

**UQTR**



Université du Québec  
à Trois-Rivières

## **PIF6004 ÉTÉ 2019**

**Enseignant: Francois Meunier**

### **Rapport final**

**réalisé par:**

**Simon Lafrenière  
& Patrick Duhaime**

---

# Table des matières

---

<b>Le projet</b>	<b>3</b>
<b>Les outils utilisés</b>	<b>3</b>
Tensorflow	3
Keras	4
ImageAI	5
<b>GTA5 simulator</b>	<b>6</b>
<b>Construction du jeu de données</b>	<b>7</b>
<b>Conversion des données</b>	<b>7</b>
<b>Entrainement du modèle</b>	<b>9</b>
<b>AlexNet</b>	<b>21</b>
<b>ResNet</b>	<b>22</b>
<b>Le matériel</b>	<b>22</b>
<b>Les résultats</b>	<b>23</b>
<b>La conclusion</b>	<b>24</b>

---

# Le projet

Le projet consiste à simuler l'automatisation la conduite automobile à l'aide d'un système de raisonnement computationnel (IA pour le reste du texte) basée sur un réseau de neurones convolutionnels. Pour se faire, la simulation prendra place dans un jeu de conduite automobile disponible gratuitement sur internet (question de budget..). L'IA devra apprendre à reconnaître les obstacle (i.e autres véhicules, immeubles, bords de route), contrôler sa vitesse et sa ligne (i.e rester sur le chemin).

Le processus final devrait aller comme suit, le système reçoit des images, il les traite à travers des filtres pour repérer les contours de la route et les obstacles puis prends une décision sur l'action à suivre.

Le projet peut être divisé en deux phases: une phase d'apprentissage de reconnaissance de formes, une phase d'apprentissage de prise de décision (apprentissage par renforcement). Et finalement la jonction des deux.

Les résultats attendus sont d'avoir une voiture autonome dans un environnement contrôlé, l'idée d'utiliser une simulation permet de limiter le nombre de variable à traiter (ce qui est une bonne chose car le matériel à notre disposition pour traiter les données est limité en terme de puissance de calculs..) ainsi que permettre un apprentissage semi-dirigé à non dirigé (les risques matériels et autres étant virtuellement absents).

## Les outils utilisés

Tous les outils trouvés fonctionnent avec le langage Python.

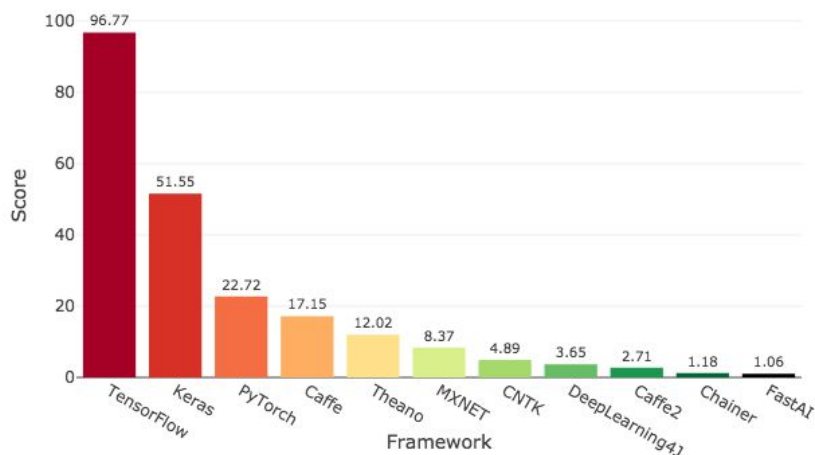
### Tensorflow

Notre recherche des outils pour réaliser le projet nous a vite fait découvrir Tensorflow. Cela n'est pas une surprise, car le monde de l'intelligence artificielle est dominé par ce logiciel ouvert selon un classement par utilisation, intérêt et popularité.<sup>1</sup>

---

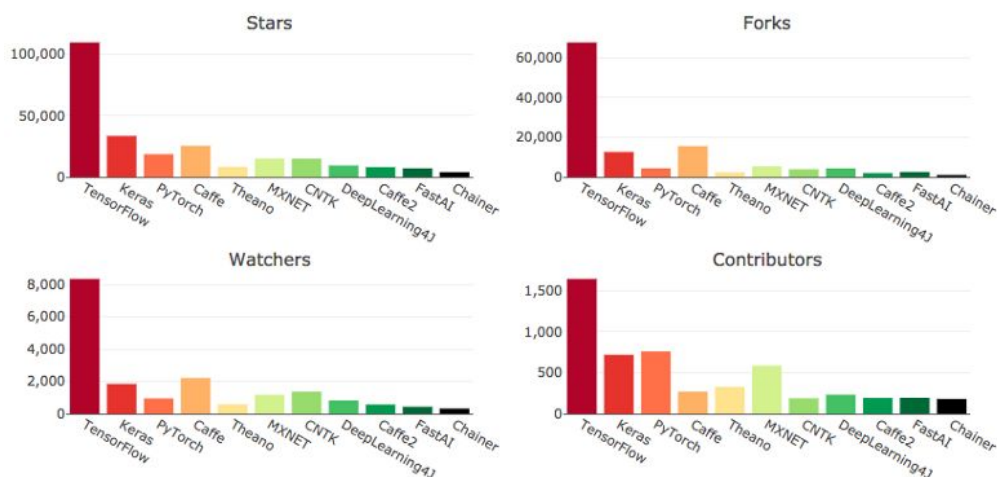
<sup>1</sup> <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

## Deep Learning Framework Power Scores 2018



TensorFlow est le champion incontesté. Il a la plupart des activités GitHub, des recherches Google, des articles Medium, des livres sur Amazon et des articles ArXiv. Il est également utilisé par la plupart des développeurs et figure dans la plupart des descriptions de tâches en ligne. TensorFlow est soutenu par Google.

### GitHub Activity



## Keras

Tensorflow depuis la version 1.12.0 utilise Keras<sup>2</sup> pour la création et l'entraînement des modèles. Keras supporte les modèles suivants:

<sup>2</sup> <https://keras.io/applications/>

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

## ImageAI

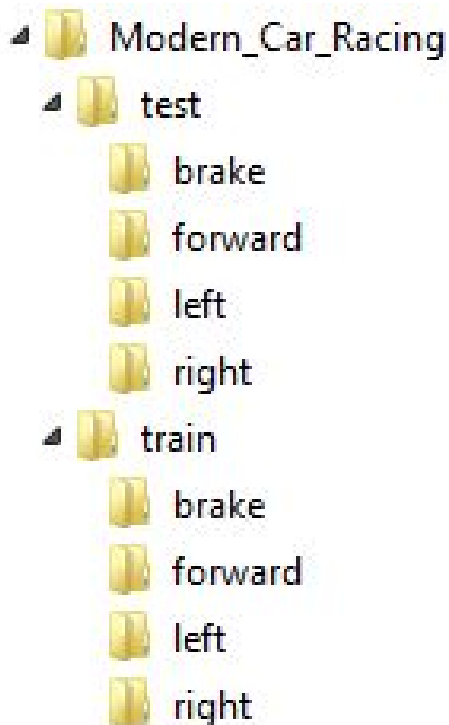
Construit dans un esprit de simplicité, ImageAI<sup>3</sup> prend en charge une liste d'algorithmes d'apprentissage automatique de pointe pour la prédiction d'image, la prédiction d'image personnalisée, la détection d'objet, la détection vidéo, le suivi d'objet vidéo et les formations de prédiction d'image. ImageAI prend actuellement en charge la prédiction d'images et l'entraînement à l'aide de 4 algorithmes différents d'apprentissage automatique formés sur le jeu de données ImageNet-1000. ImageAI prend également en charge la détection d'objet. Nous avons utilisé cet outil de détection avec le modèle YOLOv3 et le jeu de données COCO pour concevoir le capteur de proximité ([proximitySensor.py](#)) qui détecte la présence des autres véhicules sur la route.

ImageAI utilise la suite Tensorflow en simplifiant le code nécessaire pour produire des modèles ou pour faire le traitement sur les flux.

Une documentation complète en français existe: <https://imageai-fr.readthedocs.io/en/latest/>

<sup>3</sup> <https://github.com/OlafenwaMoses/ImageAI>

ImageAI en plus de fournir des outils pour la détection, permet de construire son propre modèle avec ses propres données pour la prédiction des objets. ImageAI demande de configurer les données dans une structure de dossier bien définie. Les noms des dossier représentent les actions qui contiennent les images qui lui sont associé. Cette structure doit se situer dans un dossier nommé “train” ainsi que dans un dossier nommé “test”, ces deux dossier à leur tour sont des sous dossiers du dossier racine de votre modèle.



## GTA5 simulator

Initialement nous nous sommes inspirés d’un [tutoriel](#) sur le jeu GTA5 remplissant sensiblement les objectifs que nous avons. Le but principal du tutoriel est de montrer comment décoder les images du jeu et simuler les touches de directions et ensuite appliquer certaines méthodes de “deep learning” pour l’apprentissage de la conduite autonome. Le tutoriel utilise principalement OpenCv et AlexNet pour résoudre la problématique. Nous avons gardé (partiellement) la partie sur OpenCv en l’adaptant à notre situation, comme nous utilisons deux types d’environnements logiciels différents (Mac et Windows) nous avons dû modifier le code de [capture d’écran](#) et la gestion des touches clavier. Pour la partie sur l’apprentissage nous avons finalement abandonné [AlexNet](#) en faveur de ResNet pour des raisons que nous verrons un peu plus loin.

# AlexNet

AlexNet est un réseau de neurones convolutionnels qui a pour but de classer les images en donnant une probabilité que l'image reçue (input) appartienne à une classe donnée. Donc on lui passe une image en entrée et il retourne un vecteur avec les probabilités d'appartenance aux différentes classes définies préalablement (chat, chien, etc.). Chaque élément du vecteur représente la probabilité que l'image appartienne à la classe *ix*, la somme de tous les éléments du vecteur doit donc donner 1 (ou 100%). Les images traitées doivent être de format 256x256. La structure générale du réseau repose sur 5 couches de convolution et 3 couches pleinement connectées. Bien qu'il s'agisse d'une excellente option donnant un taux d'erreurs relativement bas après entraînement nous avons choisie une autre route dû principalement au fait que ce réseau de neurones nécessite un nombre important d'images sources pour son entraînement, nous disposions d'un temps restreint pour monter notre banque de données. Le procédé d'AlexNet pour augmenter le nombre de données utilise l'inversion d'image (mirroring) ce qui ne pouvait être utilisé ici car notre système de classification repose sur la direction (on aurait obtenu des images de direction à droite dans notre fichier de direction à gauche, et vice versa...). Nous avons donc opté pour ResNet qui demande un nombre de données plus petit pour son entraînement. Notre test avec 5000 images fait sur AlexNet nous à donné un véhicule qui ne faisait qu'aller en ligne droite sans prendre en considération les autres actions. C'est suite à ce test que nous avons prit la décision d'utiliser la suite ImageAI et ResNet puisque cette librairie nous avait donné d'excellent résultats sur d'autres projets.

# ResNet

ResNet est similaire à AlexNet en ce qu'il ont pour objectif final d'analyser et classer des images. Mais ResNet (Residual Network) tente de résoudre deux problèmes sous-jacents au "deep learning", soit la difficulté d'entraînement et la perte d'efficacité des couches de neurones successives (la performance se dégrade au fur et à mesure qu'on va plus profondément dans le réseau). Dans un réseau de neurones conventionnel, chaque couche essaie d'extraire une caractéristique du modèle indépendamment des résultats des autres couches, l'apprentissage résiduel consiste à extraire les caractéristiques en se basant sur les résultats préliminaires des couches précédentes ce qui réduit fortement la dégradation des résultats. Pour le projet, notre intention était d'utiliser ResNet50 (50 couches) qui procure un bon ratio efficacité versus temps d'exécution, malheureusement nous avons utilisé par erreur ResNet152 (152 couches) qui donne des résultats étonnamment précis avec peu d'images sources pour effectuer son entraînement mais demande une puissance de calcul fortement supérieur. Étant donné les limites matérielles dans lesquelles nous nous trouvions cela a donné des résultats mitigés. Nous obtenions près de 100% de bonnes décisions mais trop lentement, c'est à dire que dans un contexte "en direct" l'interprétation des images était correcte la plupart du temps mais la décision arrivait trop tard et la voiture perdait sa ligne, nous avons résolu en partie ce problème en réduisant la vitesse de progression de la voiture sur le terrain.

# Construction du jeu de données

Afin de construire notre propre jeu de données, nous avons écrit des scripts pour capturer les images qui sont sauvegardés au moment où le joueur appuie sur une touche de contrôle soit, avancé, tourner à gauche, tourner à droite ou freiner. La touche et l'image sont sauvegardé ce qui nous permet de construire un jeu d'image associé à nos 4 actions.

Nous avons écrit 2 scripts différents pour capturer les données. Nous travaillions sur Windows et MAC et des soucis de compatibilité entre certaines librairies nous ont mené à ce choix, l'important ici n'était pas d'avoir un script de capture universel mais de capturer des données identiques sur les 2 plateformes.

Script de capture windows [ici](#)

Script de capture MAC [ici](#)

## Conversion des données

Afin de convertir les données capturés au format demandé par ImageAI nous avons écrit le script [convertTrainingData.py](#). Nous avons inclus dans le script de conversion des instructions pour faire l'augmentation des images, plus particulièrement, pour les images de freinages. Ces images étaient beaucoup moins nombreuses dans notre jeu de données et pour ne pas balancer les données vers le bas, c'est à dire éliminer des images d'accélération ou de virages qui seraient tout à fait utilisables, nous avons ajouté du bruit aux images de freinage pour sauvegarder deux images par capture, l'image originale et une image bruitée.



213p.noised



214p



214p.noised



215p



217p



217p.noised



218p



218p.noised




Notre jeu de données comprends plus de 34 000 images réparti dans la structure de dossier présenté plus haut soit les 2 dossiers train et test comprenant les sous dossier des actions.

La répartition des images est la suivante:

Le dossier test comprends 6909 images.

Le dossier train comprends 27128 images.

Les images sont réparties à peu près également entre les différentes actions.

	Modern_Car_Racing
Type :	Dossier de fichiers
Emplacement :	C:\UQTR
Taille :	51,9 Mo (54 457 444 octets)
Taille sur le disque :	132 Mo (139 386 880 octets)
Contenu :	34 037 Fichiers, 10 Dossiers
Créé le :	15 juin 2019, 18:38:50

## Entraînement du modèle

Après avoir construit le jeu de données, nous avons entraîné le modèle. ImageAI simplifie fortement le code nécessaire pour entraîner un modèle. Nous avons choisi le modèle ResNet puisque nous avons déjà travaillé avec ce modèle pour un autre projet qui faisait de la détection des objets un peu à la manière de notre proximitySensor pour le cours IAR1001.

```
from imageai.Prediction.Custom import ModelTraining
model_trainer = ModelTraining()
model_trainer.setModelTypeAsResNet()
model_trainer.setDataDirectory("MODERN_CAR_RACING")
model_trainer.trainModel(num_objects=4, num_experiments=30,
                           enhance_data=False, batch_size=16,
                           show_network_summary=True)
```

À notre insu, nous avons utilisé le type de modèle ResNet152 avec la commande `setModelTypeAsResnet()` pensant que cette commande utilisait le modèle ResNet50. Le modèle utilisé pour la prédiction est différent du modèle utilisé pour la détection ce qui a causé cette erreur. Voici les couches de notre modèle:

Layer (type)	Output Shape	Param #	Connected to
=====			
=====			
input_2 (InputLayer)	(None, 224, 224, 3)	0	
-----			
conv2d_1 (Conv2D)	(None, 112, 112, 64)	9472	input_2[0][0]
-----			
batch_normalization_1 (BatchNorm)	(None, 112, 112, 64)	256	conv2d_1[0][0]
-----			
activation_1 (Activation)	(None, 112, 112, 64)	0	batch_normalization_1[0][0]
-----			
max_pooling2d_1 (MaxPooling2D)	(None, 55, 55, 64)	0	activation_1[0][0]
-----			
conv2d_3 (Conv2D)	(None, 55, 55, 64)	4160	max_pooling2d_1[0][0]
-----			
batch_normalization_3 (BatchNorm)	(None, 55, 55, 64)	256	conv2d_3[0][0]
-----			
activation_2 (Activation)	(None, 55, 55, 64)	0	batch_normalization_3[0][0]
-----			
conv2d_4 (Conv2D)	(None, 55, 55, 64)	36928	activation_2[0][0]
-----			
batch_normalization_4 (BatchNorm)	(None, 55, 55, 64)	256	conv2d_4[0][0]
-----			
activation_3 (Activation)	(None, 55, 55, 64)	0	batch_normalization_4[0][0]
-----			
conv2d_5 (Conv2D)	(None, 55, 55, 256)	16640	activation_3[0][0]

conv2d_2 (Conv2D)	(None, 55, 55, 256)	16640	max_pooling2d_1[0][0]
batch_normalization_5 (BatchNorm)	(None, 55, 55, 256)	1024	conv2d_5[0][0]
batch_normalization_2 (BatchNorm)	(None, 55, 55, 256)	1024	conv2d_2[0][0]
add_1 (Add)	(None, 55, 55, 256)	0	batch_normalization_5[0][0] batch_normalization_2[0][0]
activation_4 (Activation)	(None, 55, 55, 256)	0	add_1[0][0]
conv2d_6 (Conv2D)	(None, 55, 55, 64)	16448	activation_4[0][0]
batch_normalization_6 (BatchNorm)	(None, 55, 55, 64)	256	conv2d_6[0][0]
activation_5 (Activation)	(None, 55, 55, 64)	0	batch_normalization_6[0][0]
conv2d_7 (Conv2D)	(None, 55, 55, 64)	36928	activation_5[0][0]
batch_normalization_7 (BatchNorm)	(None, 55, 55, 64)	256	conv2d_7[0][0]
activation_6 (Activation)	(None, 55, 55, 64)	0	batch_normalization_7[0][0]
conv2d_8 (Conv2D)	(None, 55, 55, 256)	16640	activation_6[0][0]
batch_normalization_8 (BatchNorm)	(None, 55, 55, 256)	1024	conv2d_8[0][0]
add_2 (Add)	(None, 55, 55, 256)	0	batch_normalization_8[0][0] activation_4[0][0]

activation_7 (Activation)	(None, 55, 55, 256)	0	add_2[0][0]
conv2d_9 (Conv2D)	(None, 55, 55, 64)	16448	activation_7[0][0]
batch_normalization_9 (BatchNorm	(None, 55, 55, 64)	256	conv2d_9[0][0]
activation_8 (Activation)	(None, 55, 55, 64)	0	batch_normalization_9[0][0]
conv2d_10 (Conv2D)	(None, 55, 55, 64)	36928	activation_8[0][0]
batch_normalization_10 (BatchNor	(None, 55, 55, 64)	256	conv2d_10[0][0]
activation_9 (Activation)	(None, 55, 55, 64)	0	batch_normalization_10[0][0]
conv2d_11 (Conv2D)	(None, 55, 55, 256)	16640	activation_9[0][0]
batch_normalization_11 (BatchNor	(None, 55, 55, 256)	1024	conv2d_11[0][0]
add_3 (Add)	(None, 55, 55, 256)	0	batch_normalization_11[0][0] activation_7[0][0]
activation_10 (Activation)	(None, 55, 55, 256)	0	add_3[0][0]
conv2d_13 (Conv2D)	(None, 28, 28, 128)	32896	activation_10[0][0]
batch_normalization_13 (BatchNor	(None, 28, 28, 128)	512	conv2d_13[0][0]
activation_11 (Activation)	(None, 28, 28, 128)	0	batch_normalization_13[0][0]
conv2d_14 (Conv2D)	(None, 28, 28, 128)	147584	activation_11[0][0]

batch_normalization_14 (BatchNor (None, 28, 28, 128)	512	conv2d_14[0][0]
<hr/>		
activation_12 (Activation)	(None, 28, 28, 128) 0	batch_normalization_14[0][0]
<hr/>		
conv2d_15 (Conv2D)	(None, 28, 28, 512) 66048	activation_12[0][0]
<hr/>		
conv2d_12 (Conv2D)	(None, 28, 28, 512) 131584	activation_10[0][0]
<hr/>		
batch_normalization_15 (BatchNor (None, 28, 28, 512)	2048	conv2d_15[0][0]
<hr/>		
batch_normalization_12 (BatchNor (None, 28, 28, 512)	2048	conv2d_12[0][0]
<hr/>		
add_4 (Add)	(None, 28, 28, 512) 0	batch_normalization_15[0][0] batch_normalization_12[0][0]
<hr/>		
activation_13 (Activation)	(None, 28, 28, 512) 0	add_4[0][0]
<hr/>		
conv2d_16 (Conv2D)	(None, 28, 28, 128) 65664	activation_13[0][0]
<hr/>		
batch_normalization_16 (BatchNor (None, 28, 28, 128)	512	conv2d_16[0][0]
<hr/>		
activation_14 (Activation)	(None, 28, 28, 128) 0	batch_normalization_16[0][0]
<hr/>		
conv2d_17 (Conv2D)	(None, 28, 28, 128) 147584	activation_14[0][0]
<hr/>		
batch_normalization_17 (BatchNor (None, 28, 28, 128)	512	conv2d_17[0][0]
<hr/>		
activation_15 (Activation)	(None, 28, 28, 128) 0	batch_normalization_17[0][0]
<hr/>		
conv2d_18 (Conv2D)	(None, 28, 28, 512) 66048	activation_15[0][0]
<hr/>		
<hr/>		

batch_normalization_18 (BatchNor	(None, 28, 28, 512)	2048	conv2d_18[0][0]
----------------------------------	---------------------	------	-----------------

add_5 (Add)	(None, 28, 28, 512)	0	batch_normalization_18[0][0] activation_13[0][0]
-------------	---------------------	---	---

activation_16 (Activation)	(None, 28, 28, 512)	0	add_5[0][0]
----------------------------	---------------------	---	-------------

conv2d_19 (Conv2D)	(None, 28, 28, 128)	65664	activation_16[0][0]
--------------------	---------------------	-------	---------------------

batch_normalization_19 (BatchNor	(None, 28, 28, 128)	512	conv2d_19[0][0]
----------------------------------	---------------------	-----	-----------------

activation_17 (Activation)	(None, 28, 28, 128)	0	batch_normalization_19[0][0]
----------------------------	---------------------	---	------------------------------

conv2d_20 (Conv2D)	(None, 28, 28, 128)	147584	activation_17[0][0]
--------------------	---------------------	--------	---------------------

batch_normalization_20 (BatchNor	(None, 28, 28, 128)	512	conv2d_20[0][0]
----------------------------------	---------------------	-----	-----------------

activation_18 (Activation)	(None, 28, 28, 128)	0	batch_normalization_20[0][0]
----------------------------	---------------------	---	------------------------------

conv2d_21 (Conv2D)	(None, 28, 28, 512)	66048	activation_18[0][0]
--------------------	---------------------	-------	---------------------

batch_normalization_21 (BatchNor	(None, 28, 28, 512)	2048	conv2d_21[0][0]
----------------------------------	---------------------	------	-----------------

add_6 (Add)	(None, 28, 28, 512)	0	batch_normalization_21[0][0] activation_16[0][0]
-------------	---------------------	---	---

activation_19 (Activation)	(None, 28, 28, 512)	0	add_6[0][0]
----------------------------	---------------------	---	-------------

conv2d_22 (Conv2D)	(None, 28, 28, 128)	65664	activation_19[0][0]
--------------------	---------------------	-------	---------------------

batch_normalization_22 (BatchNor	(None, 28, 28, 128)	512	conv2d_22[0][0]
----------------------------------	---------------------	-----	-----------------

---

activation_20 (Activation)	(None, 28, 28, 128)	0	batch_normalization_22[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_23 (Conv2D)	(None, 28, 28, 128)	147584	activation_20[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_23 (BatchNor	(None, 28, 28, 128)	512	conv2d_23[0][0]
----------------------------------	---------------------	-----	-----------------

---

activation_21 (Activation)	(None, 28, 28, 128)	0	batch_normalization_23[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_24 (Conv2D)	(None, 28, 28, 512)	66048	activation_21[0][0]
--------------------	---------------------	-------	---------------------

---

batch_normalization_24 (BatchNor	(None, 28, 28, 512)	2048	conv2d_24[0][0]
----------------------------------	---------------------	------	-----------------

---

add_7 (Add)	(None, 28, 28, 512)	0	batch_normalization_24[0][0] activation_19[0][0]
-------------	---------------------	---	---

---

activation_22 (Activation)	(None, 28, 28, 512)	0	add_7[0][0]
----------------------------	---------------------	---	-------------

---

conv2d_26 (Conv2D)	(None, 14, 14, 256)	131328	activation_22[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_26 (BatchNor	(None, 14, 14, 256)	1024	conv2d_26[0][0]
----------------------------------	---------------------	------	-----------------

---

activation_23 (Activation)	(None, 14, 14, 256)	0	batch_normalization_26[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_27 (Conv2D)	(None, 14, 14, 256)	590080	activation_23[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_27 (BatchNor	(None, 14, 14, 256)	1024	conv2d_27[0][0]
----------------------------------	---------------------	------	-----------------

---

activation_24 (Activation)	(None, 14, 14, 256)	0	batch_normalization_27[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_28 (Conv2D)	(None, 14, 14, 1024)	263168	activation_24[0][0]
--------------------	----------------------	--------	---------------------

---

conv2d_25 (Conv2D)	(None, 14, 14, 1024)	525312	activation_22[0][0]
--------------------	----------------------	--------	---------------------

---

batch_normalization_28 (BatchNor	(None, 14, 14, 1024)	4096	conv2d_28[0][0]
----------------------------------	----------------------	------	-----------------

---

batch_normalization_25 (BatchNor	(None, 14, 14, 1024)	4096	conv2d_25[0][0]
----------------------------------	----------------------	------	-----------------

---

add_8 (Add)	(None, 14, 14, 1024)	0	batch_normalization_28[0][0] batch_normalization_25[0][0]
-------------	----------------------	---	--

---

activation_25 (Activation)	(None, 14, 14, 1024)	0	add_8[0][0]
----------------------------	----------------------	---	-------------

---

conv2d_29 (Conv2D)	(None, 14, 14, 256)	262400	activation_25[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_29 (BatchNor	(None, 14, 14, 256)	1024	conv2d_29[0][0]
----------------------------------	---------------------	------	-----------------

---

activation_26 (Activation)	(None, 14, 14, 256)	0	batch_normalization_29[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_30 (Conv2D)	(None, 14, 14, 256)	590080	activation_26[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_30 (BatchNor	(None, 14, 14, 256)	1024	conv2d_30[0][0]
----------------------------------	---------------------	------	-----------------

---

activation_27 (Activation)	(None, 14, 14, 256)	0	batch_normalization_30[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_31 (Conv2D)	(None, 14, 14, 1024)	263168	activation_27[0][0]
--------------------	----------------------	--------	---------------------

---

batch_normalization_31 (BatchNor	(None, 14, 14, 1024)	4096	conv2d_31[0][0]
----------------------------------	----------------------	------	-----------------

---

add_9 (Add)	(None, 14, 14, 1024)	0	batch_normalization_31[0][0] activation_25[0][0]
-------------	----------------------	---	---

---

---



activation_28 (Activation)	(None, 14, 14, 1024)	0	add_9[0][0]
<hr/>			
conv2d_32 (Conv2D)	(None, 14, 14, 256)	262400	activation_28[0][0]
<hr/>			
batch_normalization_32 (BatchNor	(None, 14, 14, 256)	1024	conv2d_32[0][0]
<hr/>			
activation_29 (Activation)	(None, 14, 14, 256)	0	batch_normalization_32[0][0]
<hr/>			
conv2d_33 (Conv2D)	(None, 14, 14, 256)	590080	activation_29[0][0]
<hr/>			
batch_normalization_33 (BatchNor	(None, 14, 14, 256)	1024	conv2d_33[0][0]
<hr/>			
activation_30 (Activation)	(None, 14, 14, 256)	0	batch_normalization_33[0][0]
<hr/>			
conv2d_34 (Conv2D)	(None, 14, 14, 1024)	263168	activation_30[0][0]
<hr/>			
batch_normalization_34 (BatchNor	(None, 14, 14, 1024)	4096	conv2d_34[0][0]
<hr/>			
add_10 (Add)	(None, 14, 14, 1024)	0	batch_normalization_34[0][0] activation_28[0][0]
<hr/>			
activation_31 (Activation)	(None, 14, 14, 1024)	0	add_10[0][0]
<hr/>			
conv2d_35 (Conv2D)	(None, 14, 14, 256)	262400	activation_31[0][0]
<hr/>			
batch_normalization_35 (BatchNor	(None, 14, 14, 256)	1024	conv2d_35[0][0]
<hr/>			
activation_32 (Activation)	(None, 14, 14, 256)	0	batch_normalization_35[0][0]
<hr/>			
conv2d_36 (Conv2D)	(None, 14, 14, 256)	590080	activation_32[0][0]
<hr/>			
<hr/>			

batch_normalization_36 (BatchNor (None, 14, 14, 256)	1024	conv2d_36[0][0]
<hr/>		
activation_33 (Activation)	(None, 14, 14, 256) 0	batch_normalization_36[0][0]
<hr/>		
conv2d_37 (Conv2D)	(None, 14, 14, 1024) 263168	activation_33[0][0]
<hr/>		
batch_normalization_37 (BatchNor (None, 14, 14, 1024)	4096	conv2d_37[0][0]
<hr/>		
add_11 (Add)	(None, 14, 14, 1024) 0	batch_normalization_37[0][0] activation_31[0][0]
<hr/>		
activation_34 (Activation)	(None, 14, 14, 1024) 0	add_11[0][0]
<hr/>		
conv2d_38 (Conv2D)	(None, 14, 14, 256) 262400	activation_34[0][0]
<hr/>		
batch_normalization_38 (BatchNor (None, 14, 14, 256)	1024	conv2d_38[0][0]
<hr/>		
activation_35 (Activation)	(None, 14, 14, 256) 0	batch_normalization_38[0][0]
<hr/>		
conv2d_39 (Conv2D)	(None, 14, 14, 256) 590080	activation_35[0][0]
<hr/>		
batch_normalization_39 (BatchNor (None, 14, 14, 256)	1024	conv2d_39[0][0]
<hr/>		
activation_36 (Activation)	(None, 14, 14, 256) 0	batch_normalization_39[0][0]
<hr/>		
conv2d_40 (Conv2D)	(None, 14, 14, 1024) 263168	activation_36[0][0]
<hr/>		
batch_normalization_40 (BatchNor (None, 14, 14, 1024)	4096	conv2d_40[0][0]
<hr/>		
add_12 (Add)	(None, 14, 14, 1024) 0	batch_normalization_40[0][0] activation_34[0][0]

---

activation_37 (Activation)	(None, 14, 14, 1024)	0	add_12[0][0]
----------------------------	----------------------	---	--------------

---

conv2d_41 (Conv2D)	(None, 14, 14, 256)	262400	activation_37[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_41 (BatchNor	(None, 14, 14, 256)	1024	conv2d_41[0][0]
----------------------------------	---------------------	------	-----------------

---

activation_38 (Activation)	(None, 14, 14, 256)	0	batch_normalization_41[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_42 (Conv2D)	(None, 14, 14, 256)	590080	activation_38[0][0]
--------------------	---------------------	--------	---------------------

---

batch_normalization_42 (BatchNor	(None, 14, 14, 256)	1024	conv2d_42[0][0]
----------------------------------	---------------------	------	-----------------

---

activation_39 (Activation)	(None, 14, 14, 256)	0	batch_normalization_42[0][0]
----------------------------	---------------------	---	------------------------------

---

conv2d_43 (Conv2D)	(None, 14, 14, 1024)	263168	activation_39[0][0]
--------------------	----------------------	--------	---------------------

---

batch_normalization_43 (BatchNor	(None, 14, 14, 1024)	4096	conv2d_43[0][0]
----------------------------------	----------------------	------	-----------------

---

add_13 (Add)	(None, 14, 14, 1024)	0	batch_normalization_43[0][0] activation_37[0][0]
--------------	----------------------	---	---

---

activation_40 (Activation)	(None, 14, 14, 1024)	0	add_13[0][0]
----------------------------	----------------------	---	--------------

---

conv2d_45 (Conv2D)	(None, 7, 7, 512)	524800	activation_40[0][0]
--------------------	-------------------	--------	---------------------

---

batch_normalization_45 (BatchNor	(None, 7, 7, 512)	2048	conv2d_45[0][0]
----------------------------------	-------------------	------	-----------------

---

activation_41 (Activation)	(None, 7, 7, 512)	0	batch_normalization_45[0][0]
----------------------------	-------------------	---	------------------------------

---

conv2d_46 (Conv2D)	(None, 7, 7, 512)	2359808	activation_41[0][0]
--------------------	-------------------	---------	---------------------

---

batch_normalization_46 (BatchNor	(None, 7, 7, 512)	2048	conv2d_46[0][0]
----------------------------------	-------------------	------	-----------------

---

activation_42 (Activation)	(None, 7, 7, 512)	0	batch_normalization_46[0][0]
----------------------------	-------------------	---	------------------------------

---

conv2d_47 (Conv2D)	(None, 7, 7, 2048)	1050624	activation_42[0][0]
--------------------	--------------------	---------	---------------------

---

conv2d_44 (Conv2D)	(None, 7, 7, 2048)	2099200	activation_40[0][0]
--------------------	--------------------	---------	---------------------

---

batch_normalization_47 (BatchNor	(None, 7, 7, 2048)	8192	conv2d_47[0][0]
----------------------------------	--------------------	------	-----------------

---

batch_normalization_44 (BatchNor	(None, 7, 7, 2048)	8192	conv2d_44[0][0]
----------------------------------	--------------------	------	-----------------

---

add_14 (Add)	(None, 7, 7, 2048)	0	batch_normalization_47[0][0] batch_normalization_44[0][0]
--------------	--------------------	---	--

---

activation_43 (Activation)	(None, 7, 7, 2048)	0	add_14[0][0]
----------------------------	--------------------	---	--------------

---

conv2d_48 (Conv2D)	(None, 7, 7, 512)	1049088	activation_43[0][0]
--------------------	-------------------	---------	---------------------

---

batch_normalization_48 (BatchNor	(None, 7, 7, 512)	2048	conv2d_48[0][0]
----------------------------------	-------------------	------	-----------------

---

activation_44 (Activation)	(None, 7, 7, 512)	0	batch_normalization_48[0][0]
----------------------------	-------------------	---	------------------------------

---

conv2d_49 (Conv2D)	(None, 7, 7, 512)	2359808	activation_44[0][0]
--------------------	-------------------	---------	---------------------

---

batch_normalization_49 (BatchNor	(None, 7, 7, 512)	2048	conv2d_49[0][0]
----------------------------------	-------------------	------	-----------------

---

activation_45 (Activation)	(None, 7, 7, 512)	0	batch_normalization_49[0][0]
----------------------------	-------------------	---	------------------------------

---

conv2d_50 (Conv2D)	(None, 7, 7, 2048)	1050624	activation_45[0][0]
--------------------	--------------------	---------	---------------------

---

batch_normalization_50 (BatchNor (None, 7, 7, 2048)	8192	conv2d_50[0][0]
---	------	-----------------

---

add_15 (Add)	(None, 7, 7, 2048)	0	batch_normalization_50[0][0] activation_43[0][0]
--------------	--------------------	---	---

---

activation_46 (Activation)	(None, 7, 7, 2048)	0	add_15[0][0]
----------------------------	--------------------	---	--------------

---

conv2d_51 (Conv2D)	(None, 7, 7, 512)	1049088	activation_46[0][0]
--------------------	-------------------	---------	---------------------

---

batch_normalization_51 (BatchNor (None, 7, 7, 512)	2048	conv2d_51[0][0]
--	------	-----------------

---

activation_47 (Activation)	(None, 7, 7, 512)	0	batch_normalization_51[0][0]
----------------------------	-------------------	---	------------------------------

---

conv2d_52 (Conv2D)	(None, 7, 7, 512)	2359808	activation_47[0][0]
--------------------	-------------------	---------	---------------------

---

batch_normalization_52 (BatchNor (None, 7, 7, 512)	2048	conv2d_52[0][0]
--	------	-----------------

---

activation_48 (Activation)	(None, 7, 7, 512)	0	batch_normalization_52[0][0]
----------------------------	-------------------	---	------------------------------

---

conv2d_53 (Conv2D)	(None, 7, 7, 2048)	1050624	activation_48[0][0]
--------------------	--------------------	---------	---------------------

---

batch_normalization_53 (BatchNor (None, 7, 7, 2048)	8192	conv2d_53[0][0]
---	------	-----------------

---

add_16 (Add)	(None, 7, 7, 2048)	0	batch_normalization_53[0][0] activation_46[0][0]
--------------	--------------------	---	---

---

activation_49 (Activation)	(None, 7, 7, 2048)	0	add_16[0][0]
----------------------------	--------------------	---	--------------

---

global_avg_pooling (GlobalAverag (None, 2048)	0	activation_49[0][0]
---	---	---------------------

---

---

dense_1 (Dense)	(None, 10)	20490	global_avg_pooling[0][0]
-----------------	------------	-------	--------------------------

---

activation_50 (Activation)	(None, 10)	0	dense_1[0][0]
----------------------------	------------	---	---------------

---

---

Total params: 23,608,202

Trainable params: 23,555,082

Non-trainable params: 53,120

---

---

Using Enhanced Data Generation

Found 6909 images belonging to 4 classes.

Found 27128 images belonging to 4 classes.

JSON Mapping for the model classes saved to

C:\Users\User\modern\_car\_racing\json\model\_class.json

Number of experiments (Epochs) : 30

---

---

Le fichier training.log:

[https://github.com/simonlafreniere/.../getData/Modern\\_Car\\_Racing/training.txt](https://github.com/simonlafreniere/.../getData/Modern_Car_Racing/training.txt)

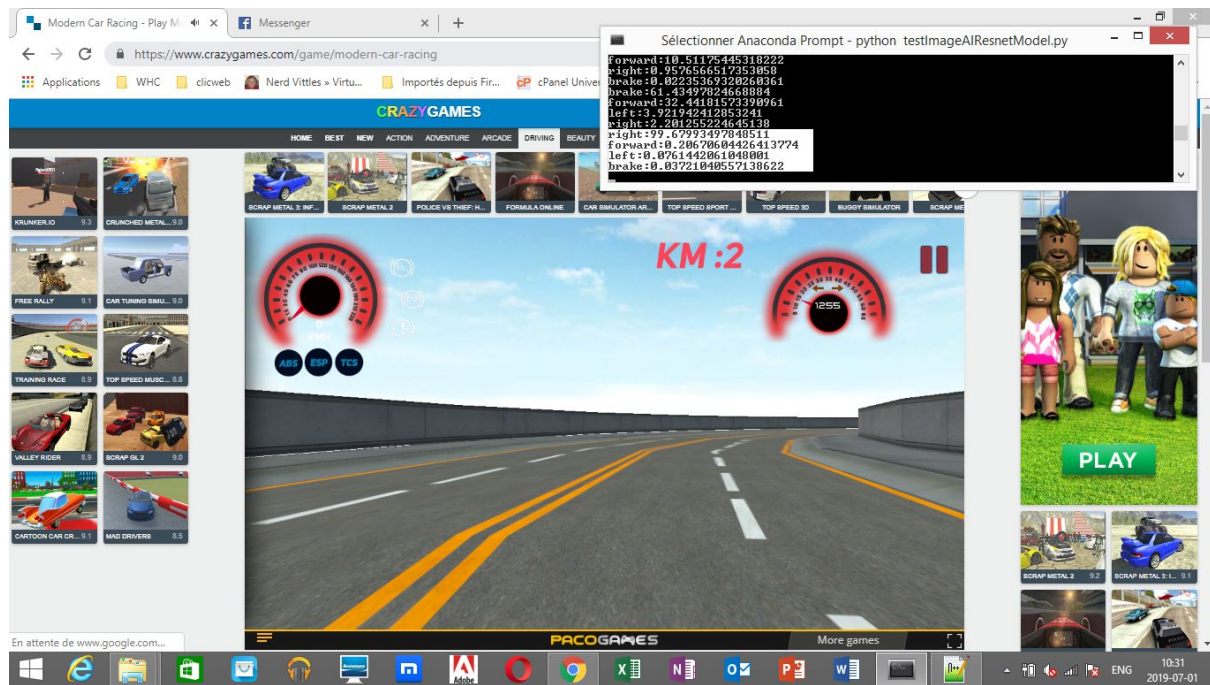
Lien vers les modèles générés:

[https://github.com/simonlafreniere/...Modern\\_Car\\_Racing/models](https://github.com/simonlafreniere/...Modern_Car_Racing/models)

## Les résultats

La réalisation de ce projet demandait au départ au minimum un GPU NVIDIA récent afin de disposer de la puissance de calcul nécessaire pour l'entraînement du modèle et le traitement sur les flux. Puisque le modèle final utilisé est très lourd, une carte GeForce GTX 1060 possédant 1280 cuda cores n'est pas assez performante pour effectuer des tests en mouvement en temps réel puisque le temps de calcul dépasse fortement le temps critique de réaction, il est cependant possible de faire des tests stationnaires comme par exemple sur les images suivantes ou la prédiction nous indique la bonne action à prendre et cela dans tous les cas que nous avons testés:





Nous avons utilisé un poste de travail du bureau où Simon travaille qui est monté avec 2 cartes GEFORCE RTX 2070 possédant chacune 2304 cuda cores pour effectuer une capture vidéo de notre code en action. Il ne fut pas possible d'utiliser le proximitySensor simultanément car encore une fois la puissance de calcul a été atteinte. Par chance le jeu nous permet de jouer avec ou sans circulation d'autres véhicules. Nous avons choisi de faire la vidéo sans circulation en utilisant pas le proximitySensor, nous utilisons que le [speedSensor](#) pour réguler la vitesse, ce dernier n'utilise pas beaucoup de ressources.

Voici un lien vers la vidéo:

<https://youtu.be/TXMGFdO8wPA>

## La conclusion

La réalisation de ce simulateur est basée sur l'approche du tutoriel de simulation du jeu GTA5 <https://pythonprogramming.net/game-frames-open-cv-python-plays-gta-v/>. Ce tutoriel a sûrement déjà été exploité comme ressource par d'autres équipes, ce fut même le cas dans notre propre groupe ou un autre équipe semble avoir utilisé cette même référence comme base de départ. Bien que nous nous sommes inspiré de cette ressource nous avons utilisé notre propre code pour réaliser sensiblement les mêmes tâches mais avec des résultats beaucoup plus satisfaisant sauf au niveau des ressources matériel. À cause de notre erreur au niveau du choix du modèle, nous avons mal évalué la lourdeur de ResNet pour la



prédiction des images. Le code en tant que tel est solide et fonctionne comme il se doit, par contre il demande un matériel très performant et très coûteux.

Nous avons appliqué toutes les notions vues durant le cours, le développement de notre programme nous a permis d'expérimenter et d'approfondir nos connaissances en matière d'intelligence artificielle et de traitement de l'image.

Si cela serait à refaire, nous passerions probablement le plus de temps à lire et à tester différents modèles sans utiliser ImageAI car cette librairie restreint le choix à 4 modèles lourds, qui demandent des ressources matérielles importantes. L'approche TFLearn + AlexNet demande peu de ressource matériel mais est très peu performante, le but serait de trouver le modèle qui donne de bons résultats avec un matériel de moyenne gamme.