**DEEP LEARNING**

1. Table des matières

[I. Table des matières 1](#_Toc46727652)

[II. Théorique 2](#_Toc46727653)

[1) Calcul de j1 3](#_Toc46727654)

[2) Calcul de j2 4](#_Toc46727655)

[3) Calcul de o1 4](#_Toc46727656)

[4) Calcul de o2 5](#_Toc46727657)

[5) Analyse des résultats 5](#_Toc46727658)

[6) Calcul de l’erreur de o1 et o2 5](#_Toc46727659)

[7) Calcul de l’impact de chaque poids (gradient) 6](#_Toc46727660)

[8) La fonction de normalisation : StandardScaler 11](#_Toc46727661)

[9) La fonction d’erreur : compile 12](#_Toc46727662)

[10) Jeu d’entrainement, jeu de validation et jeu de tests 13](#_Toc46727663)

[III. Installer une VM Debian 9 15](#_Toc46727664)

[IV. Installer « vmware tools » sur Debian 9 15](#_Toc46727665)

[V. Installer JUPYTER sur DEBIAN 9 (facultatif) 16](#_Toc46727666)

[11) Mettre à jour les paquets 16](#_Toc46727667)

[12) Installer Python 3 et pip 16](#_Toc46727668)

[13) Créer un environnement virtuel 16](#_Toc46727669)

[Installer virtualenv 16](#_Toc46727670)

[Créer un répertoire appelé machinelearning 16](#_Toc46727671)

[14) Installer et utiliser JUPYTER 16](#_Toc46727672)

[15) Installer et utiliser Vscode 17](#_Toc46727673)

[VI. Installer les dépendances python et les packages utilisés 17](#_Toc46727674)

[I. Sources 17](#_Toc46727675)

[16) Machine learning générale 17](#_Toc46727676)

[17) Faire du deep learning 17](#_Toc46727677)

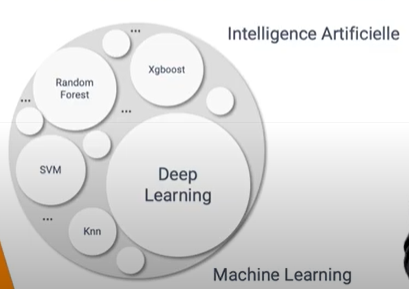
[VII. Cours Pratique 18](#_Toc46727678)

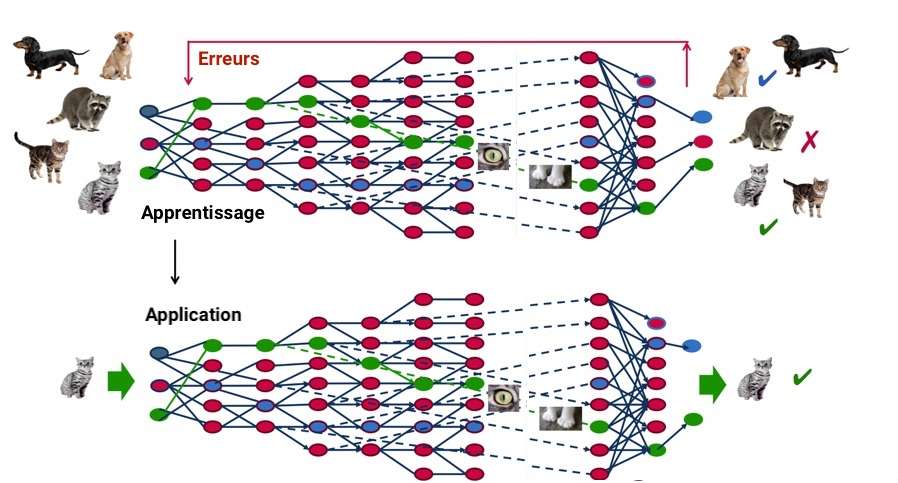
[VIII. Glossaire 18](#_Toc46727679)

[IX. Autres 18](#_Toc46727680)

1. Théorique

Le Deep Learning = un ensemble de méthodes du machine Learning



Le Deep Learning a un algorithme diffèrent des autres méthodes. Il peut créer des réseaux de neurones. Il peut apprendre à partir de beaucoup de données (exemple : images, mots, sons). 

Le modèle apprend grâce à une base de données. Il pourra, en fonction des images de références stockées dans la base, reconnaitre l’image que l’on a montré.

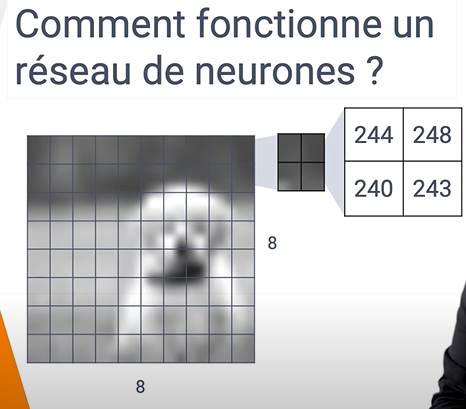
Il existe plusieurs Framework de deep Learning :



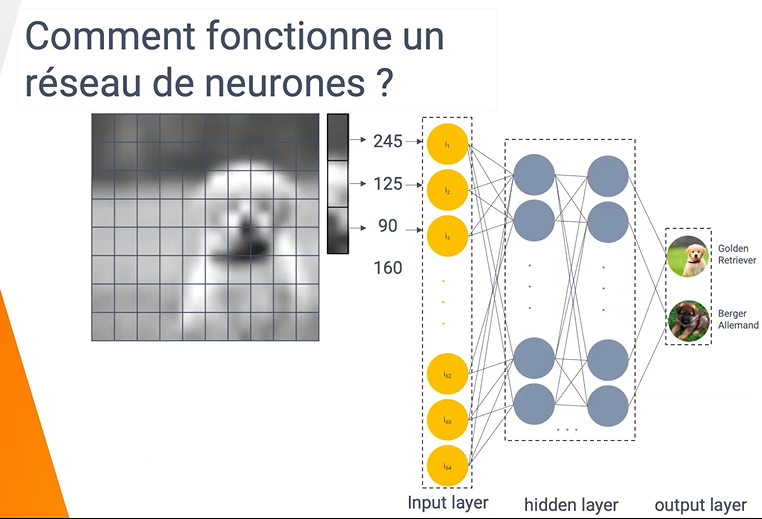
On a choisi tensorflow pour débuter (utilisé avec python)

**Tensorflow**: c’est une suite d’informations (on l’appelle un réseau de neurones) pour pouvoir mettre en place un modèle qui va nous permettre de faire des prédictions.

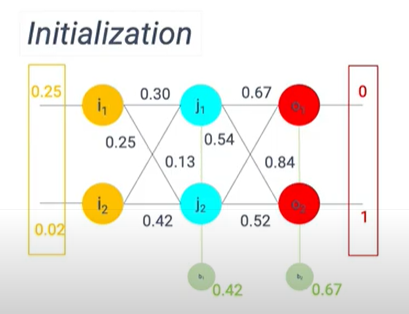
Sur une photo, on récupère chaque pixel on récupère l’information du pixel (entre 0 et 255 valeurs car c’est en noir et blanc).



Il existe 3 couches pour faire fonctionner un réseau de neurones : input, layer et output.



Tout d’abord nous allons initialiser le modèle.



L’objectif est qu’en fonction des valeurs en jaune, on puisse avec 0 et 1 en sortie (rouge). En jaune, les données de l’image en fonction des couleurs (de 0 à 1)

## Calcul de j1

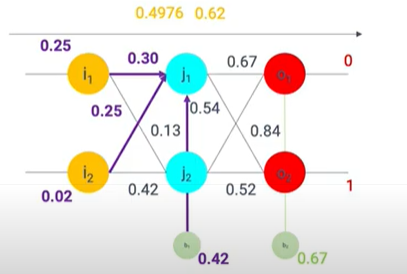
Les deux entrées vont vers j1 donc le calcul est :

j1 = i1\*w1 + i2\*w2 + b1

j1= 0.25\*0.30+0.02\*0.25+0.42

j1 = 0.075+0.005+0.42

j1 =0.5



Maintenant nous devons calculer la valeur de j1 livrera au reste du réseau :

F() est la fonction d’activation. Elle sert à rompre la linéarité de notre code de réseau neuronal.

F(j1) =

On refait la même chose avec j2.

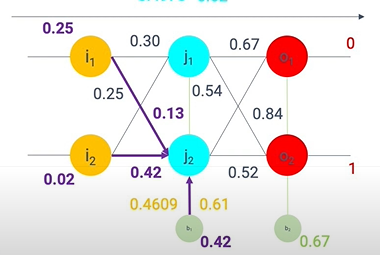
## Calcul de j2

j2 = i1\*w3 + i2\*w4 + b1

j2 = 0.25\*0.13 + 0.02\*0.42 + 0.42

j2 = 0.0325 + 0.0086 +0.42

j2 = 0.4609



F(j2) =

## Calcul de o1

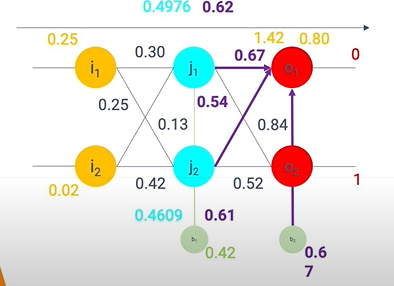
o1 = f(j1)\*w5 + f(j2)\*w6 + b2

o1 = 0.62\*0.67 + 0.61\*0.54 + 0.67

o1 = 0.42 + 0.33 + 0.67

o1 = 1.42

F(o1) =



## Calcul de o2

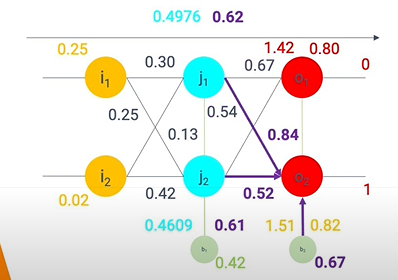
o2 = f(j1)\*w5 + f(j2)\*w6 + b2

o2 = 0.62\*0.61 + 0.61\*0.52 + 0.67

o2 = 0.52 + 0.32 + 0.67

o2 = 1.51

F(o2) =



## Analyse des résultats

L’objectif est qu’on ait :

* 0 en sortie de o1
* 1 en sortie de o2

Ce que nous avons :

* 0.80 en sortie de o1
* 0.82 en sorite de o2

**Il y a donc un grand écart entre l’objectif et ce que nous avons.**

Nous devons donc ajuster les poids pour nous rapprocher les valeurs souhaitées en sortie.

Pour chaque neurone nous allons calculer l’erreur :

Etotal =

## Calcul de l’erreur de o1 et o2

Eo1 =

Eo1 =

L’erreur de o1 est de 0.32

Eo2 =

Eo2 =

L’erreur de o1 est de 0.0162

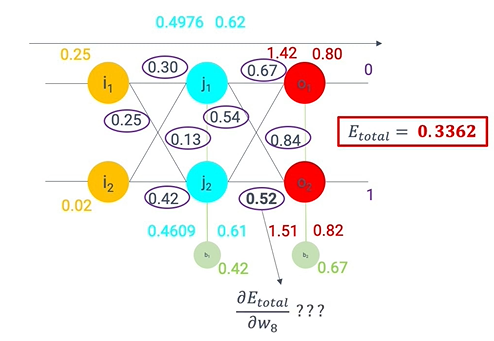
Etotal = Eo1 + Eo2 = 0.32 + 0.0162 = 0.3362

L’erreur totale est de 0.3362

## Calcul de l’impact de chaque poids (gradient)

Nous allons calculer l’impact de chaque poids pour réduire l’erreur de w8 dans la prochaine itération.

### Impact de w8 sur le total (gradient) :



= gradient de w8 sur le total

=

outo2 = = 0.82

=

= outo2 (1-outo2) = 0.82(1-0.82) = 0.1476

=

L’erreur de w8 est de -0.016

Nous allons donc calculer la valeur de la mise a jour de w8.

## Calcul de la mise à jour de w8

Taux d’apprentissage ?

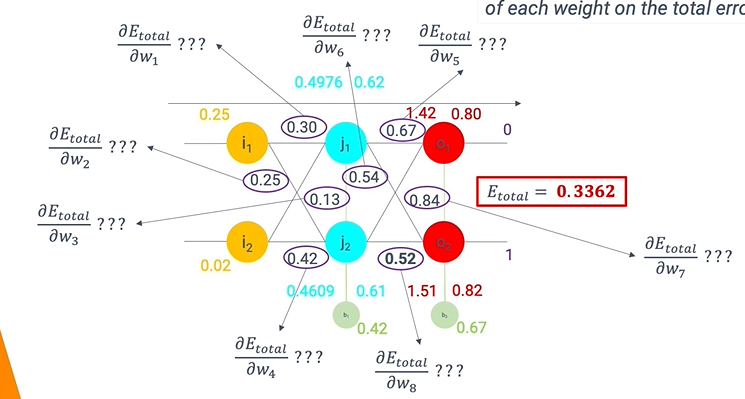
W8updated = w8 - taux d’apprentissage \* = 0.52-0.8 \* -0.016 = 0.5328

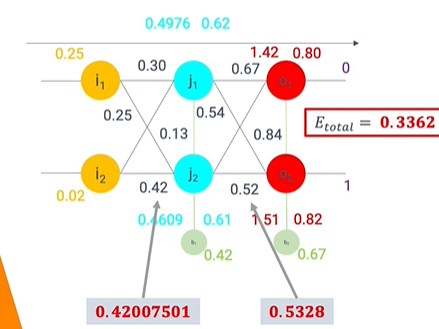
La mise a jour de w8 pour a nouvelle itération est de 0.5328

Maintenant que nous avons calculé la mise a jour de w8, il faut calculer la mise a jour de tous les autres poids.

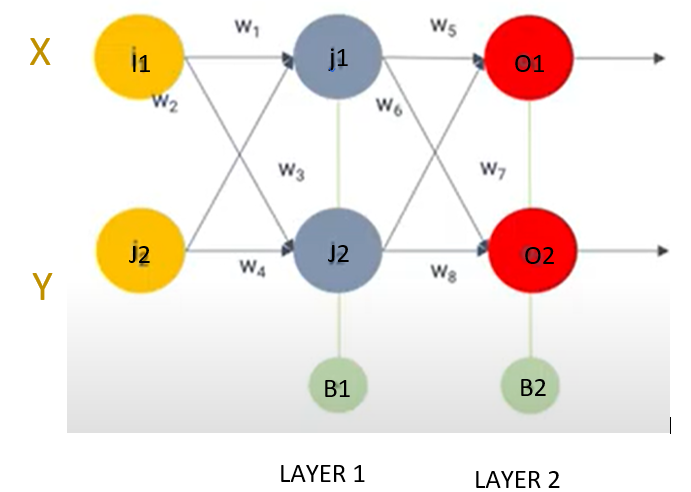
## Calcul de tous les autres poids

Nous n’allons pas calculer tous les autres poids à la main car le Framework tensorflow le fait automatiquement.





Démonstration



Sans Layers



Avec Layers



Ajout de normalisation sur des données linéaires en entrée (entrée valeur des pixels)

Il faut que les données soient proches de 0. On fait cela pour ne pas avoir des effets d’échelle (des nombres trop grands et des nombres trop petits)

## La fonction de normalisation : StandardScaler

Améliorer le traitement des données lorsqu’elles sont linéaires en entrée (valeurs des pixels). Cela améliore la structure des neurones.

Cela sert à diminuer le niveau d’échelle des images (0 à 255). Pour diminuer l’erreur on va diminuer les poids en entrée (recentrer toutes les valeurs autour de 0). L’objectif est d’augmenter la vitesse de calcul pour converger vers une solution optimale.

**StandardScaler c’est :**

**Z=(X-U) /S**

X est la valeur du pixel en entrée

U est la moyenne de toutes les entrées

S est l’écart type de toutes des entrées

Z est le résultat

**En code cela donne :**

From sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

images = scaler.fit\_transform(images)

## La fonction d’erreur : compile

Il faut définir une erreur pour que le modèle converge vers une bonne solution. Pour cela on utilise « sparse\_categorical\_crossetropy ». Son but est de minimiser l’erreur (descente de gradiant).

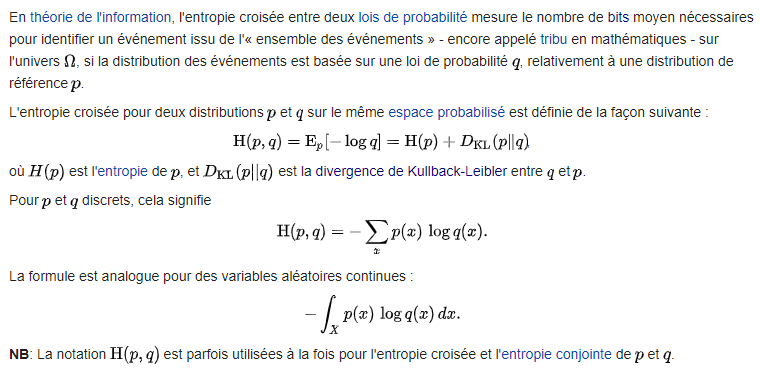
Pour cela on utilise le « log » de la probabilité

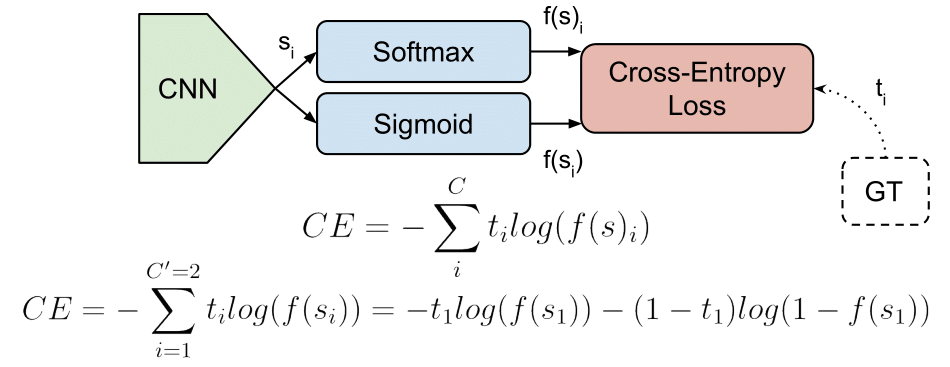


On voit que plus on se rapproche de 1, plus la valeur est haute. On va donc faire la moyenne des « logs » des probabilités filtrées. Puis l’inverse de cette moyenne.

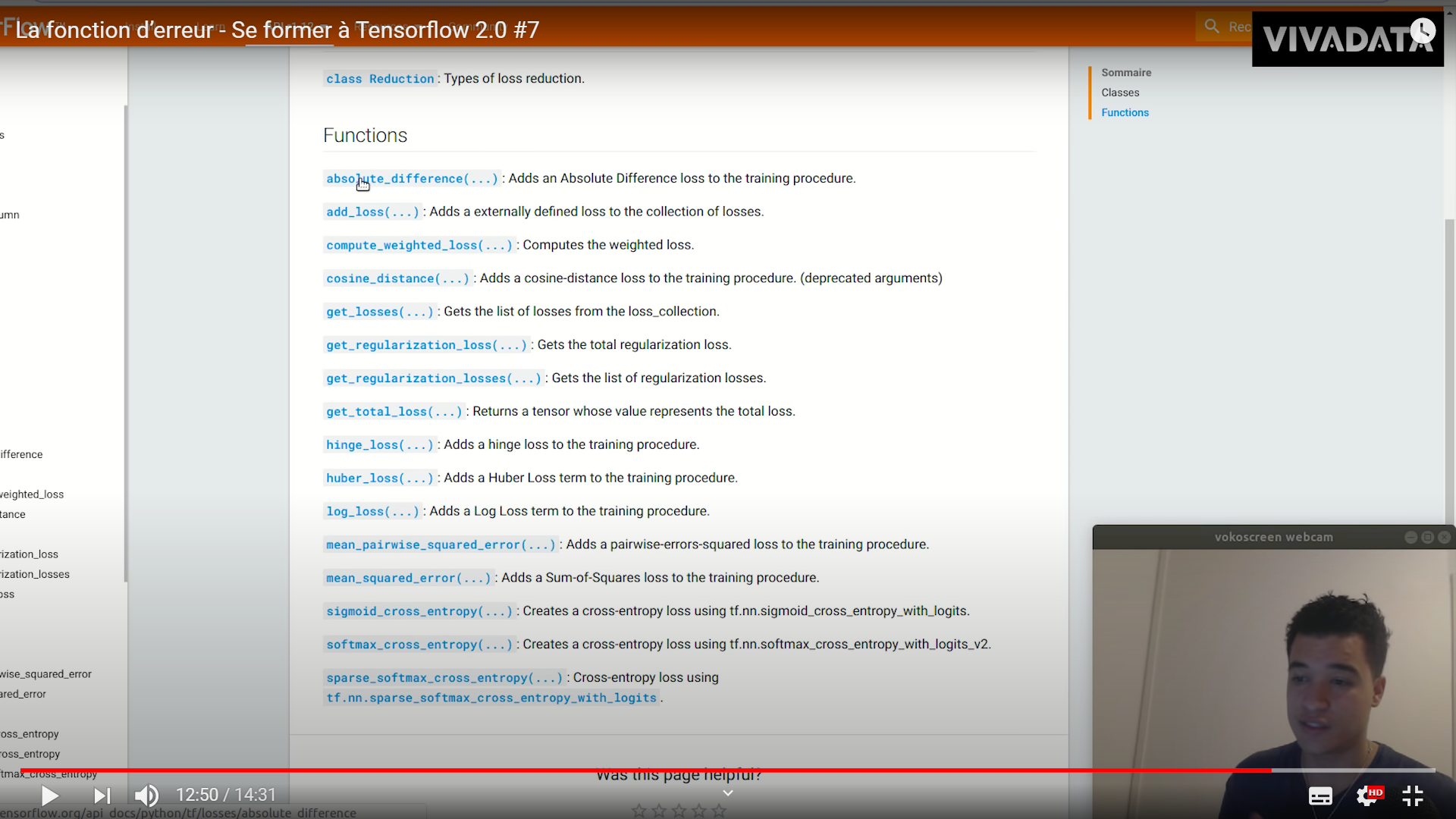
Puis l’inverse de la moyenne diminue, plus le résultat est correct.

**Pour info le calcul de la cross entropie :**





Il existe plusieurs formes de « categorical\_crossetropy » :



## Jeu d’entrainement, jeu de validation et jeu de tests

Il y a 3 sortes de données

* Les données d’entrainement : ces données savent de quelles classes sont les images, et connaissent leurs taux d’erreur. Le modelé va s’entrainer sur ces données pour voir si elle sont correctes et si notre modèle performe sur les données de validation. On vérifie si ce qui a été appris peut se généraliser correctement sur les données de validation.
* Les données de validation : le but est de vérifier, pendant l’entrainement, le modèle pour qu’il soit capable de généraliser ce qu’il a appris sur ‘l’entrainement.
* Les données de jeu : le but est de ne pas faire en sorte d’avoir de meilleurs résultats sur les données de test. Il ne faut pas optimiser à 100 % les données de test sinon il est potentiellement possible qu’il se trompe plus souvent.

<https://www.youtube.com/watch?v=ZpMV3cBCH3Y>

<https://www.youtube.com/watch?v=BtAVBeLuigI>

Cycle de travail :

1. Récupération des données

2. Nettoyage des données

3. Exploration des données

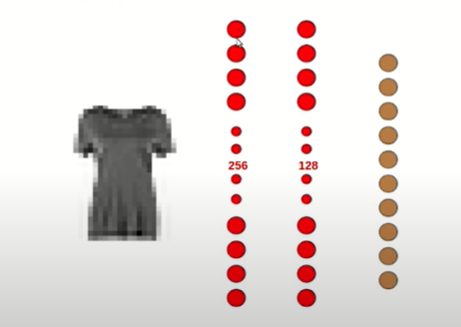
4. Créer un modèle de statistique

5. évaluation de la qualité du modèle

6. Déploiement des algorithmes



Nous on fait une classification



1. Installer une VM Debian 9

Installation d’une VM vierge Debian 9 sur VMWARE

Le login administrateur sera « root » et le mot de passe sera « root »

Le login utilisateur sera « user » et le mot de passe sera « user »

1. Installer « vmware tools » sur Debian 9

Installation de « vmware tools » : <https://www.youtube.com/watch?v=-BcO90Byf18>

1. Installer JUPYTER sur DEBIAN 9 (facultatif)

<https://hostadvice.com/how-to/how-to-install-jupyter-on-debian-9/>

## Mettre à jour les paquets

sudo apt-get update

## Installer Python 3 et pip

$ sudo apt install python3-pip python3-dev

## Créer un environnement virtuel

### Installer virtualenv

$ sudo -H pip3 install virtualenv

### Créer un répertoire appelé machinelearning

$ mkdir ~/machinelearning

$cd ~/machinelearning

Il s'agit du répertoire où tous les fichiers du projet seront conservés. Dans ce répertoire, exécutez la commande ci-dessous pour créer l'environnement virtuel Python:

$ virtualenv machinelearning

Cela entraînera la création d'un nouveau répertoire ; machinelearning dans le répertoire machinelearning. Ensuite, dans ce répertoire, pip et Python seront installés dans leurs versions locales. Vous pouvez maintenant l'utiliser pour configurer un environnement Python distinct pour l'application Jupyter.

Avant d'installer Jupyter, exécutez la commande ci-dessous pour se connecter a l’environnement virtuel Python

user@machinelearning:~$ source machinelearning/machinelearning/bin/activate

Pour quitter ensuite l'environnement virtualenv :

$deactivate

## Installer et utiliser JUPYTER

$ pip3 install jupyter

Cela imprimera une liste des activités de Jupyter Notebook

$ jupyter notebook

## Installer et utiliser Vscode

<https://linuxize.com/post/how-to-install-visual-studio-code-on-debian-9/>

Si probleme installation de tensoflow

sudo pip3 install -U pip

1. Installer les dépendances python et les packages utilisés

pip install opencv-python

Ajouter dans dockerfiles

sudo apt-get install opencv

(machinelearning) user@machinelearning:~$ pip3 install matplotlib==3.0.3

(machinelearning) user@machinelearning:~$ pip3 install tensorflow==2.0a0

(machinelearning) user@machinelearning:~$ pip3 install numpy==1.17.0

(machinelearning) user@machinelearning:~$ sudo -H pip3 install pandas

user@machinelearning:~$ pip3 install sklearn

user@machinelearning:~$ pip3 install opencv-python

user@machinelearning:~$ pip3 install tqdm

Keras-Applications (1.0.8)

Keras-Preprocessing (1.1.2)

<https://stackoverflow.com/questions/57507832/unable-to-allocate-array-with-shape-and-data-type>

si pas assez de mémoire sur la machine

en root :

echo 1 > /proc/sys/vm/overcommit\_memory

# Sources

<https://www.tensorflow.org/tutorials>

## Machine learning générale

<https://openclassrooms.com/fr/courses/6417031-objectif-ia-initiez-vous-a-lintelligence-artificielle>

<https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning>

<https://openclassrooms.com/fr/courses/4297211-evaluez-les-performances-dun-modele-de-machine-learning>

## Faire du deep learning

<https://www.youtube.com/watch?v=wQ8BIBpya2k&list=PLQVvvaa0QuDfhTox0AjmQ6tvTgMBZBEXN><https://www.youtube.com/watch?v=wQ8BIBpya2k>

<https://www.youtube.com/watch?v=0LwswxqacTs>

<https://nokomprendo.frama.io/tuto_fonctionnel/posts/tuto_fonctionnel_28/2018-10-16-README.html>

## Cours de deep learning

<https://www.youtube.com/watch?v=YOlOLxrMUOw&t=301s>

<https://www.youtube.com/watch?v=wQ8BIBpya2k&list=PLQVvvaa0QuDfhTox0AjmQ6tvTgMBZBEXN>

1. Cours Pratique

<https://www.youtube.com/watch?v=hP7Ac8S9Tgs>

<https://pythonprogramming.net/loading-custom-data-deep-learning-python-tensorflow-keras/>

1. Glossaire

|  |  |
| --- | --- |
| JUPYTER | Jupyter Notebook est une application web robuste qui vous permet de créer et de distribuer des documents contenant des équations, du code en direct, du texte narratif et des visualisations |
| Perceptron | C’est la modélisation d’un neurone qui va nous permettre de faire une prédiction en sortie. Il prend X entrées pour produire une sortie. Chaque entrée à un poids ce qui différencie l’importance des données.  Chaque entrée est multipliée par son poids. Pour une image il y a autant d’entrées qu’il y a de pixels dans l’image. |
| La descente de gradient | Le but de la descente de gradient est de minimiser l’erreur du modèle. Si on réduit l’erreur, on va de plus en plus proche du résultat attendu. Pour réduire l’erreur il faut changer les poids. |
| TensorBoard | c’est une application pratique qui vous permet de visualiser des aspects de votre modèle dans votre navigateur |
|  |  |

1. Autres

La technique du « MILPA »

[https://www.miimosa.com/fr/projects/une-agriculture-durable-geree-par-l-ia#description](https://www.miimosa.com/fr/projects/une-agriculture-durable-geree-par-l-ia" \l "description)





<https://www.debroussaillez.fr/ameliorer-le-rendement-avec-la-technique-des-3-soeurs/>

sudo apt-get install docker docker-compose

ouvrir un fichier Dockerfile et ajouter

FROM lbelmarletelier/debian9

#importer une image de base

WORKDIR /code

COPY requirements.txt requirements.txt

RUN sudo apt-get install -y opencv python3-pip python3-dev

# installer opencv et python3

RUN sudo pip3 install -r requirements.txt

# installer les dependances python3 de la liste du fichier requirements.txt

EXPOSE 8888

# port de jupyter par defaut

mettre le fichier requirement,txt au meme endroit que Dockerfile