Deep learning pour la Data Visualisation

Étudiants :
Renaud D'harréville et Théo Jaunet

Encadrants :
Romain Vuillemot et Aurélien Tabard Date :

11/01/2017

Table des matières

1 Semaine du 9 au 13 janvier

En début de semaine, nous avions mal interprété le sujet, nous étions donc partis dans une mauvaise direction. Cet incident n'a pas eu beaucoup de conséquence puisque le début été surtout de la recherche sur le deep learning. Nous avons cependant fais du travail peu utile a présent , notamment une réflexion sur l'architecture possible de la solution applicative.

Nous avons regarder la vidéo suivante : `https://vimeo.com/180044029`. Cette vidéo nous a permis d'avoir un bref aperçu de ce qu'est le Deep Learning. Les points clés à retenir de cette vidéo sont la manière de créer un réseaux de neurones, couches après couches. Ces couches permettent entre autres un niveau d'abstraction, de plus en plus précis, des données mises en entrées du réseaux de neurones. Nous avons aussi retenu une méthode permettant de compresser les données sur un nombre restreints de neurones en sortie. Cette méthode permet certes la compression de données, ce qui est déjà très intéressant en soit, mais également, à l'aide du réseau de neurones, de pouvoir retrouver les motifs primaires caractérisants les images données en entrée, d'un autre point de vu que celui humain (couleur,hauteur par exemple). Notre but,dans un premier temps, est donc d'appliquer un algorithme de deep learning sur des images de visualisation de données afin d'y extraire des motifs primaires.

Nous avons également appris d'avantages sur les réseaux neuronaux grâce au livre communautaire de Gene Kogan, plus précisément le chapitre "Looking inside neural nets" présent ici : `https://ml4a.github.io/ml4a/looking_inside_neural_nets/`

A la suite de notre réunion avec Romain Vuillemot le jeudi 11 janvier, plusieurs points sont a noter.

- Premièrement, la décomposition d'image en quelques noeuds contraints,en effet comme vu plus haut, le deep learning permet de réduire le nombre de noeuds en entrée afin de concentrer sur des éléments clés. Cela sera notre priorité de recherche. Pour effectuer cette recherche nous allons, pour l'instant, nous concentrer sur l'application du deep learning avec des images de diagrammes en barres.
- Ensuite, une fois l'algorithme entraîné, la prochaine étape est de le tester avec des données contrôlées. En effet , jusqu'à présent, l'algorithme utilisait des visualisations quelconques. Nous devons donc le tester avec un diagramme en barres généré avec des données que nous pouvons modifier afin d'observer l'impact de ce changement.
- Une fois les étapes précédentes effectuées nous pouvons étendre l'application du deep learning a plusieurs visualisations différentes. Pour cette étapes , nous avons déjà choisis plusieurs types de visualisation afin de couvrir un grand nombre de type de données différentes {nominales, ordinales ,quantitatives}.
- Le dernier point de cette réunion fut le transfert de style pour le passage d'une visualisation à une autre.

2 semaine du 16 au 20 janvier

Lors de cette semaine, nous nous sommes attelés à la rédaction du cahier des charges. Le lundi et le mardi nous avons donc commencer à le rédiger en grande partie. Le mercredi nous avons eu une petite réunion avec Romain Vuillemot afin de clarifier certains points du premier rendu. La suite de la semaine nous avons donc fini le cahier des charges.

3 semaine du 23 au 27 janvier

4 semaine du 30 janvier au 3 février

Nous avons commencé par écrire un planning prévisionnel, disponible sur github. Voici donc les étapes détaillées que nous avons pu faire pendant la semaine :

4.1 Recherche d'un jeu de donnée d'images bar chart

Temps estimé : 4h Temps effectif : environ 2h Pour commencer, nous avons voulu savoir s'il était possible de télécharger les images d'une recherche google. Cela est possible, mais malheureusement, la grande majeur partie des images (de notre recherche), ont des restrictions de droit d'auteur. Nous n'avons ainsi pu récolter que deux graphiques intéressants... La méthode utilisée pour télécharger rapidement les images s'applique à firefox (cela doit certainement être possible avec chrome aussi) est la suivante : clique droit sur la page -> informations sur la page -> médias -> sélection des images voulues -> enregistrer sous. Nous sommes donc allé sur le site BlockBuilder, que nous avait présenté R. Vuillemot, pour recommencer ce test. Cela a marché, et nous avons donc chargé un grand nombre d'images (environ 1200) que nous avons téléchargé.

Pour plus de simplicité, nous avons créé un répertoire dropbox dans lequel nous avons mis ces images, afin de les trier. Nous en avons fait 3 sous dossiers : Bar chart, à supprimer, et autres. Au total, 423 visualisations de bar chart ont été récupérées, environ 150 visualisations ont été supprimées (principalement des images blanches, ou ne contenant que du texte, mais aussi des graphiques très peu intéressants). Il nous reste donc 610 autres visualisations à notre disposition.

4.2 Installation et prise en main de TensorFlow

Temps estimé : 4h Temps effectif : 2h

Pour cette partie, nous avons suivi ce tutoriel : `https://www.tensorflow.org/get_started/os_setup#optional_install_cuda_gpus_on_linux`

4.2.1 Python et TensorFlow

Pour toutes les parties de deep learning, nous avons choisi, avec l'aide de notre encadrant d'utiliser la librairie open source TensorFlow `https://www.tensorflow.org/`. Cet outil est très utilisé et possède une grande communauté, ce qui devrait faciliter nos recherches pour l'utilisation.

Nous avons préféré utilisé la version 3.5 de python pour ce projet, et avons donc correctement mis à jour ce paquet. Il a fallu en même temps installer et mettre à jour pip3. Ceci afin de pouvoir installer la librairie TensorFlow (pip3 install tensorflow).

4.2.2 TensorFlow-gpu et CUDA

TensorFlow existe aussi dans une version plus optimisée utilisant les ressources beaucoup plus intéressantes pour ce domaine (domaine des matrices, et du calcul graphique) du GPU. Mais cela nécessite l'installation d'un autre outil, lui protégé, CUDA, proposé par Nvidia. Pour notre part, nous avons tous les deux une carte graphique Nvidia GEFORCE, et avons donc installé la version de CUDA Toolkit correspondante `https://developer.nvidia.com/cuda-gpus`. Il a fallu aussi installer CuDNN.

Pour l'installation de CUDA, il faut d'abord télécharger la dernière version disponible (8.0 dans notre cas), et en faisant cela, suivre les instructions proposées lors du choix du paquet à installer (il faut d'abord faire tous les choix pour qu'apparaisse les instructions). Pour notre part, nous étions sous Ubuntu 16.04, et avons téléchargés le deb local. Une fois cela fait, il nous a suffit de poursuivre le tutoriel de TensorFlow, nous expliquant la procédure pour installer CuDNN. Attention, il ne faut pas oublier de télécharger les mises à dépendances (sudo apt-get install libcupti-dev).

4.3 Premier essai sur un test exemple

Pour ce premier essai, nous avons continuer à suivre les instructions du tutoriel, et avons donc testé le fichier 'convolutional.py' donné lors de l'installation de TensorFlow. Celui-ci nous a causé de nombreux soucis : - Nous avons eu des soucis liés à python, qui n'arrivait pas à importer les librairies. Pip3 n'était pas bien mis à jour, il a fallu refaire en partie l'installation de TensorFlow. - Une autre erreur que nous n'avons pas réussi à comprendre, était un souci d'extraction des images MNIST des librairies téléchargées (directement depuis le fichier convolutional.py), qui n'était pas sous la bonne forme. Ce que nous avons compris, c'est que l'image attendue n'était pas au format attendu (28x28), mais en 64x28. Ce qui n'était pas permis. Nous en sommes restés la le mercredi, et le jeudi matin, en revenant, tout cela fonctionnait... Peut-être est-ce une mise à jour de la banque d'image du site de Yann Lecun ? - L'installation de CUDA n'a pas non plus été fonctionnelle le mercredi, mais en redémarrant l'ordinateur pour le lendemain, cela a fonctionner. C'était certainement une nécessité du driver installé pour la carte graphique.

Nous avons donc beaucoup fouiller, sur internet et sur le fichier python donné, afin de résoudre ces problèmes. Il s'avère qu'il s'agissait simplement d'une mauvaise configuration de notre système. Il faut donc bien vérifier que toutes les étapes ont correctement étés faites. La lecture du programme python nous a aussi permis de comprendre certaines caractéristiques de la méthode utilisée pour la résolution de ce problème. Une section plus bas décrira les tests que nous avons effectués, et les éléments que nous avons pu en retirer.

Une fois ces problèmes résolus, le lancement de 'convolution.py' s'effectue correctement, et nous obtenons des taux d'erreurs inférieurs à 1%. En haut du document, il est spécifié que le taux d'erreur attendus est d'environ 0.7%, ce qui correspond à nos résultats.

Références