Mankouri Jalil |   |



Table des matières

[Perspective d’évolution 1](#_Toc508811981)

[Architecture du système 2](#_Toc508811982)

[I- Google Cloud Platform 2](#_Toc508811983)

[1) Cloud Datalab 3](#_Toc508811984)

[2) Cloud Dataproc 4](#_Toc508811985)

[3) BigQuery 5](#_Toc508811986)

[II- TensorFlow et ses réseaux de neurones 7](#_Toc508811987)

[1) L’inférence 7](#_Toc508811988)

[2) Fonction de coût 8](#_Toc508811989)

[3) Entrainement 8](#_Toc508811990)

[Solutions 9](#_Toc508811991)

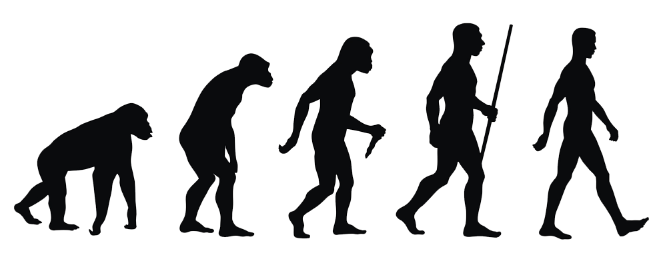
[I- Nos données pour l’apprentissage 9](#_Toc508811992)

[II- Optimisation du réseau de neurones 10](#_Toc508811993)

[III- Résultats 11](#_Toc508811994)

[Bibliographie 12](#_Toc508811995)

# Perspective d’évolution



L’objectif premier de cette perspective d’évolution de notre projet serait d’analyser toute sorte d’image de nos utilisateurs automatiquement.

Lister le contenu présent sur chaque image pourrait nous permettre d’affiner le profil de nos utilisateurs au fur et à mesure qu’ils utilisent notre application.



Tout cela afin de leurs proposer ensuite un meilleur contenu à découvrir, donc de toucher la dimension marketing de STYLBUM en fidélisant les utilisateurs par le biais d’une excellente expérience utilisateur.

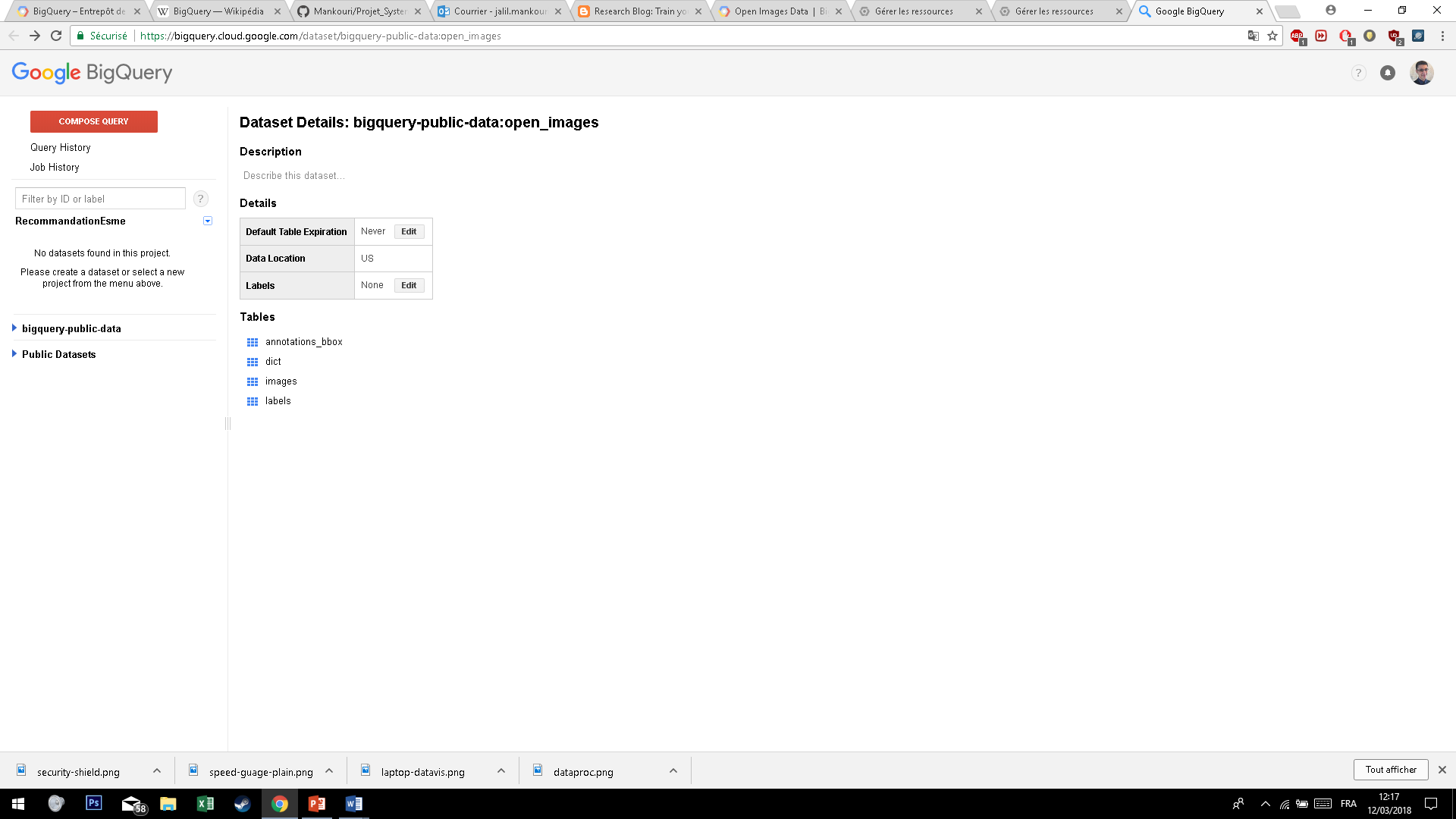


# Architecture du système

## Google Cloud Platform



Google Cloud Platform est une des solutions fournies par Google de Cloud Computing. Elle propose un hébergement sur la même infrastructure que celle utilisée par son moteur de recherche. Google Cloud Platform est donc une plateforme fournissant aux développeurs des services permettant de construire plus aisément des sites web comme des applications complexes comme le Deep Learning.



Basé sur le cloud, cette plateforme met à disposition une multitude de services modulaires comme le stockage d’informations, le calcul, des applications de traduction de texte et prévision, etc…

Notre choix c’est porté sur Google Cloud Platform plutôt qu’AWS d’Amazon pour ses prix concurrentiel, sa facilité d’utilisation, la performance qu’elle fournit malgré la gratuité de sa solution et ses services que nous détaillerons plus bas.

### Cloud Datalab



Cloud Datalab est une solution Open Source interactive permettant d’analyser, parcourir, transformer et visualiser des données en plus de fournir la possibilité d’élaborer des modèles de Machine Learning depuis la plateforme principale.



Cloud Datalab a pour but de facilité la compréhension de nos analyses en matière de science des données. Il se combine parfaitement avec BigQuery, Cloud Storage et des programmes Python.

Ce qui nous a particulièrement intéressé était sa capacité à créer des modèles de Machine Learning à partir de nos données pour faire des prédictions tout en les évaluant et optimisant à l’aide de TensorFlow.

### Cloud Dataproc



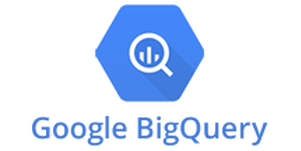
Afin de traiter rapidement de grand set de données, nous avons de plus souvent besoin de recourir aux services d’Apache comme Hadoop, Spark, Pig et Hive.

Cloud Dataproc nous apporte ses solutions afin de créer rapidement des clusters de tailles variable, désactivable quand le besoin ne se fait plus ressentir afin de réduire les coûts.

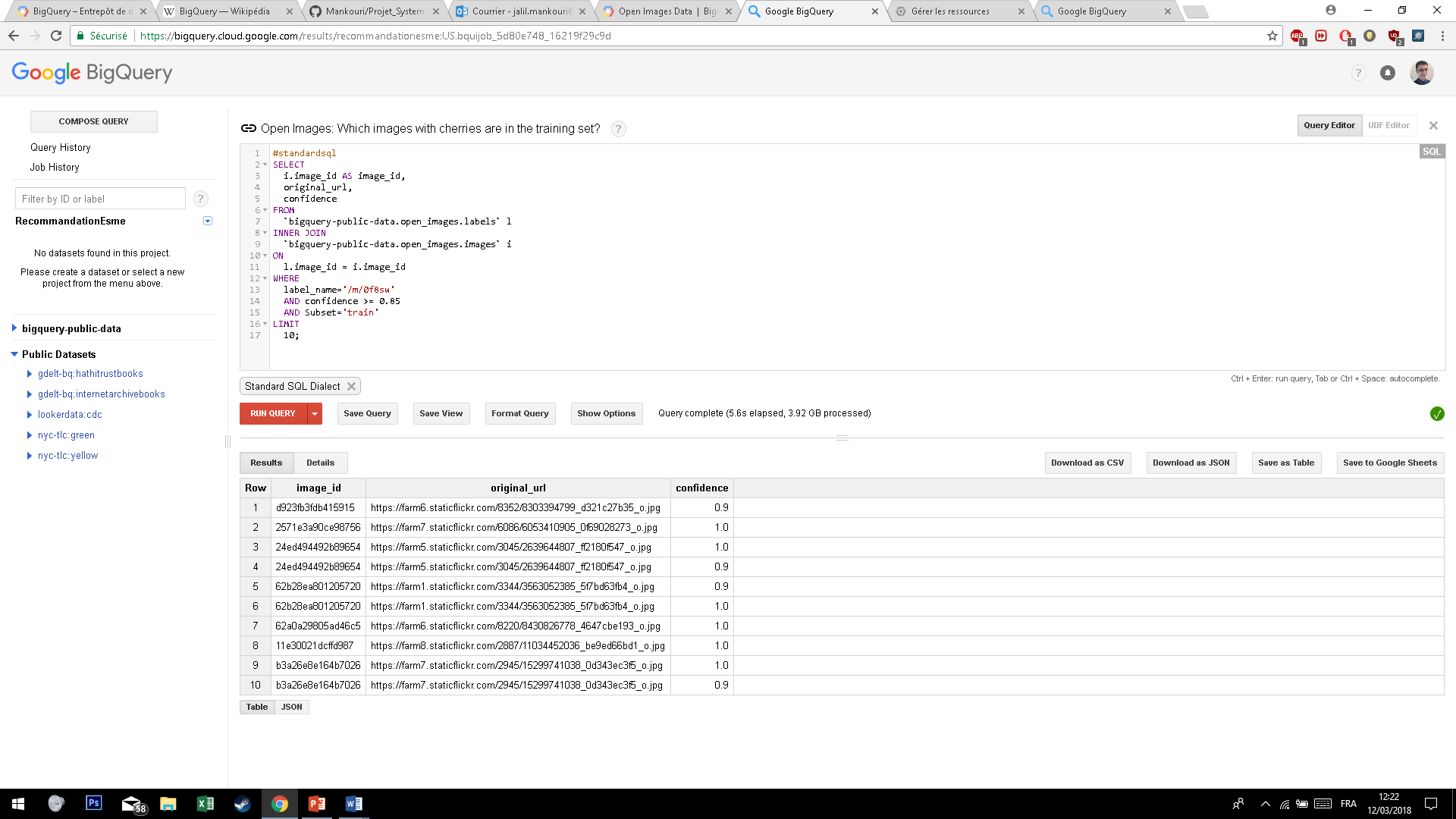


Ce qui nous a tout de suite plu dans les services que proposait Google Cloud Platform était que chaque outil que nous implémentions dans notre solution était complémentaire d’un autre. Par exemple, il est pour nous envisageable que nous utilisions, en association de BigQuery, Hive sur Hadoop. Ou bien si nous voulions créer des pipelines de transformation de données à l’aide de Spark, nous pourrions utiliser un autre service appelé Cloud Dataflow.

### BigQuery



BigQuery nous permet d’analyser jusqu’à des pétaoctets de données, tout cela sans avoir aucune infrastructure à gérer étant donné qu’il fonctionne sans serveur. Cela nous permet donc de nous concentrer sur le traitement de nos données afin d’en dégager les informations dont nous avons besoin par le biais de requêtes SQL puissante et optimiser pour de grand ensemble.

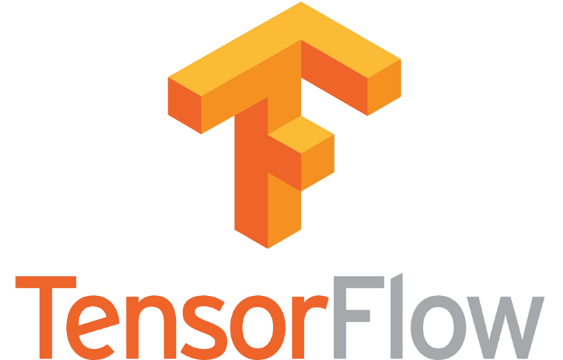


BigQuery est très utilisé dans le domaine du Big Data, qu’il s’agisse de start-up ou d’entreprises présentent dans le CAC40.



Ce que nous aimons par-dessus tout est l’externalisation de la sécurisation de nos services, en effet Google nous protège des menaces extérieures en sécurisant nos applications en chiffrant et répliquant nos données sur plusieurs datacenters pour nous offrir une sécurité, disponibilité et durabilité maximale.

## TensorFlow et ses réseaux de neurones



TensorFlow est une librairie open-source pour la programmation de taches traitants des flux de données, elle est utilisée dans des applications d’apprentissage automatique telles que les réseaux de neurones. Sa force réside dans son efficacité à faire avancer nos recherches et productions.

Les calculs qu’effectue TensorFlow sont représentés par le biais de graphes de flux de données avec état. Etant donné que l’on appelle « tenseurs » les tableaux de données multidimensionnels des réseaux de neurones sur lequel on applique des opérations complexes.

Nous avons du donc apprendre à manipuler cette puissante librairie spécialisée dans l’apprentissage profond. (Deep Learning)

### L’inférence

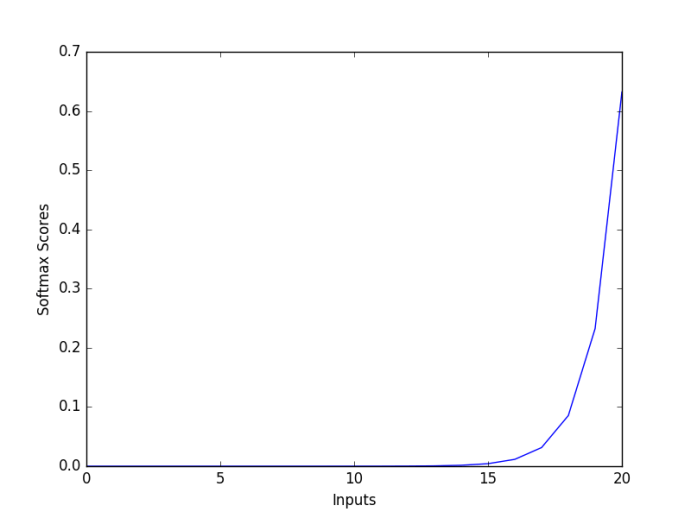


L’inférence représente toutes les étapes dont nous avons besoin afin de réaliser des prédictions des classes à partir des images de notre set d’images. C’est là que nous construisons l’architecture de notre réseau de neurone.

TensorFlow nous permet à cette étape pour chaque image de calculer un vecteur de probabilités grâce à l’aide d’une « Softmax Regression ».

On note que si les valeurs des poids et des biais sont hautes, les neurones auront tendance à apprendre la même chose…

Pour visualiser la variation de valeur du softmax, TensorFlow nous fournis un histogramme :



### Fonction de coût



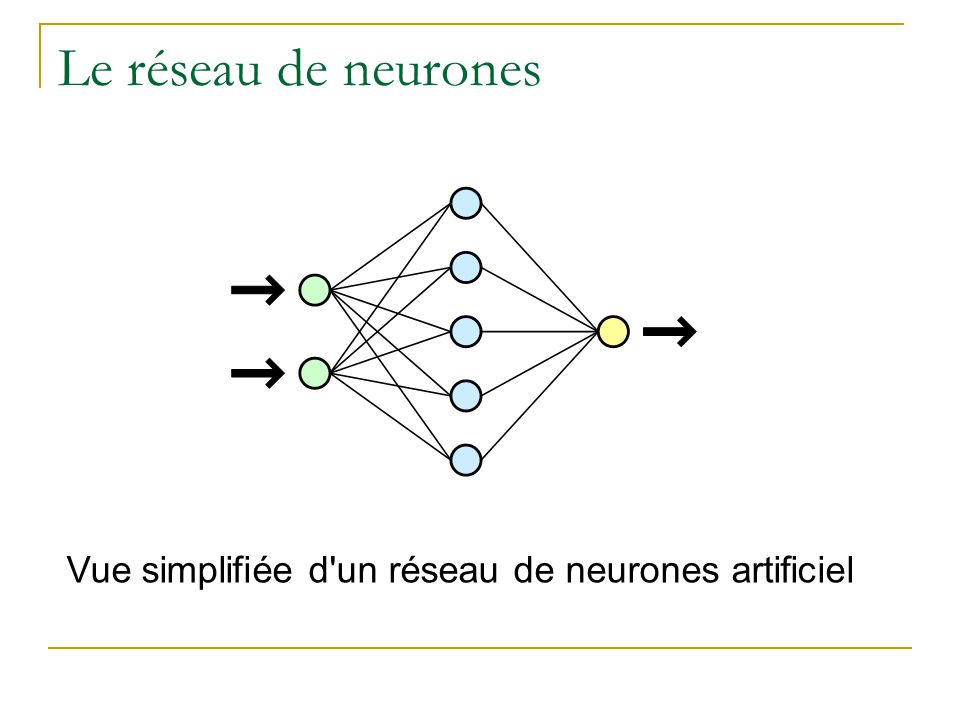
Pour que notre réseau de neurone obtienne des résultats cohérents, il nous faut définir une fonction de coût que l’on optimise à chaque nouvelle itération de notre programme. Celle-ci sert donc à diriger l’apprentissage du réseau pendant laquelle on cherche à minimiser l’erreur de prédiction.

L’entropie croisée (cross-entropie) est généralement utilisée comme fonction de coût. Elle a la particularité de mesurer le nombre de bits moyen dont nous avons besoin pour identifier un évènement (tags) issue d’un ensemble d’évènement (dictionnaire)

Pour faire simple, elle nous permettra de mesurer à quel point les prédictions que nous aurons calculées sont éloignées de la vérité. Les poids et les biais que nous avons vu plus haut seront donc mis à jour par le biais d’une fonction appelé à chaque itération de notre programme.

### Entrainement

L’entrainement représente l’étape d’optimisation de notre réseau de neurone. On procède en affectant des règles sur l’attribution des poids de notre système (pour chaque itération).



On s’appuie sur la méthode de rétropropagation du gradient qui calcule le gradient de l’erreur pour chaque neurone de notre système (sur toute les couches) combiné avec des algorithmes par exemple la descente du gradient stochastique.

C’est donc cette erreur que nous cherchons à minimiser !

Pour cela, il nous faut définir la vitesse d’entrainement de notre système que l’on appellera Learning Rate. Cette variable impacte la variation des poids et des biais de notre réseau de neurone.

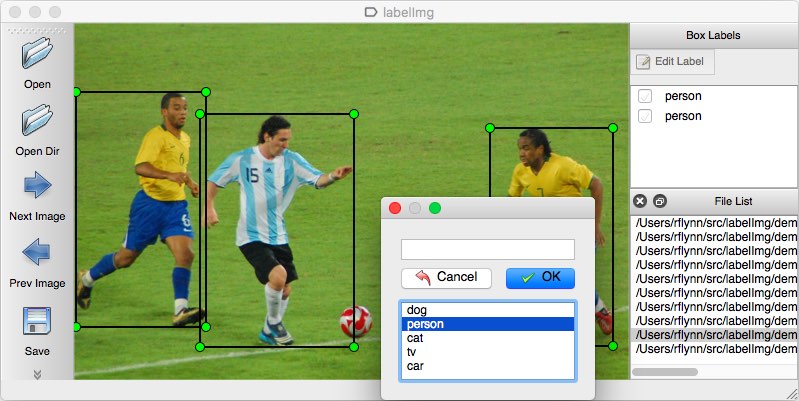
Selon les valeurs que prendront cette variable, la convergence de notre système se verra différente.

Avec un Learning Rate élevé, on observe de nette variations des poids et biais quand l’erreur est jugée importante. Point négatif, il ne faut pas que cette valeur soit trop élevée sinon notre système diverge.

# Solutions

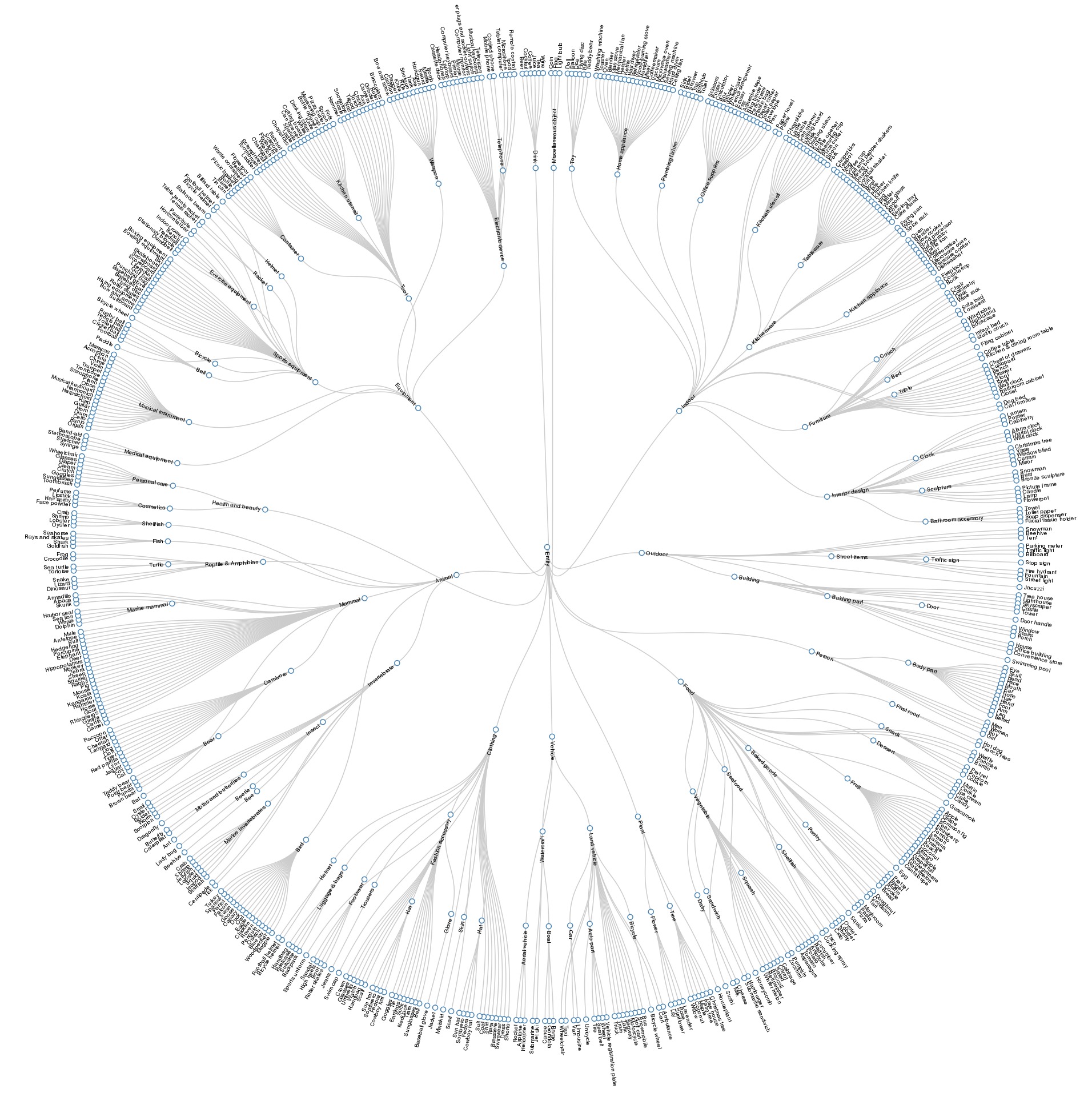
## Nos données pour l’apprentissage

Pour l’apprentissage de notre réseau de neurone, nous nous sommes appuyés sur un grand set de donnée d’images. On compte parmi celui-ci plus de 9 millions d’images différentes sur lesquels des utilisateurs ont délimiter des zones en détaillant ce qu’il s’y trouvait par le biais de Tag/Label.



Sur ces images, plus de 20 millions de Tag/Label ont été enregistré. Etant donnée le volume conséquent de donnée à traiter, la plateforme Google Cloud Platform a été d’un grand secours afin d’assurer une bonne performance pour nos traitements.

## Optimisation du réseau de neurones



Une fois notre réseau de neurone entrainé, il nous a fallu optimiser les paramètres le régissant afin de l’adapter au maximum au modèle de donnée que nous lui fournissons. Après plusieurs tests, en changeant différents paramètres comme le Learning Rate, le nombre de couches de notre réseau de neurone ou bien tout simplement le nombre de neurone de notre système, etc, nous avons tout en gardant une cohérence acceptable sur nos résultats diviser le temps de traitement par 3 dans la plupart des sets de tests.

TensorBoard et ses visualisations des variations des paramètres furent très utile dans cette étape délicate de notre projet.

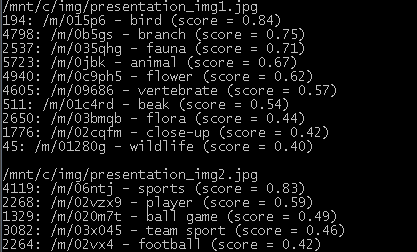
## Résultats

En choisissant arbitrairement un score de 40% (>0,40) en-dessous duquel nous n’afficherons pas les Tags/Labels que notre algorithme aurait décelé sur les images que nous lui auront passé en entrée.

Il nous suffit donc d’indiquer à notre programme python où chercher les images à analyser puis d’attendre que la magie de TensorFlow et ses réseaux de neurones opèrent !



Comme nous partons du principe qu’un exemple vaut mieux qu’un long discours, en partant de deux images (voir ci-dessus), nous arrivons à repérer le contenu de chaque image ainsi que ce que les antagonistes de celles-ci font :



# Bibliographie

**Aucune source spécifiée dans le document actif.**