Tutorial V: Deep models

Bern Winter School on Machine Learning, 28.01-01.02 2019
Mykhailo Vladymyrov

In this session we will use the pretrained Inception model to build own image classifier. We will aslo learn how to save our trained models.

Double-cliquez (ou appuyez sur Entrée) pour modifier

unpack libraries

if using colab, upload the `material.tgz` and run the next cell

```
!tar -xvzf material.tgz
```

1. Load necessary libraries

```
import sys
import os

import numpy as np
import matplotlib.pyplot as plt
import IPython.display as ipyd
import tensorflow as tf
from PIL import Image

# We'll tell matplotlib to inline any drawn figures like so:
%matplotlib inline
plt.style.use('ggplot')
from utils import gr_disp
from utils import inception

from IPython.core.display import HTML
HTML("""<style> .rendered_html code {
    padding: 2px 5px;
    color: #0000aa;
    background-color: #cccccc;
} </style>""")
```

```
def tfSessionLimited(graph=None):
    session_config=tf.ConfigProto( gpu_options=tf.GPUOptions(per_process_gpu_memo
    session_config.gpu_options.visible_device_list = str(0) #use 1st gpu
    return tf.Session(graph=graph, config=session_config)
```

2. Load the model

inception module here is a small module that performs loading the inception model as well as image preparation for the training.

```
net, net_labels = inception.get_inception_model()
```

```
#get model graph definition and change it to use GPU
gd = net

str_dg = gd.SerializeToString()
#uncomment next line to use GPU acceleration
str_dg = str_dg.replace(b'/cpu:0', b'/gpu:0') #a bit extreme approach, but works
gd = gd.FromString(str_dg)

#gr_disp.show(gd)
```

3. Create the graph

This whole model won't fit in GPU memory. We will take only the part from input to the main output and copy it to a second graph, that we will use further.

```
gd2 = tf.graph_util.extract_sub_graph(gd, ['output'])
g2 = tf.Graph() # full graph
with g2.as_default():
    tf.import_graph_def(gd2, name='inception')
```

One can see all operations defined in the graph:

```
gr_disp.show(g2.as_graph_def())
```

```
#get names of all operation
names = [op.name for op in g2.get_operations()]
names
```

4. Build own regressor on top

3 cellules masquées

5. Dataset

8 cellules masquées

6. Load trained variables

2 cellules masquées

7. Loading graph and variables. Saving constant subgraph.

10 cellules masquées

8. Loading constant graph

↳ *3 cellules masquées*

9. Improving the results

↳ *17 cellules masquées*

12. Excercise 1

In the above example is a serius problem: the training and validation datasets are not independent. We generated 5 randomly scaled images from each initial image. With high probability from 5 images (generated from same initial one!) some will end up im the training and some in validation datasets. Since they are generated from the same initial ones, they are not fully independent. This compromises evaluation of model performance, leading to overestimate of the performance.

1. Modify the generation of the training and validation datasets to fulfil requirenment of independance.
2. Check how validation accuracy and loss changes

13. Excercise 2

(Hope we have time left....) Test the performance of model trained on NOT rescaled images, on the wiki screenshots.

```
#copy the above code here
#load the checkpoint ch-0 instead of ch-1

....
```

14. Homework (3 options)

14.1 Improve training set

So far we scaled images as a whole.

- Try to scale differently in x and y direction.
- Check how it affects performace.
- Which else transformation would make sence for the text data?
- Get hands dirty.

14.2 Try to use lower layers' outputs from Inception to build the classifier.

So far we used last output of Inception.

- Look at the Inception more carefully.
- Inspect the size of the data array at different layers.
- Since inside you have 3D data (2D image * features at each position) you will need to flatten it. Look how this is done in last layers (`head0`).
- Ask, google it, and get your hands dirty!

14.3 Classify 3 languages.

So far we tried two languages.

- Create 50 crops of text in another language (better use 5 sources with different fonts, otherwise you risk to learn font, not language), images size > 300 x 300 (to allow scaling).
- Upload them to the `ML3` directory inside of a new directory `xx`.
- Repeat everything with 3 classes.
- Think of the case when this approach won't work.
- Get hands dirty!!!