Deep Learning

Soutenance MOOC4

Thomas KOWALSKI

Le deep learning : pourquoi ?

- Un sujet “à la mode”
  - En informatique
  - Dans l’industrie
  - Dans la société
- Envie de découvrir ce que c’est vraiment
- Et d’appliquer les techniques à de vrais problèmes

Présentation du cours

- Cours de *deeplearning.ai* proposé sur Coursera
- Plusieurs chapitres :
  - **Neural networks & Machine Learning :** Présentation des concepts mathématiques abordés, exemples
  - **Hyperparameter tuning, Regularization and Optimization :** Présentation des algorithmes utilisées dans les vraies applications, techniques d'optimisation et de régularisation
  - **Convolutional Neural Networks :** Présentation et application des réseaux convolutifs au problème de *computer vision*
  - **Sequence Models :** Présentation et application de techniques pour les modèles séquentiels

- Objectif pour moi : les trois premiers mais, surtout, **comprendre**

Qu’est ce qu’un neurone ?

- Une application (généralement linéaire)
- Une fonction d’activation (pas linéaire : sigmoïde, *tan*, ReLU…)

$$A = \sigma(W_i \cdot X + b)$$

Qu'est ce qu'un réseau ?

- Une couche = plusieurs neurones prenant le même vecteur d'entrée
- Un réseau = plusieurs couches de suite

$$X_n = \sigma(W_n X_{n-1} + b_n)$$

$$X_{n-1} = \sigma(W_{n-1} X_{n-2} + b_{n-1})$$

$$\dots$$

$$X_0 = \text{Input}$$

Qu'est ce que l'apprentissage ?

- Pour obtenir le résultat, on effectue la *forward propagation*
- Donne la sortie du réseau (image de l'entrée par la fonction représentée)
- **Objectif ?** utiliser la fonction pour prédire la sortie quand on ne connaît que l'entrée
- **Comment faire ?** En apprenant grâce à des exemples "étiquetés" (dont on connaît la sortie)
- **Comment apprendre ?** Grâce à la fonction de coût (on la minimise)

$$\mathcal{J}((x_i)_{1\leq i\leq n}) = \frac{1}{n}\sum_{i=1}^{n} C(\hat{y}_i, y)$$

Comment minimiser le coût ?

- On souhaite apprendre des paramètres (x_i)
- On connaît la valeur de la fonction à minimiser avec ces (x_i)
- On peut calculer la dérivée de la fonction à minimiser

On utilise la méthode de descente de gradient

Puisqu'on a :

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y}\frac{\partial y}{\partial x}$$

On utilise :

$$x_{i+1} = x_i - \alpha \times \frac{\partial \mathcal{J}}{\partial x_i}$$

Conclusion sur l’apprentissage

- On a une fonction de prédiction qui dépend de paramètres
- On sait “apprendre” les valeurs optimales de ces paramètres si on a des exemples
- ⇒ On peut prédire la valeur de la fonction pour des entrées inconnues !

Réseaux convolutifs : origine

- Grande quantité de données ⇒ Problème de “grande dimension”
- Grande dimension ⇒ Beaucoup de neurones / de couches
- Beaucoup de neurones / de couches ⇒ Entraînement long
- Entraînement long ⇒ Coût monétaire élevé, pas pratique…

Alors que faire ?

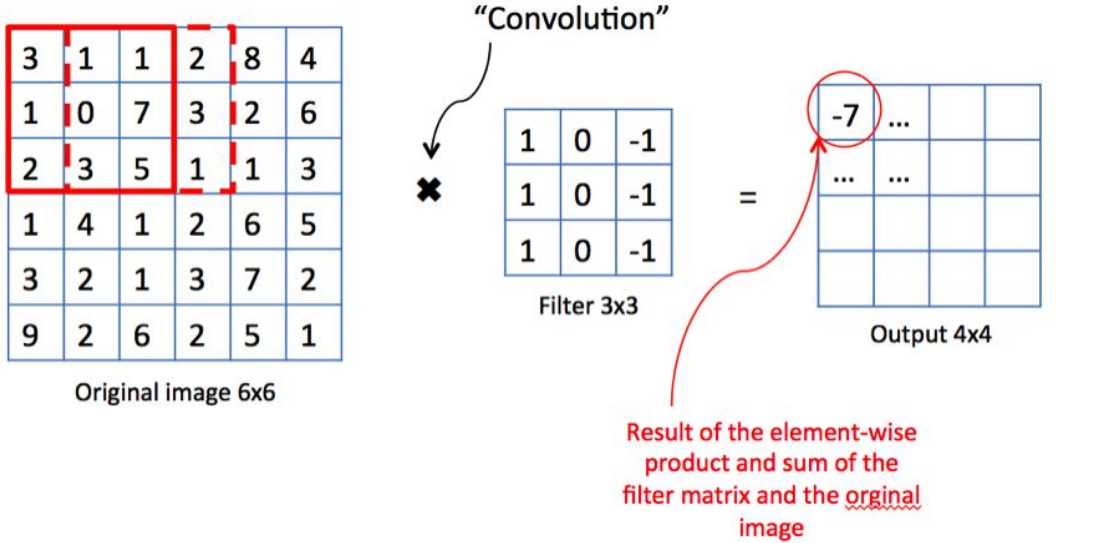
Réseaux convolutifs : idée

Plutôt que de faire un réseau “dense” (ou *fully connected*) (opération linéaire avec beaucoup de neurones tous reliés les uns aux autres) on utilise la

Convolution

Réseaux convolutifs : fonctionnement

Pour chaque couche convolutive, des neurones correspondant à des *filtres*



Plus de détails (padding, strides, dropout, pooling) dans le rapport

Réseaux convolutifs : résultats

- Beaucoup moins de paramètres à apprendre (taille du filtre et taille de l'images pas forcément corrélées)
- Permet la détection de *features* ("traits" de l'image)
- Très efficace en *computer vision*
- Autres possibilités : utiliser des convolutions 1D ou 3D (pas abordé dans le cours)

Computer vision : identification

- Réseaux convolutifs efficace pour classifier une image parmi des types d'objets
- Technique *a priori* limitée dès lors qu'on s'attaque à une image qui ne représente pas qu'un objet

Comment faire pour identifier des éléments (visages, objets, etc) ?

Computer vision : localisation

- En plus de la classification, il est utile de pouvoir localiser l'objet
- Pour cela on utilise l'algorithme d'identification dans une fenêtre
- Et on utilise des fenêtres placées à différents endroits / de différentes tailles

Autre possibilité : *YOLO*

Conclusion sur le cours

- Cours très complet
  - Pose les bases
  - Apporte des informations précieuses sur la pratique
  - Donne un premier aperçu d'une discipline moderne, le *computer vision*
- Mais très dense
  - Les TP sont longs et pas forcément faciles
  - Les cours également, et ils demandent parfois d'aller consulter d'autres sources pour une compréhension complète
- Avec quelques défauts
  - Les TP sont souvent un peu trop guidés, ce qui nous rend incapables de faire quelque chose de nous-mêmes par la suite

Projet : introduction

- Le cours ne propose aucun projet (en dehors des TP)
- Mon objectif :
  - Faire quelque chose de concret
  - Avoir un objectif raisonnable (pour être sûr d'y arriver)
  - Consolider les bases que j'ai eues avec le cours
  - Faire quelque chose qui marche en direct

Projet : présentation

Idée : un classifieur de caractères

Idée pour le futur : un réseau capable de reconnaître une expression LaTeX

Idée plus raisonnable maintenant : un réseau capable de reconnaître des caractères

Idée retenue : un réseau capable de reconnaître des nombres manuscrits

Projet : pourquoi ?

- Caractères manuscrits
  - Problème “connu”
  - Beaucoup de datasets disponibles sur Internet
  - Testable facilement avec d’autres données (caractères écrits par moi-même)
- Problème relativement simple
  - Noir et blanc
  - Images peu complexes (pas d’arrière plan / avant plan)

Projet : démarche

- Articles citant la possibilité d'utiliser un réseau classique (non convolutif)
- Mon objectif : utiliser un réseau convolutif (pour mieux maîtriser le concept)
- Comment faire ?
  - Au hasard
  - Sans chercher des réseaux qui existent déjà (sinon aucun intérêt)

Projet : préparation des données

- Utilisation de github.com/kensanata/numbers
- Milliers d'images de différentes personnes
  - Format : <Identifiant_Personne>/<Chiffre>/<nom_fichier>.png
  - On a plein d'exemples avec les étiquettes données (<Chiffre>)
  - Et de plein de sources différentes (limite l'overfitting)

Projet : préparation des données

En Python :

- Déterminer la liste des fichiers
- Pour chaque fichier :
  - Charger le fichier en mémoire
  - Le passer en noir et blanc avec scipy
  - Appliquer un filtre (pour saturer l'image et enlever les artefacts dûs à la compression)
- Stocker les données dans une structure de données Python
- Enregistrer les données avec *pickle*

Projet : modèle

- Plusieurs tentatives
- Problème : rester simple
  - Pas d’espace de calcul en ligne
  - Pas de carte graphique avec CUDA
  - Pas d’AVX sur mon binaire de TensorFlow
- Tout en restant précis

Projet : modèle

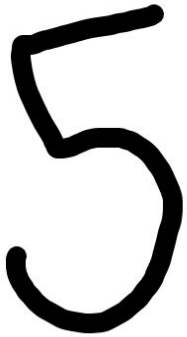
- Premier modèle trop complexe (beaucoup de Conv2D) : dix heures d'entraînement
- Second modèle trop simple (Conv2D * 32 => Out) : très rapide mais inutile

Conv2D 32 → Conv2D 16 → Conv2D 8 → Conv2D 4 → Out

~ Une heure d'entraînement

~ 98% de réussite sur le jeu de test

Projet : test avec d'autres données



- Très bon résultat sur le jeu de test
- Mais sur de nouvelles données ?
  - Création de nouvelles images (avec drawisland.com)
  - Passage dans le réseau
  - Toujours le bon résultat (sur 30 essais)
    (Trois personnes différentes)

Résultats très satisfaisants !

Même avec des données "limite" -->

Projet : conclusion

Mon projet :

- Un problème relativement simple
- Mais une solution très satisfaisante pour un débutant
- Avec en plus de bonnes performances sur de nouvelles données

Possibles autres travaux :

- Tenter avec les données de MNIST Digits
- Essayer d'effectuer des rotations sur les chiffres et voir jusqu'à quand les prédictions restent bonnes
- Tenter de reconnaître un nombre (code postal, téléphone...) avec une sliding window

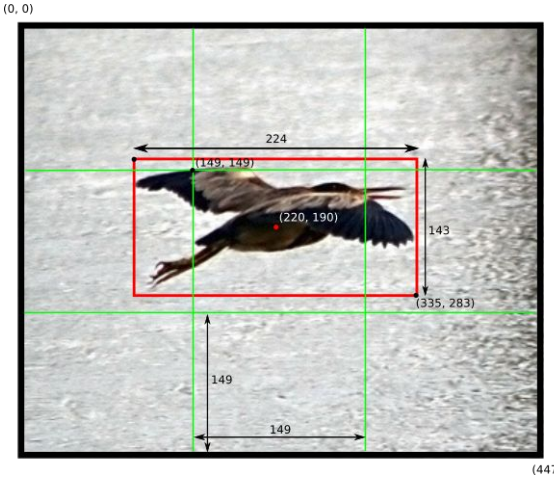
MOOC : Conclusion

- Objectif initial : en apprendre plus sur ce qu'est le deep learning
  - Ce que c'est
  - Comment ça marche
  - Ce que ça permet (ou pas) de faire
- Mon ressenti :
  - Cours de très bonne qualité (surtout comparé à MOOC3)
  - Aucun regret sur le choix du sujet du MOOC
  - Possibilité d'utiliser le deep learning dans des projets à venir sans être perdu
  - Même si l'expérience semble être un facteur indispensable pour la réussite dans ce genre de projets

Questions

Computer vision : algorithme YOLO

- Faire beaucoup de fenêtres coûte cher (= lent)
- Idée de l'algorithme YOLO : *You Only Look Once*
- Au lieu d'utiliser une fenêtre de détection :
  - Découper l'image en sous-images
  - Chaque carré peut prédire un objet (à savoir la taille de la *bounding box* par rapport à l'image et ses coordonnées dans le carré)
  - On ne garde qu'une prédiction par objet :
    - On applique la *non-max suppression*
- L'entraînement se fait avec la métrique Proba(objet dans carré) * IoU (objet réel)



x = (220-149) / 149 = 0.48
y = (190-149) / 149 = 0.28
w = 224 / 448 = 0.50
h = 143 / 448 = 0.32