Projet Fouilles

Jérôme Skoda, Joaquim Lefranc

Arboresence du dossier

- ***datasets :*** *Dossier contenant des archives de dataset*
- ***test_set :*** *Dossier contenant les images de test*
- ***training_set :*** *Dossier contenant les images de training*
- ***training-image-net-org :*** *Dossier contenant les urls des images à télécharger*
- ***clean_up.py :*** *Script permettant de supprimer les «mauvaises» images de image-net.org*
- ***data_augmentation.py :*** *Script permettant de créer des nouvelles variations des images de test*
- ***download_from_net.py :*** *Script permettant la récupération des images de image-net.org*
- ***generate-training-image-net-org.py :***
- ***mkset_by_family.py :*** *Script permettant la création du set champignon par familles*
- ***mkset_by_toxicity.py :*** *Script permettant la création du set champignon par toxicité*
- ***image_classifier.py :*** *Réseau de neurone*

Utilisation des scripts

- ***clean_up.py :*** *Ce script nécessite la création d'un dossier contenant les exemples d'images non souhaitées. Il se lance ensuite avec la commande suivante :* ***clean_up.py <dir> <dir_examples>***

- ***data_augmentation.py :*** *Ce script augmente le dataset du dossier :* ***test_set/***

- ***download_from_net.py :*** *download_from_net.py <dir_out> <first_indice> <urls_file>*

- ***generate-training-image-net-org.py :*** *Télécharge les images contenues dans training-image-net/ (très long)*

- ***mkset_by_family.py :*** *Pas d'arguments, il créer les deux dossiers* ***test_set/*** *et* ***training_set/***

- ***mkset_by_toxicity.py :*** *Pas d'arguments, il créer les deux dossiers* ***test_set/*** *et* ***training_set/***

- ***image_classifier.py :*** *Pas d'arguments, il lance directement le training du réseau.*

Choix du projet et raisons

Nous avons choisis de travailler sur un classifieur d'images couleurs car nous trouvons cela plus excitant de travailler avec des images plutôt que du texte.
Au départ, nous avions en tête l'idée de classer des champignons en fonction de leur toxicité.

Cependant après plusieurs tests il s'avère que notre dataset est trop léger pour ce type de problème. En effet la toxicité des champignons n'est pas clairement visible sur toutes les espèces. De plus il faudrait avoir des machines avec une capacité de calcul monstrueuse, de façon à appliquer le réseaux sur des images de tailles raisonnable dans le but d'analyser tout les détails permettant de distinguer précisément les caractères spécifique à la toxicité.

Nous avons donc bifurqué vers un autre problème : la classification d'espèces de champignon : même problème avec le faible nombre d'images. Alors nous avons choisis de nous intéresser aux espèces de fleurs. Ce problème de classification étant finalement assez général, le type d'images à classer importe peu. Par conséquent nous avons fait des tests avec différents datasets. Nous y reviendrons en détail plus loin dans ce rapport.

Environnement de développement

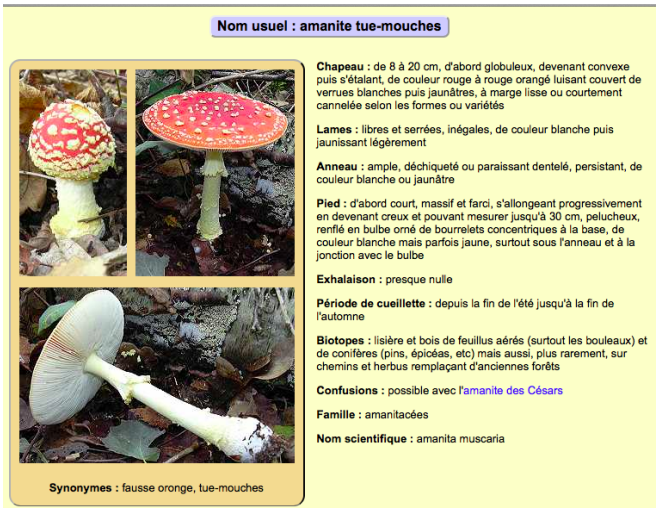
Les outils et framworks
Nous utilisons TensorFlow pour créer notre réseau de neurones. Plusieurs scripts python sont disponibles dans le dossier pour traiter les datasets.

Les machines de test

M1 : i5 4670k @ 4.2Ghz - 24Go DDR3
M2 :

Champignons (toxicité) :

Comme indiqué en préambule, nous avions commencer par la toxicité des champignons en récupérant les images du site suivant : http://mycorance.free.fr/valchamp/toxiques.htm



Avec un script python, nous avons pu télécharger toutes les images en leur ajoutant un tag de toxicité. (mkset_by_toxicity.py)

Champignons (espèces) :

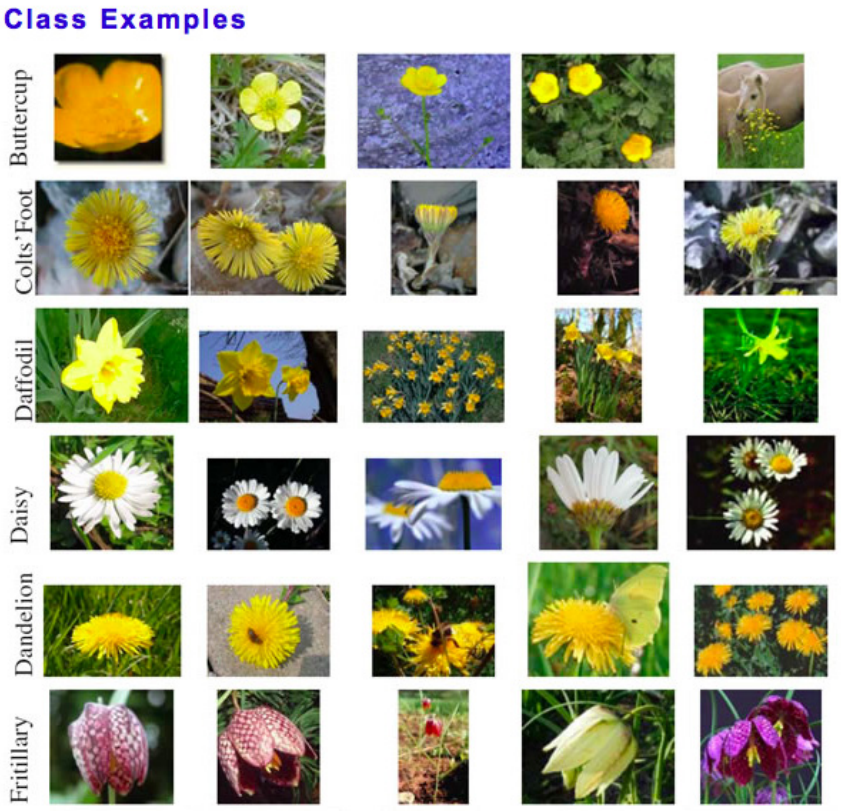
Avec ce même site, nous avons également récupéré toutes les images par espèces dans des dossiers différents grâce au script : mkset_by_family.py.

Les dataset par toxicité et par espèces ne sont pas exploitable car trop peu d'exemples par type dans le cas des espèces. Et trop peu de différences visuelles pour la toxicité. (Surtout sur des images de 64x64 ...)

Fleurs (espèces) :

Suite à de nouvelles recherches de dataset, nous avons trouvé le set de fleur suivant : http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html.

C'est un set contenant 17 espèces de fleurs avec 80 images par espèce. Il est de bonne qualité, cependant les images sont toutes en vracs dans le dossier... Nous avons fait manuellement le travail du réseau de neurone avec une fiabilité de 100% pour des non botanistes. C'est intéressant de voir la formidable capacité de reconnaissance des formes du cerveau humain. Les réseaux de neurones artificiel n'en sont pas encore là !



Set de bonne qualité, avec une augmentation par dessus on se retrouve avec un set bien peuplé avec des exemples propres.

Autres (Fleurs, Cactus, Champignons) :

Pour diversifier nos tests nous voulions faire des essais sur des choses plus évidentes à classifier. En effet les différences entre fleurs, cactus et champignons sont bien marquées. De plus cela nous à permis de développer un script permettant de récupérer facilement des grosses quantités d'images du site image-net.org.

Data augmentation

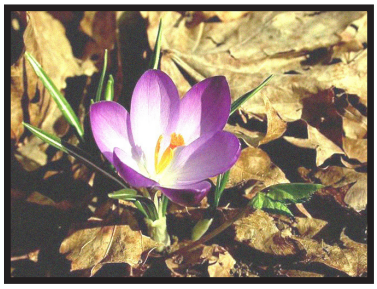
L'augmentation artificielle est une des méthodes pour accroitre le nombre d'exemple de training. Plusieurs opérations sont effectuées sur les images : flou, rotation, flipping, gamma etc. Ceci force le réseau de neurone à abstraire et ne pas se focaliser uniquement sur une seule position ou luminosité.
Nous avons donc conçu un script permettant d'automatiser cette création de variante. Voici un exemple sur une image :

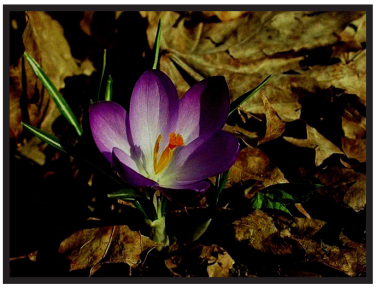

Originale


Blur


Perspective


Zoom


Gamma up


Gamma down


Flip horizontal


Flip vertical


Rotate 180


Rotate 90


Rotate 270

Il y a également toutes les variantes de blur et gamma sur chaque rotation et flip.
Ce qui génère au total 72 images à partir de l'originale.

La première version de notre script permettant l'augmentation souffrait d'un problème sur l'augmentation du gamma. En effet notre méthode «grillait» les images. Ce qui nuisait à l'apprentissage du réseau.

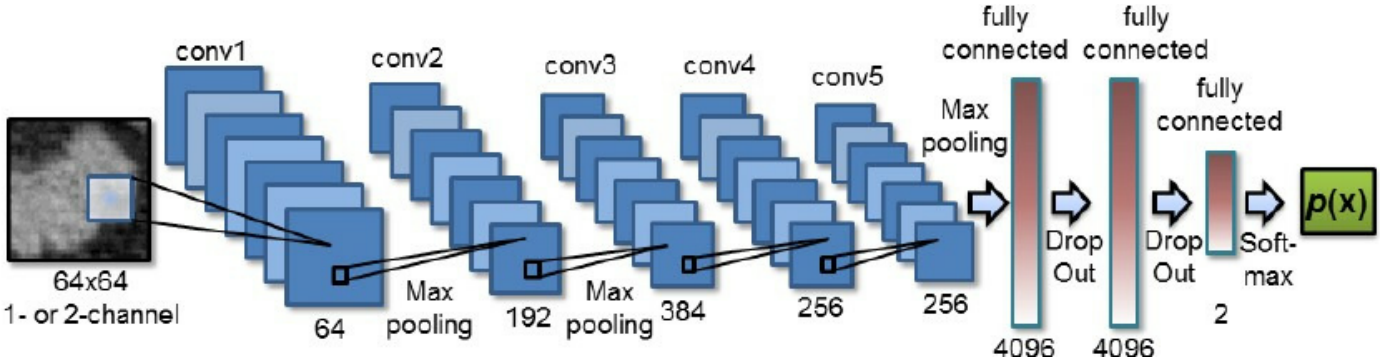
Première version

Deuxième version

Convolutional Neural Network Model

La classification d'images se fait avec un réseau convolutionel, nous avons donc utilisé cette architecture en s'inspirant d'un code trouvé sur le web (Liens dans image_classifier.py).

Cependant le modèle est un peu trop simple et nous n'obtenions pas de bon résultats. Nous avons donc modifié le modèle pour se rapprocher du modele AlexNet.



Les différents paramètres :

Batch_size : *Au début nous avions un batch_size de 64 images. Ce qui s'est révélé trop faible, cela provoquait une oscillation permanante de l'entropy et de l'accuracy. Avec 128 c'était un peu mieux, mais l'idéal étant 256. Finalement avec un réseau convolutionnel relativement profond type AlexNet, il est conseillé de choisir une valeur de 256.*

Optimizer : *Adam Optimizer (Remplace le Gradient Descent) . Avec cet optimizer, le learning rate est automatiquement modifié en fonction de l'avancement de l'apprentissage.*

Learning rate : *C'est la vitesse d'apprentissage, ici 0.001 qui est la valeur standard pour l'AdamOptimizer.*

Dropout : *Ce paramètre permet d'eviter le sur-apprentissage (Overfitting), une valeur de 0.75 implique que les neurones des couches complétement connectées ont 25% de probabilité d'être mis à 0. Ceci force le réseau à créer de nouveaux chemins d'apprentissage.*