Nathan Caracciolo, Sarah Brood, Paco Pompeani, Kevin Célié,

Projet tutoré - Images Satellites

9 mars 2019

Synthèse Etape 2

Construction des premiers algorithmes et premiers résultats

- VGG

- Tensorflow

- Keras
    - algorithme
    - premiers résultats

- Conclusion

- Sources

- Nous avons évalué trois différent algorithmes :
    - VGG
    - Tensorflow
    - Keras

VGG et fine Tuning

```
# load the model
model = VGG16()
# load an image from file
image = load_img('./images/nuage/angers.jpg_tile155.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

L'algorithme fonctionne mais ne reconnaît pas correctement les images que nous lui donnons. En effet il a été entraîné des images provenant de la banque ImageNet, qui n'a pas d'images satellite. De ce fait, malgré la dimension imposante de cette base de données, cet algorithme ne peut être adapté à notre utilisation.

Par exemple : avec une image de nuage on obtient "tasse", ou bien avec une image de désert on obtient "chaîne"..

Tensorflow

Malgré nos recherches, nous avons eu beaucoup de mal à adapter le code que nous avons observé malgré une très grosse tentative avec le code Tensorflow utilisé avec la base de données Mnist (reconnaissance de chiffres manuscrit). Nous avons eu beaucoup de mal a comprendre sa syntaxe, et à adapter les entrées et sorties de code.
Du fait que nous ne comprenons pas cet outil, nous avons préféré se baser sur d'autres modèles comme Keras.

Keras

Algorithme

Avec Keras nous avons réussi à développer un premier algorithme. A partir principalement d'une vidéo, et d'autres ressources, nous avons développé "from scratch" un prototype d'algorithme permettant de reconnaitre des 5 premiers labels qui sont les suivants :

- ➔ nuage
- ➔ mer
- ➔ champs
- ➔ désert
- ➔ forêt

Nous avons développé le chargement des images, la création du dataset (image, label) adapté à l'algorithme, la création du réseau de neurone avec Keras (plusieurs couches convolutionnels et plusieurs couches connectées), et le training sur le dataset avec le réseau de neurone précédemment créé.
Nous avons créé un outil de sauvegarde du réseau de neurone (avec JSON), et un outil de prédiction avec des images extraites du dataset utilisé pour l'entraînement ou des images extérieur.
Cet algorithme fonctionne, nous avons donc joué avec les paramètres (Dropout, nombre de couches, d'époch, et de nombre de données pour le training) pour tester différentes situations et voir comment les paramètres de sortie différaient et donc comment se comportait le réseau de neurone.

Premiers résultats

Au mieux et au plus probable nous avons les résultats suivants :

Loss : 0.0575
Accuracy : 0.9765
Value Loss : 0.1060
Value Accuracy : 0.9800

Avec l'outil de prédiction, avec des images qui on étaient extraites du dataset d'entraînement on obtient ces résultats, non concluants mais encourageants !

 -› champs

 -› champs

 -› forêt

 -› mer

 -› nuage

Nous pensons que ces erreurs proviennent du manque de données, et de la qualité des données, ainsi que la répartition du nombre de données. Effectivement nous avons bien moins d'images représentant le désert que d'images représentant les champs. De plus certaines images sont en noir et blanc, ou sont même à l'oeil humain difficiles à classer parmis ces labels.

Conclusion

Nous pensons continuer à utiliser cet algorithme et Keras en général. Le prochain but serait de se pencher sur chacun des paramètres, les analyser et réussir à les ajuster à la perfection !

Vous pouvez retrouver notre code sur ce GitLab :

https://gitlab.com/Garoli/projet_s4_images_satellites

Sources

- *VGG et fine tuning :*

https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html

- *Tensorflow :*

https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/blob/master/tensorflow-mnist-tutorial/mnist_2.2_five_layers_relu_lrdecay_dropout.py

- Keras

Vidéo : https://www.youtube.com/watch?v=j-3vuBynnOE


```
988/1000 [============================>.] - ETA: 3s - loss: 0.0582 - acc: 0.9762
989/1000 [============================>.] - ETA: 2s - loss: 0.0581 - acc: 0.9762
990/1000 [============================>.] - ETA: 2s - loss: 0.0581 - acc: 0.9763
991/1000 [============================>.] - ETA: 2s - loss: 0.0580 - acc: 0.9763
992/1000 [============================>.] - ETA: 2s - loss: 0.0580 - acc: 0.9763
993/1000 [============================>.] - ETA: 1s - loss: 0.0579 - acc: 0.9763
994/1000 [============================>.] - ETA: 1s - loss: 0.0579 - acc: 0.9764
995/1000 [============================>.] - ETA: 1s - loss: 0.0578 - acc: 0.9764
996/1000 [============================>.] - ETA: 1s - loss: 0.0577 - acc: 0.9764
997/1000 [============================>.] - ETA: 0s - loss: 0.0577 - acc: 0.9764
998/1000 [============================>.] - ETA: 0s - loss: 0.0576 - acc: 0.9765
999/1000 [============================>.] - ETA: 0s - loss: 0.0576 - acc: 0.9765
1000/1000 [==============================] - 267s 267ms/step - loss: 0.0575 - acc: 0.9765 - val_loss: 0.1060 - val_acc: 0.9800
```

```
------------------------
test for desert.jpg
[0]
prediction : champs
test for field.jpeg
[0]
prediction : champs
test for forest.jpeg
[2]
prediction : forêt
test for nuage.jpg
[3]
prediction : mer
test for sea.jpeg
[4]
prediction : nuage
```