Majeure Machine Learning

Deep learning
Méthodologie

Contenu

- Augmenter le jeu de données
- Design
- Débugger
- Mettre en production un modèle
- Faire de la veille

Ce que vous devrez savoir faire

- Comprendre le principe et l'utilité de la Data Augmentation
- Savoir quoi faire pour designer un réseau de neurones
- Savoir comment débugger l'apprentissage d'un réseau de neurones
- Avoir une idée des techniques actuelles pour mettre en production un modèle
- Savoir qui suivre et où faire de la veille

Gérer le jeu de données

Répartition Train / Test

Objectif : Détection de visage

Données en provenance :

-USA
-Inde
-France
-Allemagne
-Russie
-Serbie
-Australie

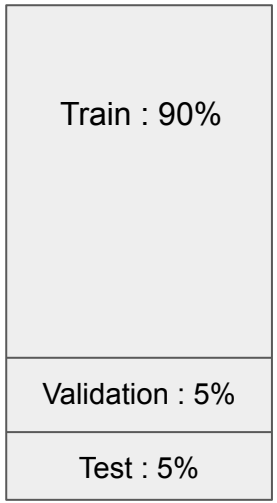
Sexe :

-Homme
-Femme

A ne surtout pas faire !

Découpage

1.



2.

Pour i allant de 1 à *nb_epochs*
    Pour j allant de 1 à *train_size*/*batch_size*
        Tirer *batch_size* exemples parmi ceux du TrainSet non tirés durant cet epoch
        Faire la ForwardPropagagtion pour ce mini-batch
        Calculer la Loss
        Mettre à jour les poids

    Calculer l'accuracy sur le TrainSet
    Calculer l'accuracy sur le ValidationSet
    Afficher les résultats

Calculer l'accuracy sur le TestSet
Afficher

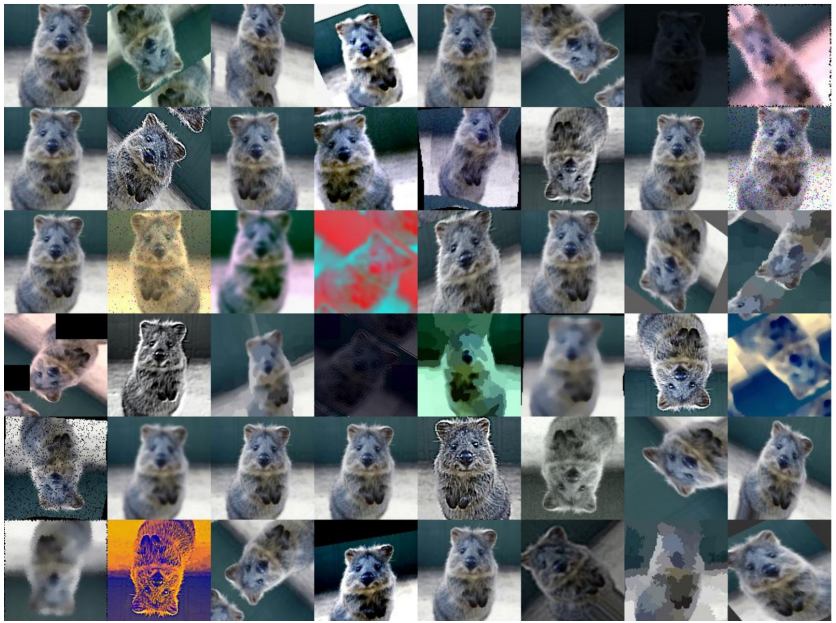
Découpage

```
n_epochs = 100
batch_size = 32

for i in range(n_epoch):
    for j in range(len(X_train)/batch_size):
        X, Y = get_minibatch(X_train, Y_train)
        Y_pred = NN.predict(X)
        loss = categorical_crossentropy(Y, Y_pred)
        NN.optimize(loss)

    Y_train_pred = NN.predict(X_train)
    Y_val_pred = NN.predict(X_val)
    print("Train Accuracy : " + str(accuracy_score(Y_train, Y_train_pred))
    print("Validation Accuracy : " + str(accuracy_score(Y_val, Y_val_pred))

Y_test_pred = NN.predict(X_test)
print("Test Accuracy : " + str(accuracy_score(Y_test, Y_test_pred))
```

Data Augmentation

Problématique : Pas assez de données pour entraîner une réseau de neurones profond

Principe : Augmenter le nombre d'exemples de données en générant de nouvelles données à partir des existantes modifiées



+ rend le réseau plus robuste !

Gestion d’un projet de Deep Learning

Précision vs Performance

Classifier	Accuracy	Learning time (j)	Running time (ms)
A	90%	1	80
B	**95%**	**1,5**	**95**
C	99%	3	1 500
D	**99,5%**	**10**	**3 000**

Performance de l’homme vs modèle

Tant que notre modèle est moins précis que l’homme, on peut :

- Observer les données mal prédites et essayer d’identifier la cause

- Labeliser (plus) des données humainement

- Analyser les biais / overfitting

Dès lors que notre modèle est plus précis que l’homme, on peut :

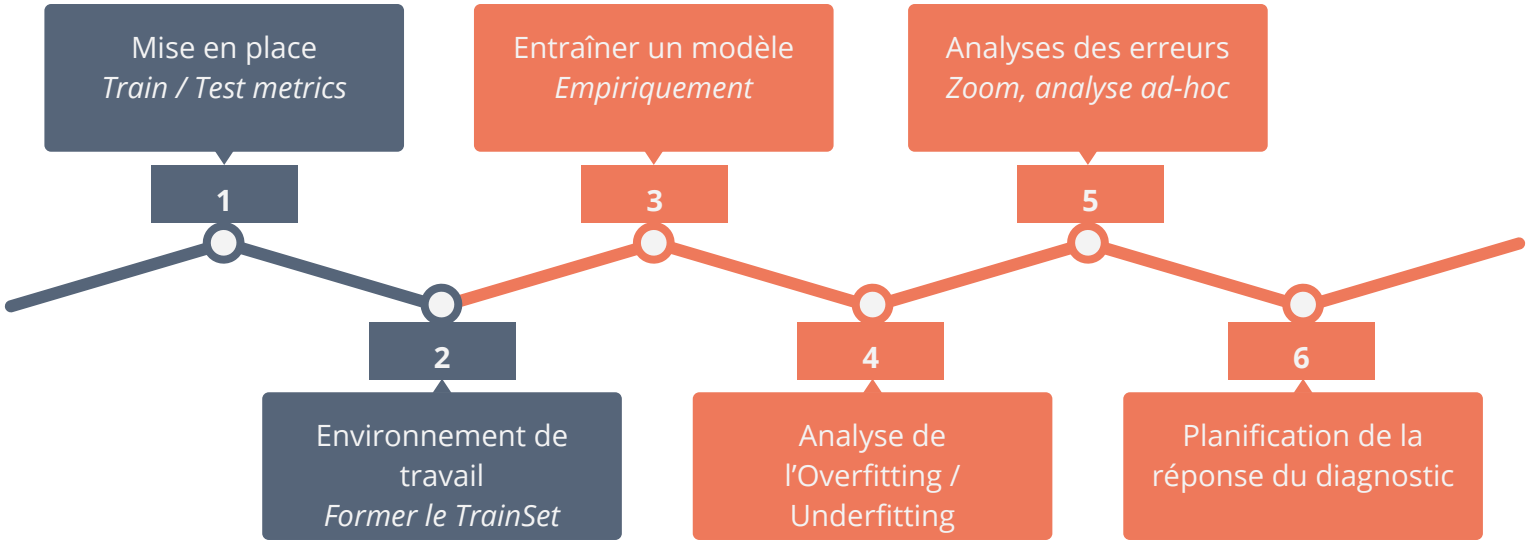
- Identifier si le modèle est aussi précis que le meilleur des hommes

- Estimer si le modèle peut réellement s’améliorer

- Ajouter de nouvelles données qualifiées

- Créer des modèles plus complexes

Méthode projet



Démarrage

Cycle projet : itérations

Approche itérative

Itération 1 :

- Mise en place d'un premier réseau avec un peu de Dropout et de régularisation.
- Utiliser Adam ou SGD + momentum + learning rate decay
- Mettre en place du Early stopping
- Utiliser du Transfer Learning si l'opportunité se présente

Itérations suivantes :

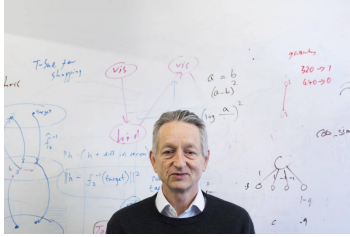
- Si Underfitting : Augmenter la capacité (profondeur / densité)
- Si Overfitting : Augmenter le Dropout / Régularisation, réduire la capacité
- Si problèmes d'optimisation : Surveiller les valeurs des gradients, réduire le learning rate, autres (gradient clipping, batch normalization)...

Designer son réseau de neurones

Le choix de l'architecture

Types de données	Architectures à privilégier
Vecteur de taille fixe	FeedForward
Structure topologique	Convolution
Séquence	Récurrence

Densité (Geoff Hinton)



3 hypothèses :

- Le nombre de neurones cachés doit être compris entre la taille de la couche d'entrée et la taille de la couche de sortie.
- Le nombre de neurones cachés doit être 2/3 de la couche d'entrée, plus celle de sortie.
- Le nombre de neurones cachés doit être inférieur à deux fois la taille de la couche d'entrée.

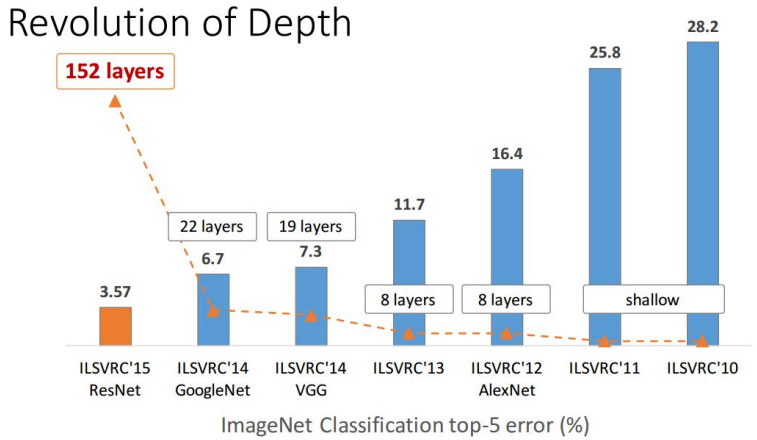
Profondeur

Nombre de couches :

0 - Uniquement capable de représenter des fonctions ou des décisions séparables linéaires.

1 - Peut théoriquement approximer n'importe quelle fonction si possède suffisamment de neurones et si l'algorithme d'optimisation le permet

>1 - Peut théoriquement approximer n'importe quelle fonction (comme 1) mais avec moins de neurones. Fait également l'hypothèse que le problème peut être découpé en sous problèmes.

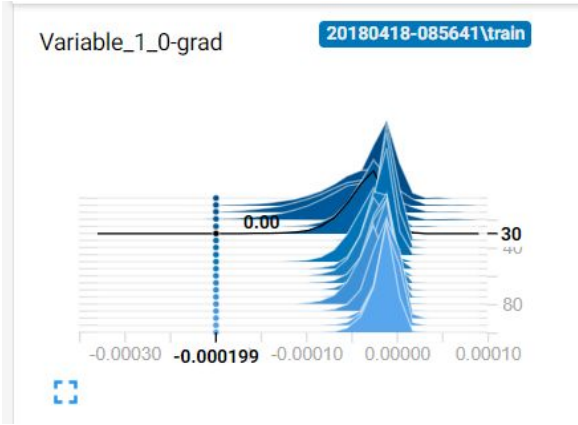


Débuggage

Métriques

Afficher un maximum d'informations !

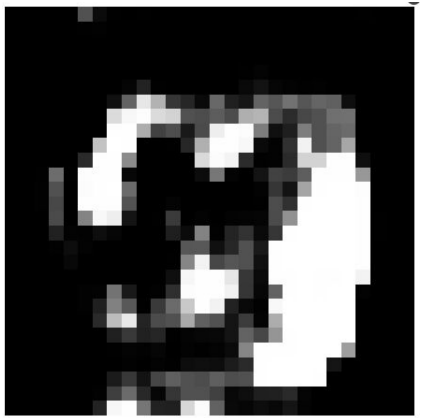
Loss



Gradients

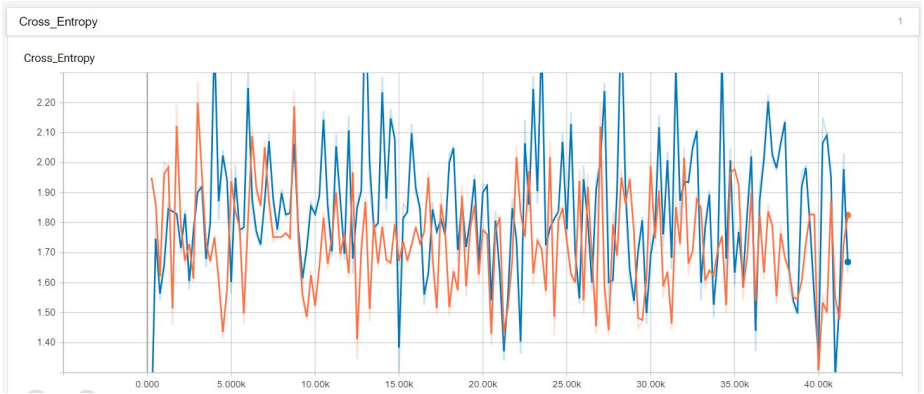
Feature Maps

Prédictions
(et pires erreurs)



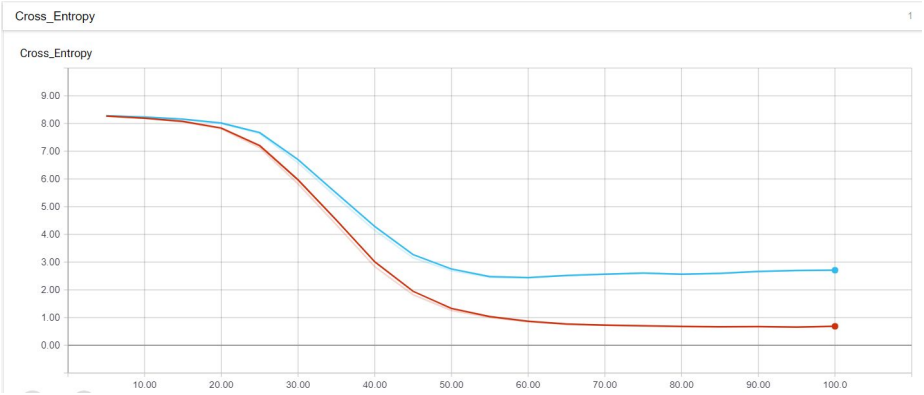
Valider son implémentation

Mon réseau de neurones n'apprend pas !!



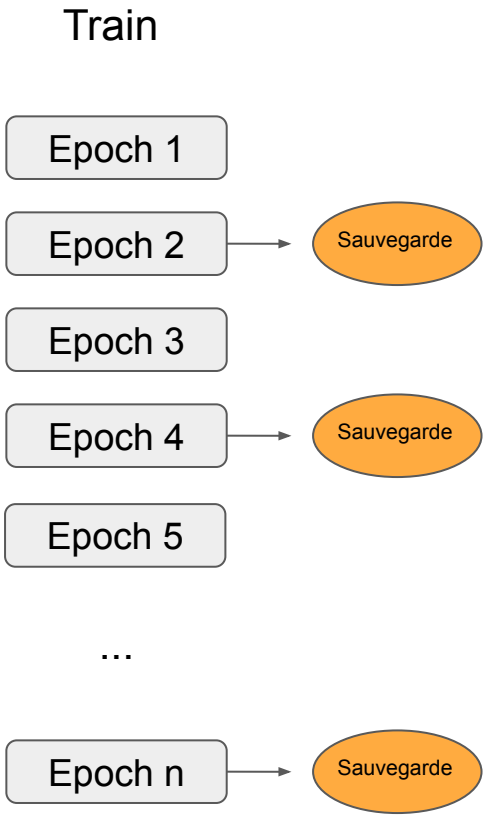
Si Backpropagation custom : Gradient Checking

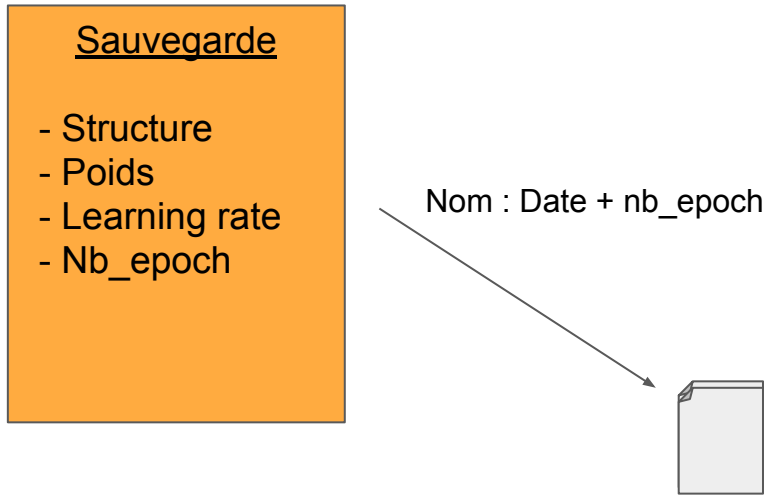
Réduire son jeu de données à quelques exemples pour valider l'overfitting (et donc l'apprentissage !)



Mise en production

Sauvegarder un modèle

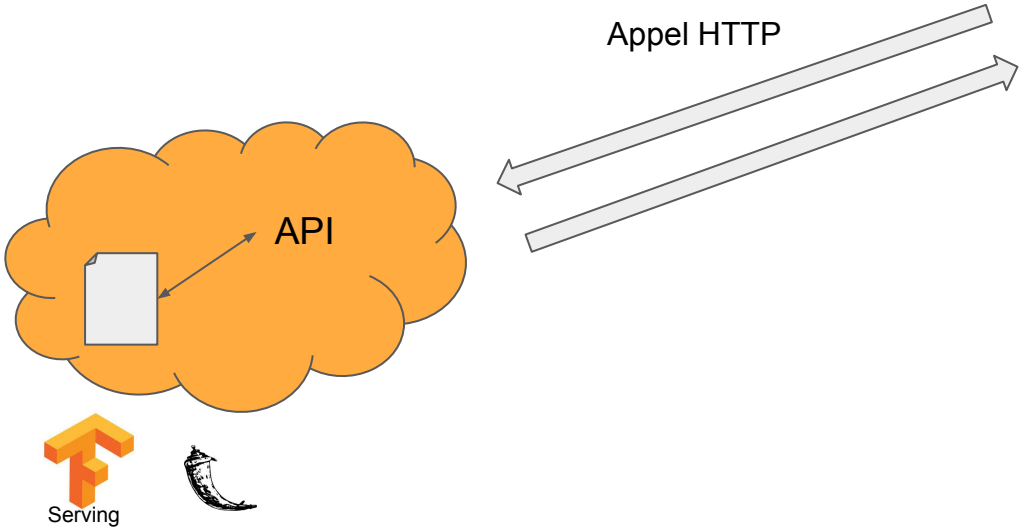




Intérêts :

- Avoir plusieurs versions pour sélectionner la meilleure
- Pouvoir reprendre l'apprentissage à partir d'une des sauvegardes

Mise en place d'un API



<u>Avantages :</u>

- Appels possibles depuis n'importe qu'elle solution (logiciel, site web, embarqué…)
- Peu importe le langage de développement
- Gestion du load balancing que sur l'API

Faire de la veille sur le Deep Learning

Nos références

Twitter

- Andrew Ng (@*AndrewYNg*)
- Yann Lecun (@*ylecun*)
- Ian Goodfellow (@*goodfellow_ian*)
- François Chollet (@*fchollet*)
- Arthur Juliani (RL) (@*awjuliani*)

Sites / Blogs

- ActuIA (actuia.com)
- AI Weekly (http://aiweekly.co/)
- Data Science Weekly (datascienceweekly.org)
- KDnuggets (kdnuggets.com)
- ML Mastery (machinelearningmastery.com)

Autre : arXiv.org

Fin du chapitre 5.4