Introduction to deep learning: Image recognition, transfer learning and fine-tuning

Stéphane Canu
scanu@insa-rouen.fr, asi.insa-rouen.fr\~scanu

October the 10th 2019

The objectives of the lab

The purpose of this lab is to build an image recognition system, using transfer learning and fine-tuning based on pre tuned convolutional neural networks on ImageNet. (`http://www.image-net.org/`). I started with `https://deeplearningsandbox.com/`. With Python you may use keras pre trained model (see for instance `https://keras.io/applications/`).

Figure 1: Result of practical session 3: an example of CNN at work

Ex. 1 — ImageNet

1. what is ImageNet?
2. how many different kinds of cheese can you find in ImageNet?
3. what is the best classifier on ImageNet and what is its error rate?

Ex. 2 — Build an image recognition system

1. Build an image recognition system for a 1000 everyday object categories (ImageNet ILSVRC) using Keras and TensorFlow[1].
2. import the relevant modules from keras and the pre trained ResNet50[2].

```
from keras.preprocessing import image
from keras.applications.nasnet import preprocess_input, decode_predictions
from keras.applications.nasnet import NASNetLarge, NASNetMobile
```

3. define ResNet50 as you model and check its architecture

```
model = NASNetLarge(input_shape=(331, 331, 3))
#model.summary() # if you want to have a look the the architecture of the NASNet
```

4. open an image ("my_image.jpg" in the following example) representing a single object (if possible represented in ImageNet)

```
img = Image.open("my_image.jpg")
```

5. reshape the image to fit the input format of your model

[1] `deeplearningsandbox.com`
[2] `https://keras.io/applications/`

```
img=np.array(cv2.resize(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB),(331,331)))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
```

6. preprocess the input

```
x = preprocess_input(x)
```

7. get the model predictions

```
preds = model.predict(x)
```

8. display the top 5 recognized objects. Do you find the one of your image?

```
decode_predictions(preds, top=5)[0]
```

Ex. 3 — object recognition using your web cam

1. import the relevant modules from keras and resnet50.

```
import keras
from keras.applications.vgg16 import VGG16
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions
from keras.applications.resnet50 import preprocess_input, decode_predictions

model=keras.applications.resnet50.ResNet50(
include_top=True,
weights='imagenet',
input_tensor=None,
input_shape=None,
pooling=None,
classes=1000
)
```

2. import pylab, opencv and PIL.image. Turn you webcam on.

```
import pylab as pl
import cv2
import PIL.Image

cap = cv2.VideoCapture(0)
```

3. set some variables

```
pred=0
idscreen=0
```

4. perform an inifite loop Capturing frame-by-frame

```
while(True):
ret, frame = cap.read()
```

5. transform the image in an input for your deep neural network

```
img=np.array(cv2.resize(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB),(224,224)))
```

6. preprocess the image as seen before

```
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

7. smooth the prediction

```
alpha=0.9
pred = alpha*pred+(1-alpha)*model.predict(x)
```

8. compute the prediction of your model and display the top 10

```
txt=decode_predictions(pred,10)
for i,p in enumerate(txt[0]):
cv2.putText(frame,"{:.3f}: {}".format(float(p[2]),p[1]),(0,25*i+30),cv2.
    FONT_HERSHEY_PLAIN,2,[1,1,1])
```

9. Display the resulting frame

```
cv2.imshow('frame',frame)
key=cv2.waitKey(1)
if (key & 0xFF) in [ ord(' '),ord('q')]:
break
if (key & 0xFF) in [ ord('s')]:
cv2.imwrite("screen_{}.png".format(idscreen),frame)
idscreen+=1
```

10. When everything done, release the capture by CRTL q

```
cap.release()
cv2.destroyAllWindows()
```

Ex. 4 — Your turn

1. what is the best deep learning framework: caffe, pytorch or tensorflow and keras or others. . . ?
2. what is, in you opinion the best pre trained model?
3. based on your previous work, build an binary object recognition (only two objects) by transfer learning and fine tuning.
   a) choose two classes (cat/dog or muffin/chihuahua or parrot/guacamole or livarot/pont leveque or whatever. . .)
   b) download some (say 10 to 50) images of each class on the web split your images into two sets (training and testing) and setup our data with a training directory and a validation directory as follows:

```
train_dir/
   class1/
   class2/
val_dir/
   class1/
   class2/
```

   c) proceed adapting the code from
   `https://github.com/DeepLearningSandbox/DeepLearningSandbox/blob/master/transfer_learning/fine-tune.py`
4. Is it better to do transfer learning and fine tuning or both?