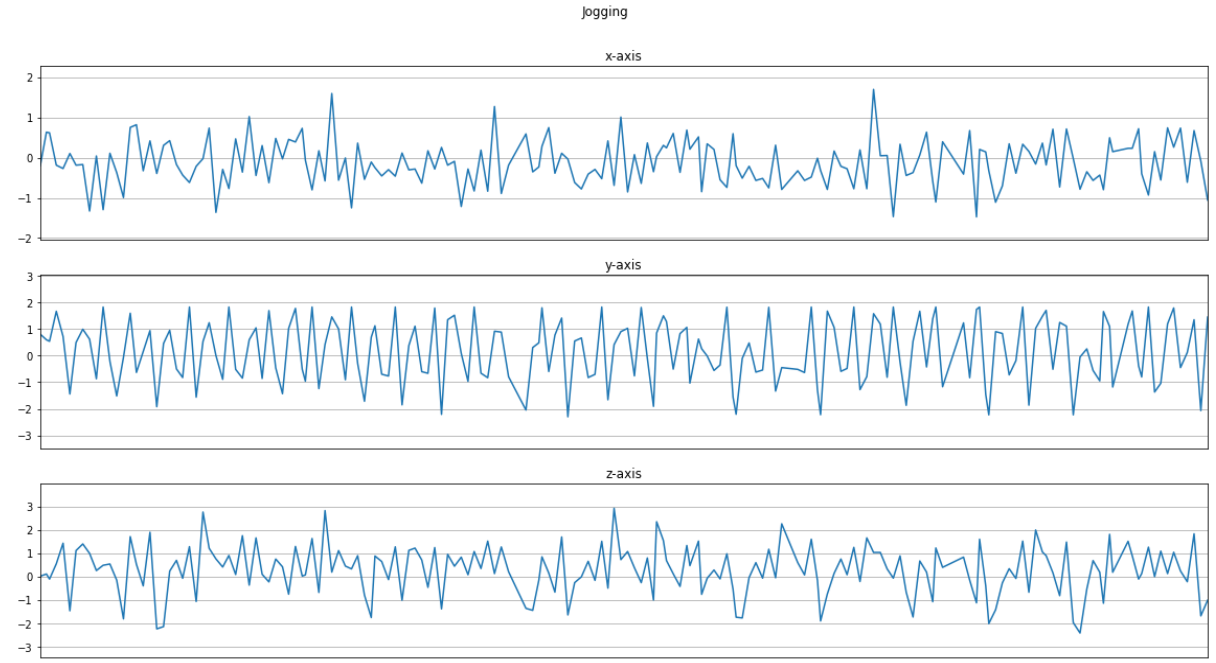


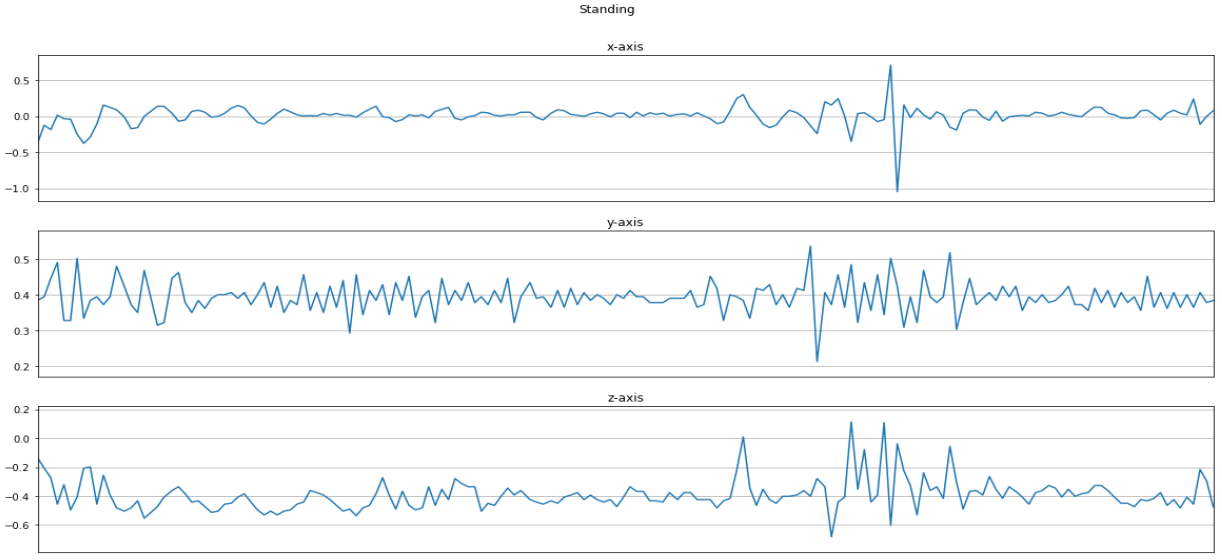
Figure 4 *: Répartition des points de données triés par activité*

Nous constatons que les données obtenues à partir de Marcher et du Jogging sont les plus capturées, car dans ces deux cas, les trois axes de coordonnées du téléphone varient le plus. Nous avons donc besoin de plus de données pour entraîner notre modèle à distinguer ces deux activites.

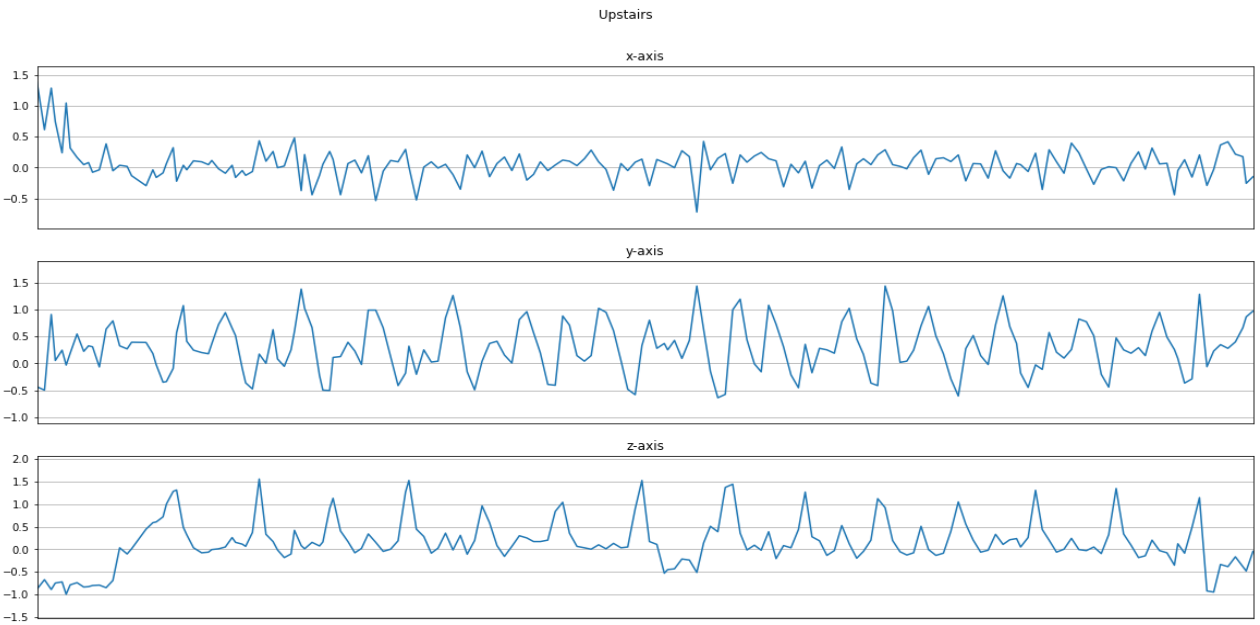
Les diagrammes de variation des coordonnées ci-dessous donneront un aperçu visuel de la différence entre chaque activité sur la base d'une séquence de 180 points de donnes. Notez que toutes les données utilisées sont des données normalisées avec moyenne = 0 et écart type = 1



Figue 5 *: Diagramme décrit la variation de x, y, z de l'activité '' Jogging ''*



Figue 6 *: Diagramme décrit la variation de x,y,z de l'activité '' Se lever ''*



***Figue 7** : Diagramme décrit la variation de x, y, z de l'activité ‘’Monter l’escalier ‘’*

4.3 Segmentation de donnes

Maintenant, nous devons préparer l'ensemble de données dans un format requis par le modèle CNN. Pour ce faire, nous définissons quelques fonctions d'assistance pour créer des segments de taille fixe à partir du signal brut.

Les données normalisées sont ensuite segmentées en tranches de temps de taille de fenêtre 50, ce qui se traduit par des blocs de données de 2.5 secondes (20 valeurs par seconde).

```
step = 10
N_TIME_STEPS = 50
N_FEATURES = 3
segments = []
labels = []

for i in range(0,  dataset.shape[0]-N_TIME_STEPS, step):
    xs = dataset['x-axis'].values[i: i + 50]
    ys = dataset['y-axis'].values[i: i + 50]
    zs = dataset['z-axis'].values[i: i + 50]
    label = stats.mode(dataset['activity'][i: i + 50])[0][0]
    segments.append([xs, ys, zs])
    labels.append(label)

#reshape the segments which is (list of arrays) to one list
reshaped_segments = np.asarray(segments, dtype= np.float32).reshape(len(segments), N_TIME_STEPS, N_FEATURES)
labels = np.asarray(pd.get_dummies(labels), dtype = np.float32)
reshaped_segments.shape
```

```
(109816, 50, 3)
```

Nous allons prendre 50 valeurs (x,y,z) à tour de rôle et ajouter une sous-liste . Le foulée est 10. Le taux de overlap sera donc de 80 %. Autrement dit, Le nombre de nouveaux points de données est 5 fois le nombre d'anciennes données

Après avoir segmenté les données, nous obtenons une nouvelle donnée composée de 10981

séquences, chaque séquence contient 50 valeurs [x,y,z]. L'étiquette (activité) pour chaque segment sera sélectionnée par l'étiquette de classe la plus fréquente présentée dans cette fenêtre

```
[[[ 0.00396193  0.90335584  0.03900325]
  [-0.4048535  -0.76694715  1.7151464 ]
  [ 2.4247332   0.11054594 -0.48807663]
  ...
  [-0.9720146  -1.2076004  -0.02967153]
  [-0.40198117 -0.36622262 -0.7721874 ]
  [ 2.2020829   1.2828777  -0.7721874 ]]

 [[ 0.13244677  0.49016026  0.6244853 ]
  [ 0.5967443   0.3908765   0.28283244]
  [ 0.97051835  1.1340445   1.4406561 ]
  ...
  [-0.6501876  -0.40198117 -0.47349828]
  [-0.16639541  0.2038108  -0.23791252]
  [-0.34518817 -0.26315385  1.3144294 ]]

 [[ 0.26677182 -0.26030803 -0.09678187]
  [ 0.9486175  -0.21358627 -0.8457901 ]
  [-0.5596193   2.7605457   0.9369371 ]
  ...
  [-0.22318842 -1.1402901  -1.5210136 ]
  [ 0.42046556 -0.2947055  -0.49032584]
  [-0.70908403  1.9686006   1.2260847 ]]

 ...
```

***Figue 8** : Nos données après etre segmente*

Pour que l'ordinateur fonctionne avec les étiquettes, nous devons également les convertir en forme 0 ou 1. Etiquette convertit en liste par la méthode de One-hot-encoded, chaque liste contient 6 valeurs 0 ou 1, la position du chiffre 1 correspond à chaque opération.

```
[[0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1.]
 ...
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0.]]
(109816, 6)
```

4.4 Diviser les données

Maintenant, nos données sont prêtes à interagir avec le modèle, mais avant de commencer à construire le modèle, nous devons diviser les données en 3 parties : Train, Validation et Test.

Cela nous aide à éviter Overfitting (le surajustement) - Le surajustement se produit lorsque le modèle s'entraîne trop longtemps sur des exemples de données ou lorsque le modèle est trop complexe, il peut commencer à apprendre le «bruit», ou des informations non pertinentes, dans l'ensemble de données. Lorsque le modèle mémorise le bruit et s'adapte trop étroitement à l'ensemble d'apprentissage, le modèle devient « sur-ajusté » et il est incapable de bien généraliser aux nouvelles données. Si un modèle ne peut pas bien généraliser à de nouvelles données, il ne pourra pas effectuer les tâches de classification ou de prédiction pour lesquelles il était destiné.

- **Ensemble de données d'entraînement (Training Data)**

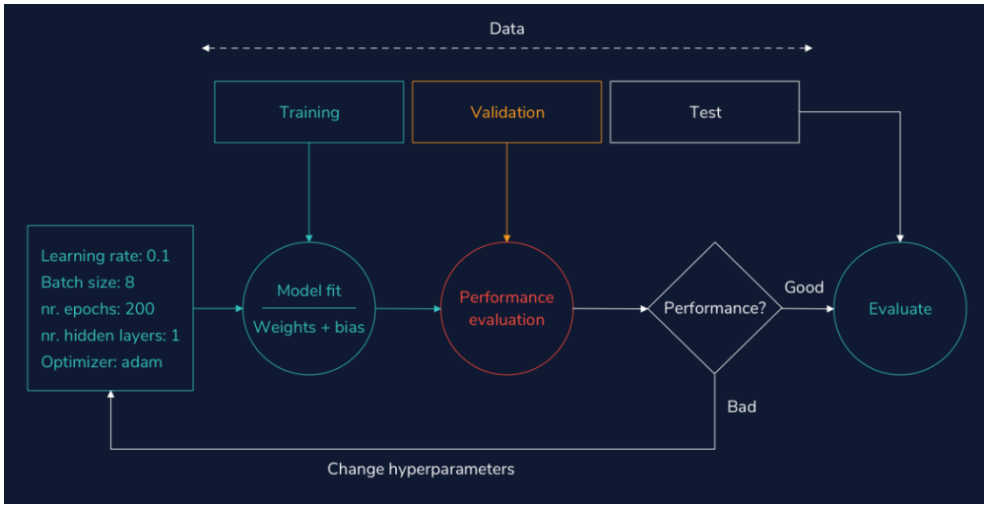
Ensemble de données utilisées pour l'apprentissage (par le modèle), c'est-à-dire pour ajuster les paramètres au modèle d'apprentissage automatique

- **Ensemble de données de validation**
  Ensemble de données utilisées pour fournir une évaluation impartiale d'un modèle adapté à l'ensemble de données d'apprentissage lors du réglage des hyperparamètres du modèle.

Il joue également un rôle dans d'autres formes de préparation de modèle, telles que la sélection de caractéristiques, la sélection de seuil de coupure.

- **Ensemble de données de test**

Ensemble de données utilisées pour fournir une évaluation impartiale d'un modèle final adapté l'ensemble de données d'apprentissage.



Figue 9 *: Rôle de la rame, de l'ensemble de test et du test de validation*

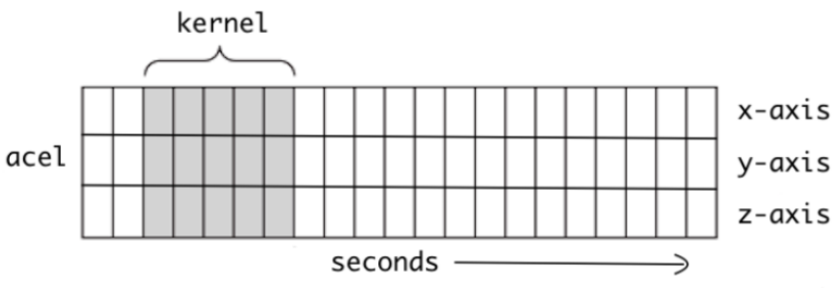
Dans notre projet, 20 % des données sont utilisées pour les données de test. 20 % du reste était un test de validation et 80 % était un donnes d'entrainement.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(reshaped_segments, labels, test_size = 0.2, random_state = 20)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(87852, 50, 3) (87852, 6)
(21964, 50, 3) (21964, 6)
```

4.5 Construction de Modele

Notre modèle est construit au-dessus de Conv1Ds (1 Dimension Convolution) car dans le noyau glisse le long d'une dimension (voir l'illustration ci-dessous), l'entrée de données de la première couche est de 50 x 3



Seule la taille des couches d'entrée et de sortie doit être spécifiée explicitement. Le réseau estime lui-même la taille des couches cachées. Ici, j'ai choisi kernel-size = 10 , numFilter1 = 100

```
model_1 = Sequential()
model_1.add(Reshape((window_size, num_data_parameters), input_shape=(numOfRows,numOfColumns)))
model_1.add(Conv1D(numFilters1, kernalSize, activation='relu', input_shape=(window_size, num_data_parameters)))
model_1.add(MaxPooling1D(3))
model_1.add(Conv1D(numNueronsFCL2, 10, activation='relu'))
model_1.add(GlobalAveragePooling1D())

model_1.add(Dropout(dropout))

model_1.add(Dense(num_classes, activation='softmax'))
print(model_1.summary())
```

Le réseau utilisé ici est de type séquentiel, ce qui signifie qu'il s'agit essentiellement d'un empilement de couches. Ces couches comprennent :

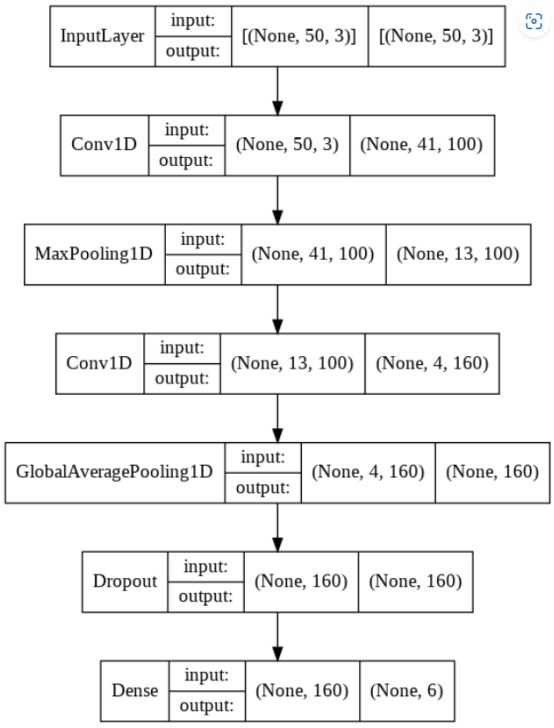


Figure 10 : *Diagramme illustrant notre modèle avec entrée et sortie de chaque couche*

Couche d'entrée : les données d'entrée se composent de 50 instances longues de tranches de temps d'un accéléromètre à 3 axes. Par conséquent, la taille de la couche d'entrée doit être remodelée à 50 x3. Les données traversent la couche d'entrée sous la forme d'un vecteur de longueur 150. La sortie de cette couche est 50x3.

Première couche CNN 1D : Cela définit un filtre de taille de noyau 10. 100 de ces filtres sont définis dans cette couche pour lui permettre d'apprendre 100 caractéristiques différentes. La sortie de

Cette couche est une matrice 41x100 de neurones où les poids de chaque filtre sont définis par colonne.

- **Couche de Max-pooling 1D** : elle est utilisée pour réduire la complexité de la sortie et pour éviter le surajustement des données. L'utilisation d'une taille de couche de regroupement de 3 réduit la taille de la matrice de sortie à 1/3 de la matrice d'entrée.
- **Deuxième couche CNN 1D** : cette couche permet au réseau de capter les fonctionnalités de niveau supérieur qui ont été manquées dans la première couche CNN. La sortie de cette couche est une matrice 4x160.
- **Couche de Average-Pooling** : cela fait la moyenne de la valeur de 4 poids dans le réseau, réduisant ainsi davantage le surajustement. La sortie est une matrice 1x160 de neurones.
- **Couche de décrochage (Drop out)** : Ici on choisit un facteur de 0,5 ce qui veut dire que 50% des neurones seront lâchés aleatoirement
- **Couche activée Softmax entièrement connectée** : cela réduit la sortie à la hauteur souhaitée de 6, ce qui indique le nombre de classes d'activité dans les données. Softmax force les six sorties du réseau neuronal à se résumer à une. La fonction d'activation Softmax calcule les probabilités relatives et choisi le neurone avec la plus grande probabilité de cette action comme predite activite.

4.6 Entrainement de la modele

Nous allons entraîner notre modèle sur l'ensemble X_train, y_train . Pour éviter le surajustement, nous devons utiliser la classe EarlyStopping. Il aide à la création d'un objet pour garder une trace de toutes les pertes subies pendant le processus de validation. L'entraînement est complètement arrêté lorsqu'il y a une diminution progressive observée dans les arrêts de perte de plusieurs époques qui se produisent continuellement à la suite.

```
callbacks_list = [
    keras.callbacks.ModelCheckpoint(
        filepath='best_model.{epoch:02d}-{val_loss:.2f}.h5',
        monitor='val_loss', save_best_only=True),
    keras.callbacks.EarlyStopping(monitor='loss', patience=1)
]

model_m.compile(loss='categorical_crossentropy',
                optimizer='adam', metrics=['accuracy'])

BATCH_SIZE = 1024
EPOCHS = epochs

history = model_m.fit(X_train,
                      y_train,
                      batch_size=BATCH_SIZE,
                      epochs=EPOCHS,
                      callbacks=callbacks_list,
                      validation_split=0.2,
                      verbose=1)
```

Un autre facteur très important auquel nous devons prêter attention est l'optimiseur. Les optimiseurs sont des algorithmes ou des méthodes utilisés pour minimiser une fonction d'erreur (loss) ou pour maximiser l'efficacité de la production. Les optimiseurs sont des fonctions mathématiques qui dépendent des paramètres apprenables du modèle, c'est-à-dire des Poids et des bias. C'est un facteur qui affecte grandement la précision de notre modèle.

Dans ce projet, nous utiliserons 4 types d'optimiseurs différents pour notre modèle :

- *Adam* (Adaptive Moment Estimation)
- *SGD (Stochastic Gradient Descent)*

- *AdaDelta* (Adaptive Learning Rate method Delta)
- RMS-Prop (Root Mean Square Propagation)

4.7 Resultat

Les trois principales mesures utilisées pour évaluer un modèle de classification sont l'exactitude, la précision et la perte.

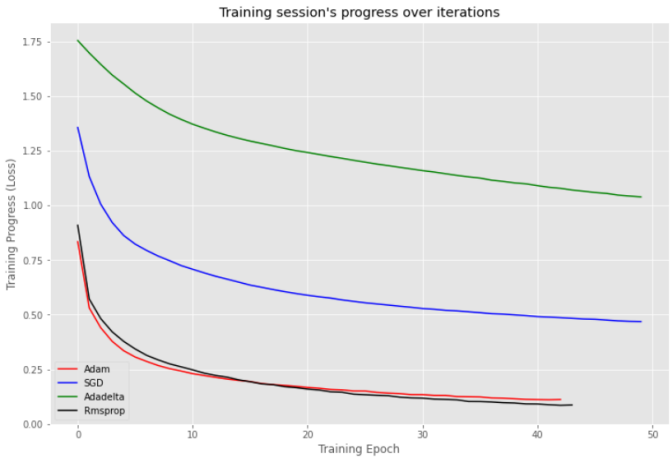
- **L' exactitude** (Recall) est définie comme le pourcentage de prédictions correctes pour les données de test. Il peut être calculé facilement en divisant le nombre de prédictions correctes par le nombre de prédictions totales.
- **La précision** est définie comme la fraction d'exemples pertinents (vrais positifs) parmi tous les exemples qui ont été prédits comme appartenant à une certaine classe.
- **Perte (Categorical – Entropy loss) :**

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Ou $\hat{y}_i$ est la i-ième valeur scalaire dans la sortie du modèle , y_i est la valeur cible correspondante et la taille de sortie est le nombre de valeurs scalaires dans la sortie du modèle. Cette perte est une très bonne mesure de la distinction entre deux distributions de probabilité discrètes. Dans ce contexte, y_i est la probabilité que l'événement a se produise
Voici les résultats de notre modèle sur 4 optimiseurs différents :

Perte / Précision	Adam	Adadelta	Rms-prop	SGD
Entrainnement	***96.37 % / 0.1122***	***65.42% / 1.0391***	***97.09 %/ 0.0872***	***83.89 % / 0.4690***
Validation	***96.53 % / 0.1071***	***67.8 % / 0.9932***	***96.50% / 0.1109***	***85.08% / 0.4311***
Test	***96.26%***	***67.61 %***	***96.49%***	***85.03 %***

Nous constatons que Rms-prop et Adam donnent une grande précision sur donnes de test **(> 96 %)** et sont tant mieux que les 2 reste. La progression d'apprendre de notre modele est illustre dans les 2 graphiques ci-dessous :



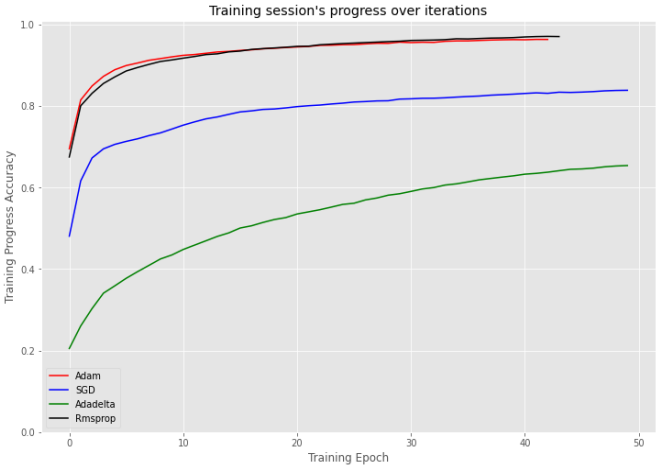


Figure 11 : *Comparaison de la précision et de la perte de chaque optimiseur sur entrainnement donnes*

Cependant, les résultats de RMS-prop sont encore légèrement meilleurs, nous utiliserons donc cet optimiseur pour d'autres recherches. Cependant, les résultats de RMS-prop sont encore légèrement meilleurs, nous utiliserons donc cet optimiseur pour d'autres recherches.

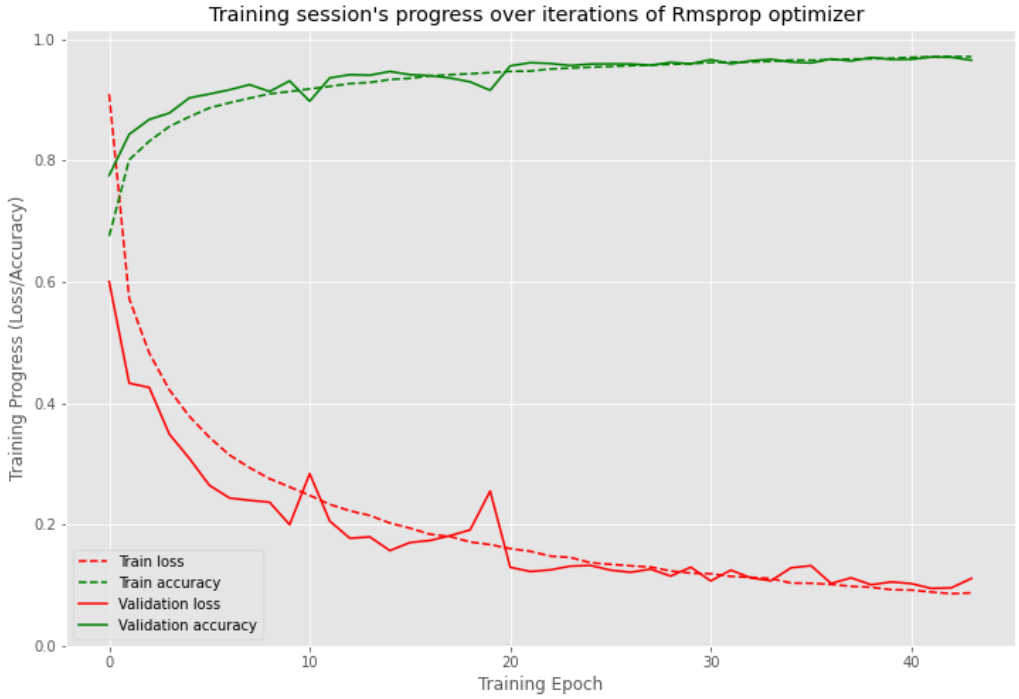


Figure 12 : *Progression d'apprentissage du modele*

Notre modèle a répondu aux exigences fixées : précision croissante et perte décroissante avec l'époque. La stabilité est atteinte après 46 époques. Aucun phénomène de surajustement ne se produit

Le tableau suivant compare la etiquette prédite et la vrai etiquette sur l'ensemble de test.

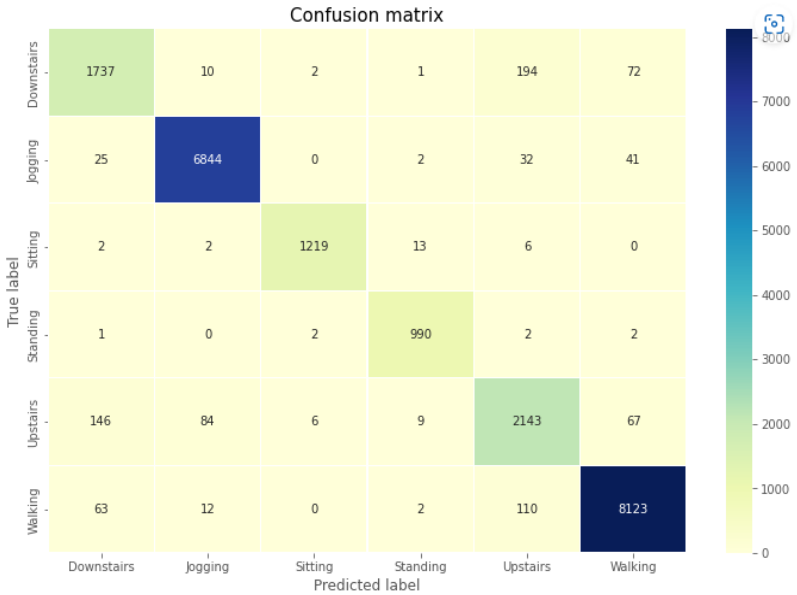


Figure 13 *: Matrice de confusion du modele sur l'ensemble de test*

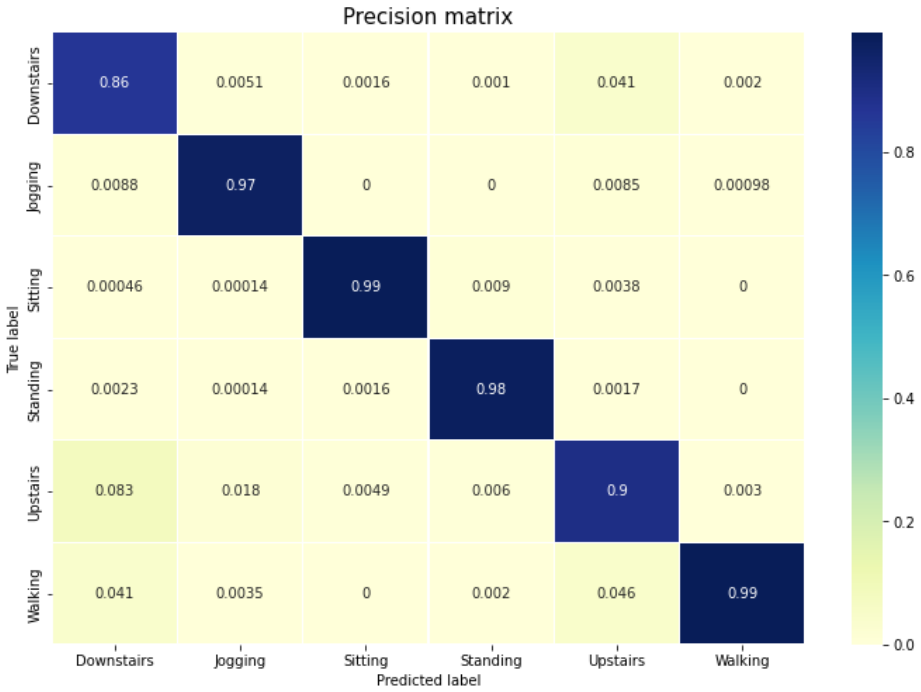
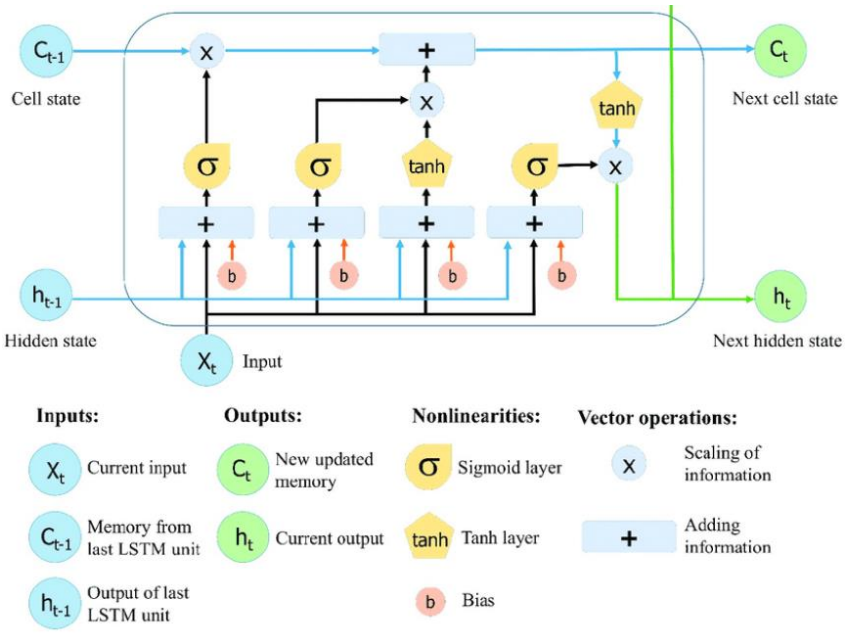


***Figure 14** : Précision du modèle pour chaque action*

Comme nous l'avons dit précédemment, les actions "marcher" et "Jogging" ont la plus grande quantité de données, donc les résultats montrent que ces deux actions ont la plus grande précision (99% et 97%). La modèle n'est toujours pas en mesure de distinguer complètement "Monter l'escalier" et "Descendre l'escalier" et confondent souvent ces 2 activités entre elles

4.8 Improvisation de notre modèle



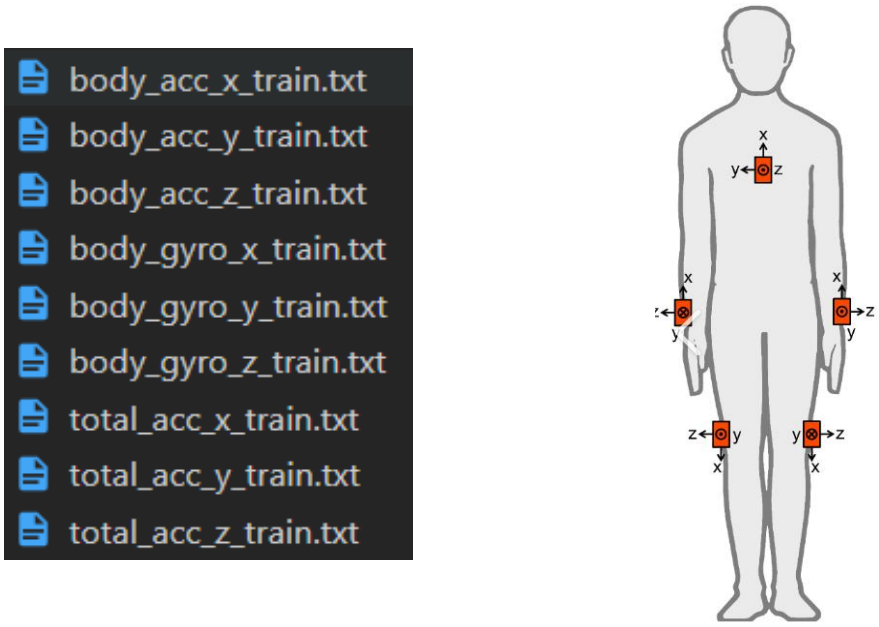
3 ***Figure 15** : Schéma du réseau LSTM .*

Notre modèle fonctionne très bien avec des données avec une petite quantité de caractéristiques (3 caractéristiques x,y,z).Cependant, avec une grande taille de donnes avec un grand nombre de caractéristiques, le temps d'apprentissage du modèle deviendra très long s'il ne fonctionne que dans le normal façon. Pour cette raison, on doit ajouter la couche de LSTM a notre modèle

LSTM (ou Long Short-Term Memory Network en anglais) sont un des types de réseau neuronal récurrent capable d'apprendre et de se souvenir sur de longues séquences de données d'entrée. Ils sont destinés à être utilisés avec des données composées de longues séquences de données, jusqu'à 200 à 400 pas de temps. Ils peuvent être un bon ajustement pour ce problème. Le modèle peut prendre en charge plusieurs séquences parallèles de données d'entrée, telles que chaque axe des données de l'accéléromètre et du gyroscope. Le modèle apprend à extraire des caractéristiques à partir de séquences d'observations et à mapper les caractéristiques internes à différents types d'activités.

En combinant les avantages des réseaux de neurones convolutifs (CNN) qui peuvent extraire des caractéristiques efficaces des données, et de la mémoire longue à court terme (LSTM) qui peut non seulement trouver l'interdépendance des données dans les données de séries chronologiques, mais aussi détecter automatiquement le meilleur mode adapté pour les données pertinentes, cette méthode peut améliorer efficacement la précision de la prédiction et réduire le temps d'apprentissage du modèle

Nous utiliserons des données plus volumineuses, qui ont 9 caractéristiques au lieu de 3



total acceleration, body acceleration et body gyroscope

Les données sont recueillies à partir d'accéléromètres et de gyroscopes montés sur le corps et sur les jambes de la personne.

Après avoir fusionné les données de 9 fichiers, nous obtenons un fichier de données

0	1	2	3	4	5	6	7	8
0.000180:	0.010766...	0.055560...	0.030191...	0.066013...	0.022858...	1.012817...	-0.12321...	0.102934...
0.010138...	0.006579...	0.055124...	0.043710...	0.042698...	0.010315...	1.022832...	-0.12687...	0.105687...
0.009275...	0.008928...	0.048404...	0.035687...	0.074850...	0.013249...	1.022027...	-0.12400...	0.102102...
0.005065...	0.007488...	0.049774...	0.040402...	0.057319...	0.017751...	1.017876...	-0.12492...	0.106552...
0.010810...	0.006140...	0.043013...	0.047096...	0.052342...	0.002553...	1.023679...	-0.12576...	0.102813...

X_train se compose de 7352 séquences, chaque séquence se compose de 128 lignes avec 9 caractéristiques et taux de overlap est 50%.

Le modèle CNN LSTM lira les sous-séquences de la séquence principale sous forme de blocs, extraira les caractéristiques de chaque bloc, puis permettra au LSTM d'interpréter les caractéristiques extraites de chaque bloc.

Une approche de mise en œuvre de ce modèle consiste à diviser chaque fenêtre de 128 pas de temps en sous-séquences que le modèle CNN doit traiter. Par exemple, les 128 pas de temps de chaque fenêtre peuvent être divisés en quatre sous-séquences de 32 pas de temps.

```
    # reshape data into time steps of sub-sequences
n_steps, n_length = 4,32
trainX = X_train.reshape((X_train.shape[0], n_steps, n_length, n_features))
testX = X_test.reshape((X_test.shape[0], n_steps, n_length, n_features))
```

Figure 16 : *Remodeler les données pour être compatible avec le réseau LSTM-CNN*

```
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3, activation='relu'), input_shape=(None,n_length,n_features)))
model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3, activation='relu')))
model.add(TimeDistributed(Dropout(0.5)))
model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(100))
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 17 : *Modèle CNN-LSTM*

L'ensemble du modèle CNN peut être enveloppé dans une couche TimeDistributed pour permettre au même modèle CNN de lire dans chacune des quatre sous-séquences de la fenêtre.

Résultats :

```
...
Epoch 24/25
115/115 [==============================] - 6s 49ms/step - loss: 0.0750 - accuracy: 0.9626
Epoch 25/25
115/115 [==============================] - 6s 52ms/step - loss: 0.0731 - accuracy: 0.9656
```

```
Epoch 28/50
69/69 [==============================] - 44s 642ms/step - loss: 0.1367 - accuracy: 0.9548 - val_loss: 0.1692 - val_accuracy: 0.9438
Epoch 29/50
69/69 [==============================] - 44s 641ms/step - loss: 0.1340 - accuracy: 0.9551 - val_loss: 0.1134 - val_accuracy: 0.9632
```

Figure 18 : *Comparaison les résultats entre le réseau CNN-LSTM (haut) et le réseau CNN (bas)*

Nous obtenons une précision très élevée (**96,56%**) et une perte assez faible. Dans le même temps, on remarque également que le temps d'apprentissage est de 52 ms/pas, bien plus rapide que les 641 ms/pas du réseau CNN classique.

5. Conclusion

Après environ 3 mois de stage, même s'il était en ligne, j'ai eu l'opportunité d'interagir et de travailler avec des assistants d'enseignement dans l'une des principales écoles européennes d'informatique. Cela m'a donné beaucoup de leçons, à la fois en termes de connaissances professionnelles et d'expérience de travail

Ce fut une belle opportunité et aussi un défi, car durant mon stage, j'ai également rencontré de nombreuses difficultés : par exemple, le manque de connaissance des mathématiques avancées utilisées en ML, j'ai donc dû beaucoup apprendre à partir des supports fournis et en ligne. Un autre problème est l'optimisation de l'algorithme pour que le modèle tourne à la vitesse optimale, ce n'est pas un problème facile car il y a beaucoup de paramètres à ajuster.

Après pres de 3 mois de stage, j'ai effectué les tâches suivantes :

- Équiper les connaissances de base du Machine Learning, des concepts algorithmiques : non régression linéaire, rétropropagation, ...
- Je suis doté d'une certaine connaissance de l'ingénierie des données, capable de gérer des données brutes
- Je suis familier avec python et les bibliothèques de Tensorflow
- Construire un modèle simple et optimiser l'algorithme.

6. Tables des illustrations

Reference:

- Tensorflow :

TensorFlow — Wikipédia (wikipedia.org)

- CNN :

What are Convolutional Neural Networks? | IBM

- Activation layer:

Activation Functions in Neural Networks [12 Types & Use Cases] (v7labs.com)

- Coche Convolutionelle :

How Do Convolutional Layers Work in Deep Learning Neural Networks? (machinelearningmastery.com)

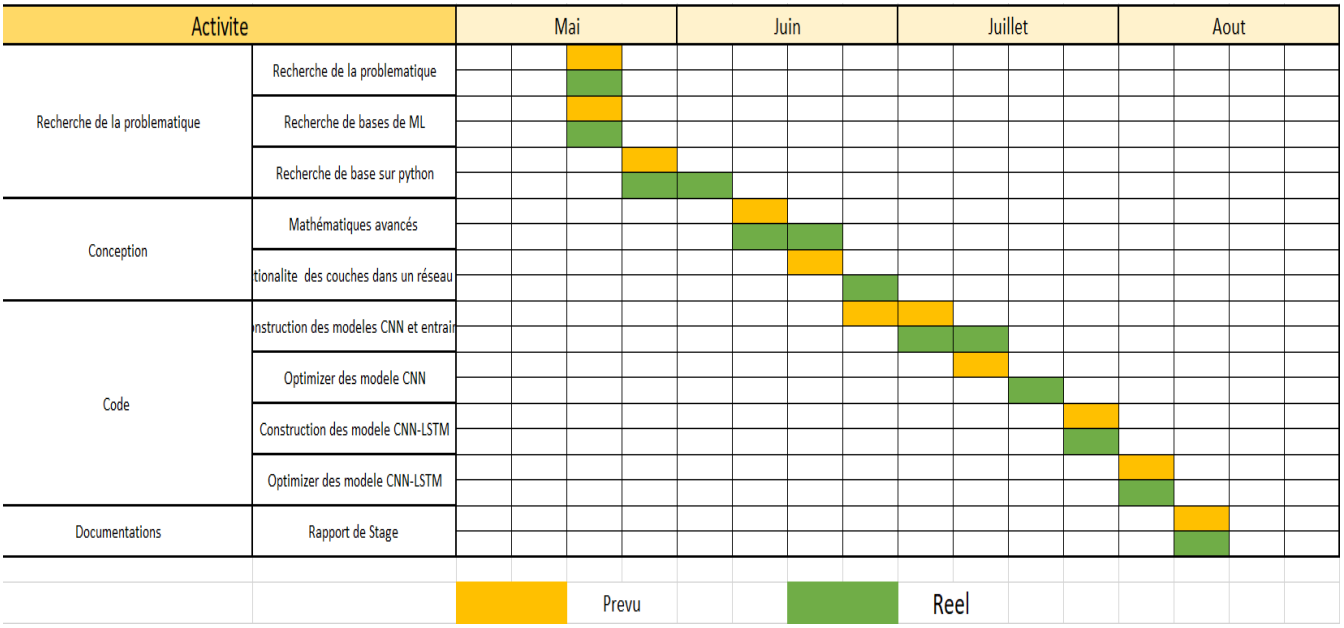
- Avantage du modèle CNN-LSTM

A CNN-LSTM-Based Model to Forecast Stock Prices (hindawi.com)

- Coding assistance :

Implementing a CNN for Human Activity Recognition in TensorFlow - Aaqib Saeed

Annexes 1:


Activite
Mai
Juin
Juillet
Aout
Recherche de la problematique
Recherche de la problematique
Recherche de bases de ML
Recherche de base sur python
Conception
Mathématiques avancés
tionalite des couches dans un réseau
Code
nstruction des modeles CNN et entrair
Optimizer des modele CNN
Construction des modele CNN-LSTM
Optimizer des modele CNN-LSTM
Documentations
Rapport de Stage
Prevu
Reel