

UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

**SIMULATION D’UNE CONDUITE AUTONOME SUR UN CIRCUIT FORMULA 1**

Par Lahcene Zinnour

DÉPARTEMENT DE MATHÉMATIQUES

ET

D'INFORMATIQUE

PIF 6004

Hiver 2019

# Table des matières

[Table des matières 3](#_Toc13280262)

[Liste des tableaux 4](#_Toc13280263)

[Liste des figures 5](#_Toc13280264)

[Méthodologie 6](#_Toc13280265)

[1. Introduction 7](#_Toc13280266)

[2. Réseaux de neurones convolutionnels 7](#_Toc13280267)

[1. Convolutions 7](#_Toc13280268)

[2. Couche entièrement connectée FL 10](#_Toc13280269)

[3. Couche de sortie 10](#_Toc13280270)

[4. Couche de perte (LOSS) 11](#_Toc13280271)

[3. K plus proches voisins (k-NN) 11](#_Toc13280272)

[Résultats 12](#_Toc13280273)

[1. Introduction 13](#_Toc13280274)

[2. Prétraitement des images 13](#_Toc13280275)

[3. ARCHITECTURE DE CNN 15](#_Toc13280276)

[Bibliographie i](#_Toc13280277)

[iii](#_Toc13280278)

[iv](#_Toc13280279)

# Liste des tableaux

**No table of figures entries found.**

# Liste des figures

# Méthodologie

## Introduction

L'apprentissage supervisé consiste en des variables d'entrée (X) et une variable de sortie (Y). Nous utilisons un algorithme pour apprendre la fonction de mappage de l'entrée à la sortie.

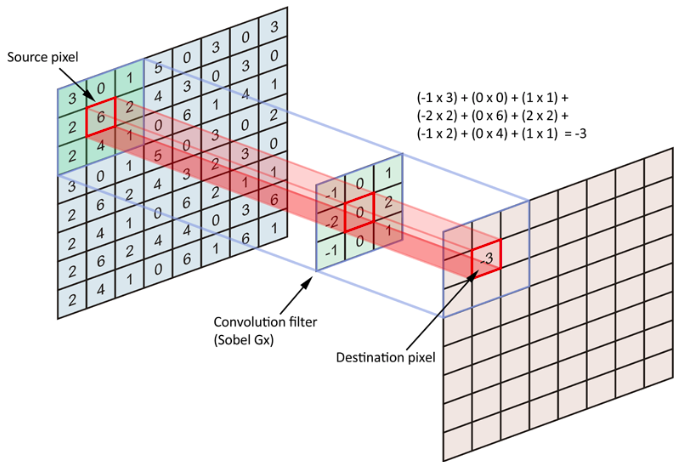
Le but est d’approcher si bien la fonction de mappage que, lorsque nous avons de nouvelles données d’entrée (X), nous pouvons prédire les variables de sortie (Y) pour ces données.

Il est appelé apprentissage supervisé parce que le processus d'un algorithme d'apprentissage à partir de l'ensemble de données de formation peut être considéré comme un enseignant responsable du processus d'apprentissage. Nous connaissons les réponses correctes, l'algorithme effectue des prédictions itératives sur les données d'apprentissage et est corrigé par l'enseignant. L'apprentissage s'arrête lorsque l'algorithme atteint un niveau de performance acceptable.

Dans notre projet, nous avons utilisé deux types d'algorithmes d'apprentissage machine supervisé sont; les réseaux de neurones convolutionnels (CNN) et le k plus proches voisins (k-NN).

## Réseaux de neurones convolutionnels

### Convolutions

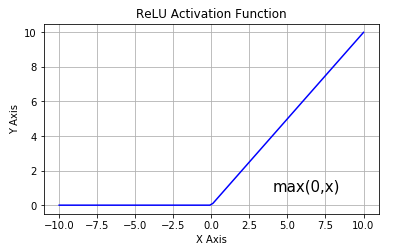
CNN utilise des filtres (kernels) pour détecter les éléments présents dans l’image, tels que les bords. Un filtre est une matrice de valeurs (poids), formés pour détecter des caractéristiques spécifiques. Le filtre se déplace sur chaque partie de l'image pour vérifier si la fonctionnalité qu'il est censé détecter est présente. Pour fournir une valeur représentant le degré de confiance qu'une caractéristique spécifique est présente, le filtre effectue une opération de convolution, qui est un produit par élément et une somme entre deux matrices.

Lorsque la caractéristique est présente dans une partie d'une image, l'opération de convolution entre le filtre et cette partie de l'image donne un nombre réel de valeur élevée. Si la fonctionnalité n'est pas présente, la valeur résultante est faible.

Le résultat de ce filtre sur l’image est une matrice de sortie qui stocke les convolutions de ce filtre sur diverses parties de l’image. Le filtre doit avoir le même nombre de canaux que l'image d'entrée pour que la multiplication par élément puisse avoir lieu. Par exemple, si l'image d'entrée contient trois canaux (RGB), le filtre doit également contenir trois canaux.

De plus, un filtre peut être glissé sur l'image d'entrée à des intervalles variables, en utilisant une valeur de déplacement. La valeur du déplacement détermine le degré de déplacement du filtre à chaque étape.

Pour que le réseau de neurones de convolution puisse apprendre les valeurs d'un filtre détectant les caractéristiques présentes dans les données d'entrée, le filtre doit être transmis via un mappage non linéaire. La sortie de l'opération de convolution entre le filtre et l'image d'entrée est additionnée d'un terme de polarisation (bias) et transmise via une fonction d'activation non linéaire. La fonction d'activation a pour but d'introduire la non-linéarité dans notre réseau. Puisque nos données d'entrée sont non linéaires (il est impossible de modéliser les pixels qui forment une signature manuscrite de manière linéaire), notre modèle doit en tenir compte. Pour ce faire, nous utilisons la fonction d'activation de l'unité linéaire rectifiée (ReLU) (dans notre CNN googleNet):



La fonction ReLU est assez simple, les valeurs inférieures ou égales à zéro deviennent zéro et toutes les valeurs positives restent les mêmes.

Après une ou deux couches convolutives, il est courant de réduire la taille de la représentation produite par la couche convolutive. Cette réduction de la taille de la représentation est appelée « Downsampling ».

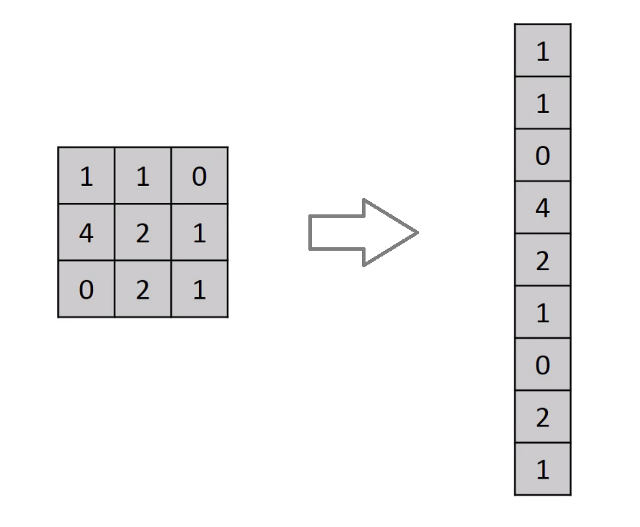
Pour accélérer le processus d’entrainement et réduire la quantité de mémoire utilisée par le réseau, nous essayons de réduire la redondance présente dans la fonction d’entrée. Il existe plusieurs manières de sous-échantillonner une image, parmi eux c’est le max-pooling (POOL).

Dans le max-pooling, une matrice passe sur une image en fonction de stride (nombre d'unités à déplacer à chaque passage). À chaque étape, la valeur maximale dans la matrice est regroupée dans une matrice de sortie.

Après plusieurs couches convolutives et opérations de Downsampling, la représentation d'image 3D (RGB) est convertie en un vecteur de caractéristiques qui est passé dans un Perceptron multicouches, qui est un réseau neuronal avec au moins trois couches. Ceci est appelé une couche entièrement connectée (Fully Connected Layer FL).

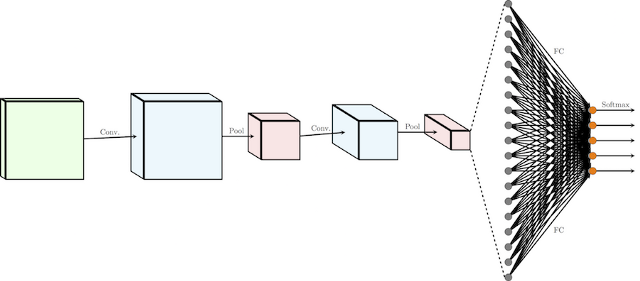
### Couche entièrement connectée FL

Dans une FL opération d'un réseau de neurones, la représentation d'entrée est aplatie dans un vecteur de caractéristiques et transmise à travers un réseau de neurones pour prédire les probabilités de sortie. L'image suivante décrit l'opération d'aplatissement:



Les lignes sont concaténées pour former un vecteur long. Si plusieurs couches d'entrée sont présentes, ses lignes sont également concaténées pour former un vecteur de caractéristiques plus long.

Le vecteur est ensuite passé à travers plusieurs couches denses. À chaque couche dense, le vecteur est multiplié par les poids de la couche, additionné de ses biais et passé à travers une non-linéarité.

L'image suivante montre la couche FL et les couches denses:

### Couche de sortie

La couche de sortie d'un CNN est chargée de produire la probabilité de chaque classe en fonction de l'image d'entrée. Pour obtenir ces probabilités, la couche Dense finale contienne le même nombre de neurones qu'il y a de classes. La sortie de cette couche dense passe ensuite par la fonction d'activation Softmax, qui mappe toutes les sorties de la couche dense finale sur un vecteur dont les éléments totalisent un.

### Couche de perte (LOSS)

Pour mesurer la précision de notre réseau dans la prévision de la sortie, nous utilisons une fonction de perte LOOS. La fonction de perte attribue un nombre à valeur réelle pour définir la précision du modèle lors de la prévision de la sortie. Une fonction de perte commune à utiliser lors de la prévision de plusieurs classes de sortie est la fonction de perte catégorique d'entropie croisée (Categorical Cross-Entropy Loss function).

## K plus proches voisins (k-NN)

Le principe de la classification du plus proche voisin consiste à trouver un nombre prédéfini, c’est-à-dire le "k" - d’échantillons d’entraînement le plus proche en distance d’un nouvel échantillon, qui doit être classé. L'étiquette du nouvel échantillon sera défini à partir de ces voisins.

Le k plus proches voisins (k-NN) est une méthode non paramétrique utilisée pour la classification et la régression. Dans les deux cas, il s'agit de classer l'entrée dans la catégorie à laquelle appartient les k plus proches voisins dans l'espace des caractéristiques identifiées par apprentissage. Le résultat dépend si l'algorithme est utilisé à des fins de classification ou de régression :

* en classification k-NN, le résultat est une classe d'appartenance. Un objet d'entrée est classifié selon le résultat majoritaire des statistiques de classes d'appartenance de ses k plus proches voisins, (k est un nombre entier positif généralement petit). Si k = 1, alors l'objet est assigné à la classe d'appartenance de son proche voisin.
* en régression k-NN, le résultat est la valeur pour cet objet. Cette valeur est la moyenne des valeurs des k plus proches voisins.

# Résultats

## Introduction

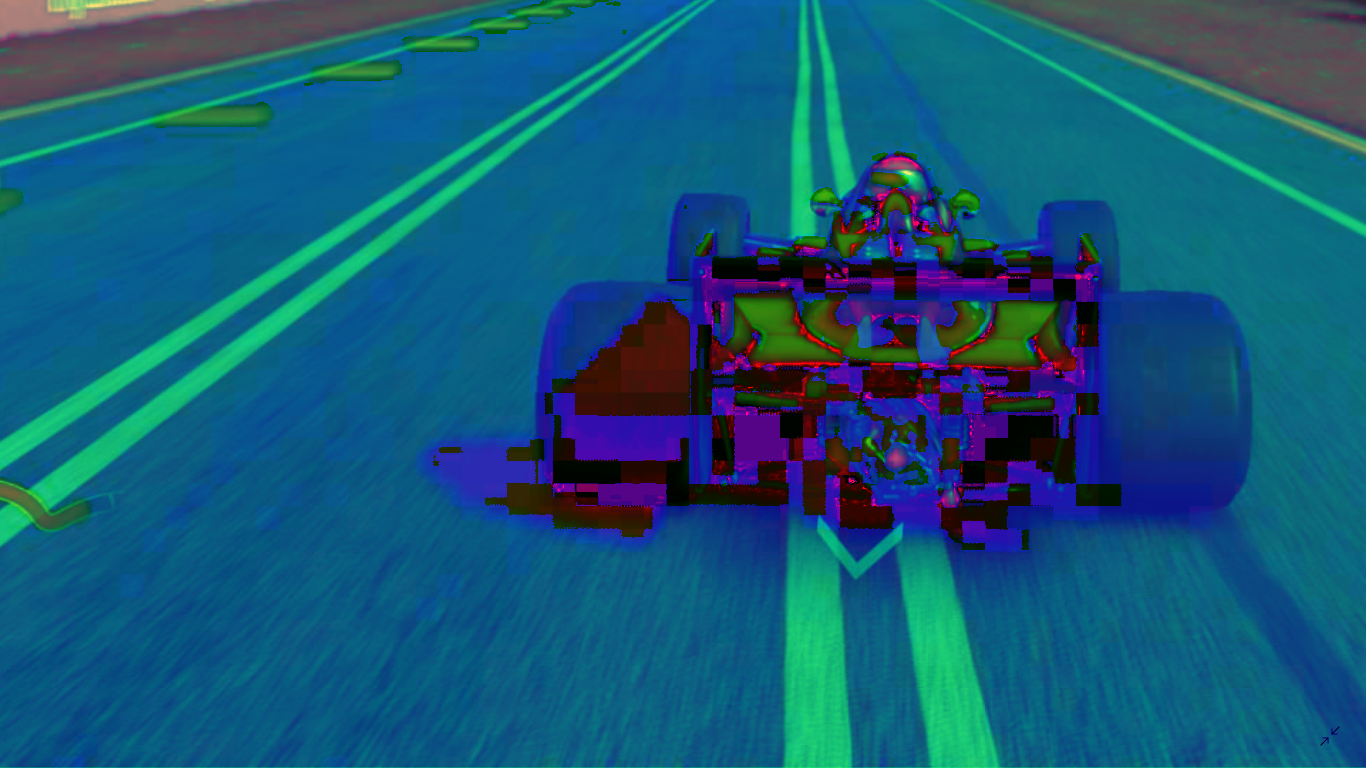
Pour ce chapitre, nous expliquerons la solution au problème, comme le prétraitement des images, l’architecture CNN utilisée pour prédire le trajet des voitures, prédire la vitesse de voiture en utilisant les données extraites des images à l’aide de K-NN, en essayant aussi de prédire la vitesse par rapport à la forme du chemin (courbe ou droite), et enfin nous discutons les résultats finaux.

## Prétraitement des images

Tout d'abord, nous décidons d'essayer d'extraire la voie sur la route pour permettre à CNN de prévoir facilement l'action par rapport à l'état de la voiture sur la route. Pour cela nous avons utilisé Python et OpenCV pour trouver des lignes dans les images de route.

Pour commencer, nous avons sélectionné la couleur blanche.

Nous convertir l'image RGB dans l'espace couleur différente, et nous augmentons la luminosité pour mieux détecter le couleur blanc :



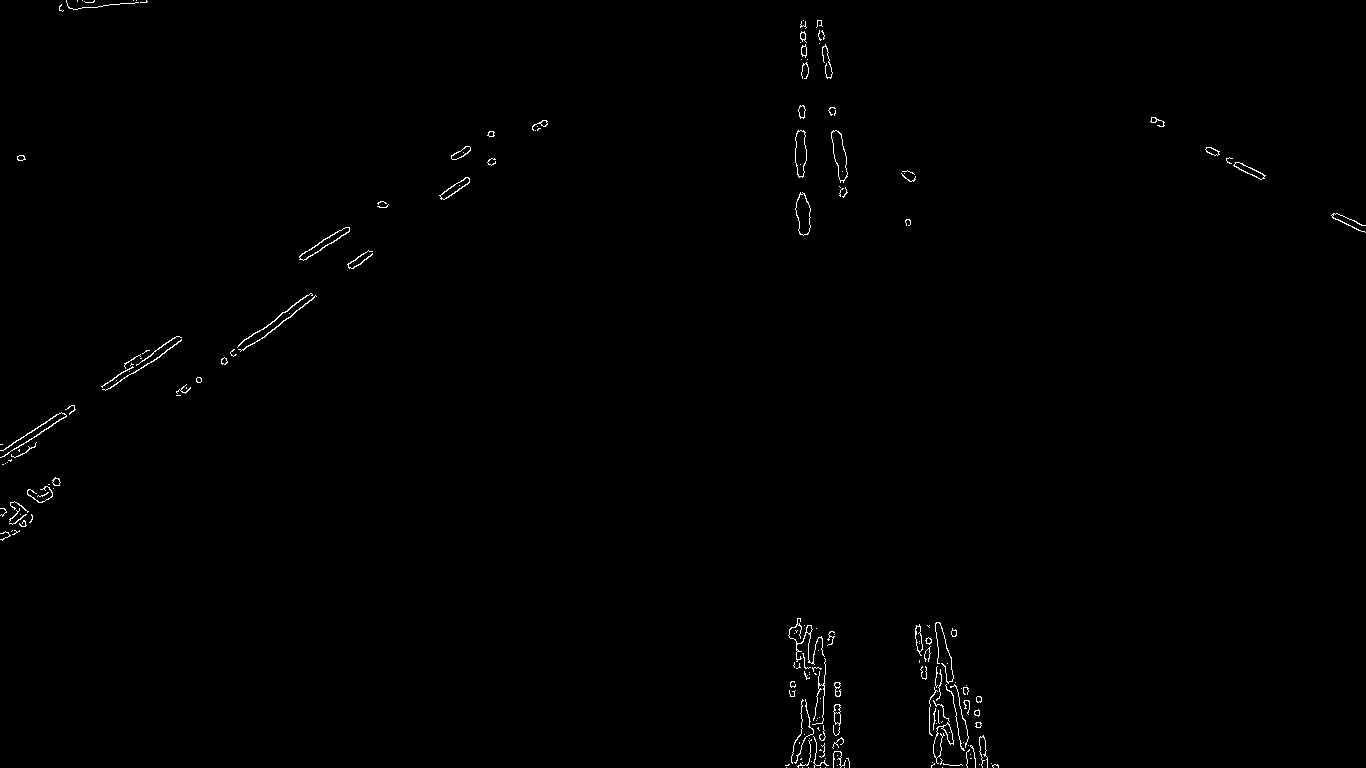
Après en détecte les lignes de route :



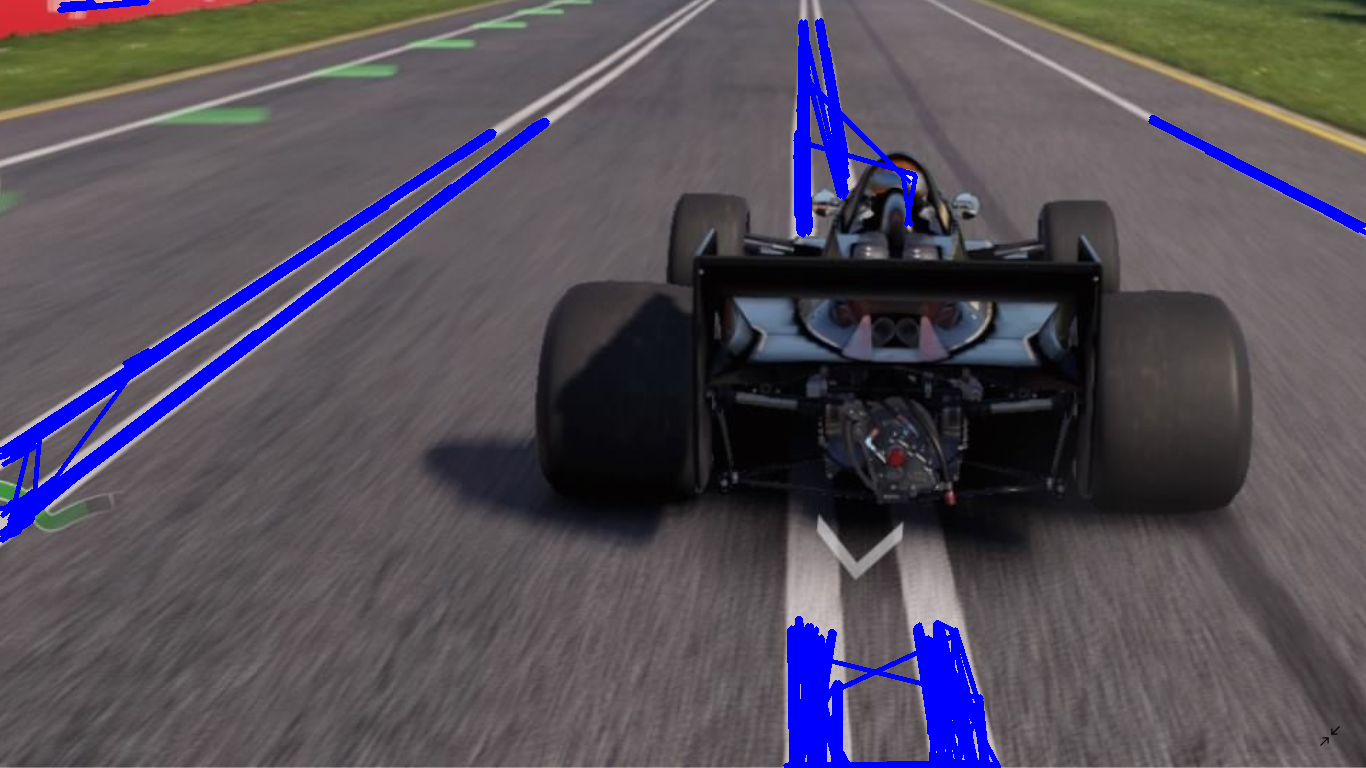
Puis nous détectons les bordures afin de trouver des lignes:

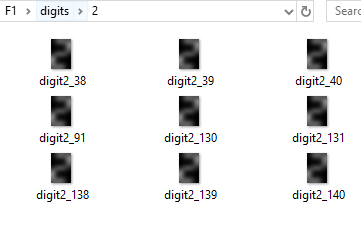
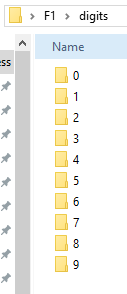
Apres en aplique l’algorithme Canny :



L’image suivant montre le résultat final :



Pour entrainer le k-NN à prédire les chiffres, nous collectons les images de la capture d'écran lorsque nous jouons au jeu :



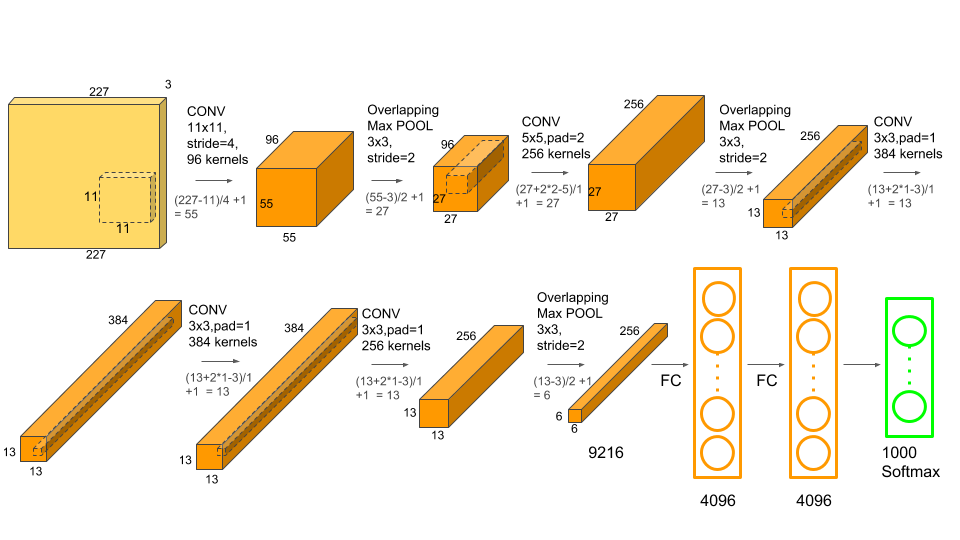
## ARCHITECTURE DE CNN

Pour implémenter l'architecture CNN, nous avons choisi le modèle AlexNet proposé par Alex Krizhevsky en 2012 lors de la compétition de défit ImageNet.

AlexNet se compose de 5 couches et convolutifs 3 couches entièrement connectées. Plusieurs filtres extraient des caractéristiques intéressantes dans une image. Dans une seule couche de convolution, il y a généralement plusieurs filtres de la même taille. Par exemple, la première couche de convergence d’AlexNet contient 96 filtres de taille 11x11x3. Notez que la largeur et la hauteur du noyau sont généralement les mêmes et que la profondeur correspond au nombre de canaux.

Les deux premières couches convolutives sont suivies des couches max-pooling. Les troisième, quatrième et cinquième couches convolutives sont connectées directement. La cinquième couche de convolution est suivie par une couche max-pooling, dont la sortie se trouve dans une série de deux couches FL. La deuxième couche FL alimente un classifieur softmax avec 5 classes dans notre projet (accélération, avant, droit, gauche, arrière). La fonction d’activation ReLU est appliquée après toutes les couches de convolution et FL.

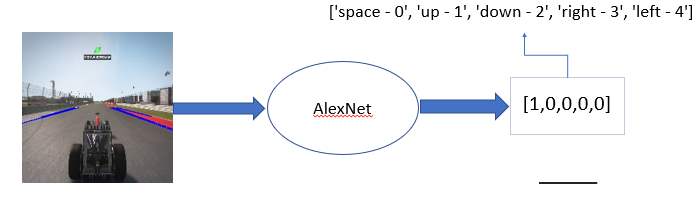
L’image montre l’architecture de AlexNet :



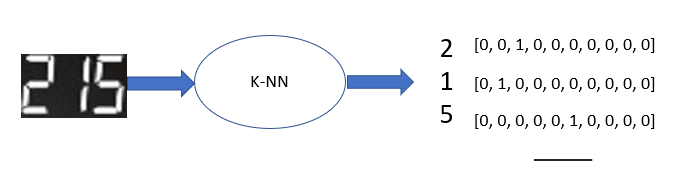
L’image montre le code python pour implémente l’architecture d’AlexNet à l’aide de bibliothèque TfLearn.



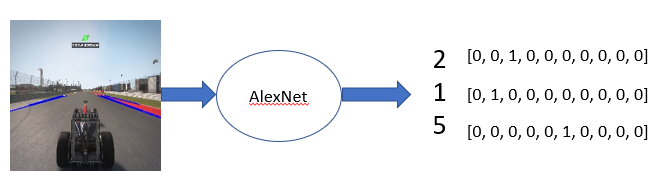
L’image suivant montre l’architecture utilisée pour entrainer AlexNet à prédire les touches de clavier.



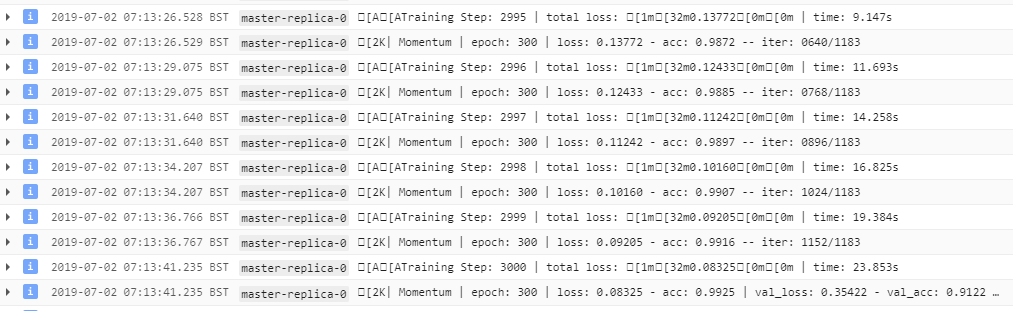
L’image suivant montre l’architecture utilisée pour entrainer le k-NN à prédire les chiffres.



L’image suivant montre l’architecture utilisée pour entrainer AlexNet à prédire la vitesse par rapport à les images.



L’image suivant montre le résultat de l’étape précèdent 1 avec une précision égale 0,91 et le « loss » égale à 0,08.



# Bibliographie

https://towardsdatascience.com/deep-learning-for-self-driving-cars-7f198ef4cfa2

https://voiture.kidioui.fr/lexique-automobile/voiture-autonome.html

https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception

https://www.learnopencv.com/understanding-alexnet/?fbclid=IwAR0dCMRHKtQv\_sV7pfwXJ1a7AC975A7NI4IK22fpKERyHrHughpP-RAILU8

https://towardsdatascience.com/tutorial-build-a-lane-detector-679fd8953132

https://becominghuman.ai/behavioral-cloning-in-deep-learning-using-keras-3786f5914b72

https://towardsdatascience.com/convolutional-neural-network-to-steer-a-vehicle-inside-a-game-2aab41a5ef60

https://github.com/Sentdex/pygta5

https://towardsdatascience.com/finding-lane-lines-on-the-road-30cf016a1165

https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1

# 

# 