PRACTICE I:

CONVOLUTIONAL NEURAL NETS

Due date: 14th December 2020

A. GOALS

This practice session pursues the following objectives:

- To be familiar with convolutional neural nets as a powerful supervised machine learning classification technique.

- To be familiar with TensorFlow & Keras as frameworks to develop deep neural networks.

- Understand how to regularize a neural net.

- Use Transfer Learning to improve the accuracy of your net.

B. STUDENT WORK

This is a workgroup practice, where each group must **submit in campus**:

- A PDF document, with all the answers and explanations of your work.

- The code or notebooks derived of your work.

  The final submission will be a compressed file (.zip) with the following name: **ML3_EN2020A_PracticeI_X.zip**; where X is your group letter.

Notes: the notebooks cited in this document are available in Campus: Additional Documentation / Practice I

1. CATS & DOGS

We will start by following the notebook. 01_cnn_template.ipynb, where our goal will be to build a neural net for classifying cats and dogs images.

EVALUATION QUESTIONS AND CODING EXERCISES

You will find three different TODOs in the code. All of them are solved with a couple of few lines of code. Do them, and most important, understand every piece of the code till the end!

[1 point] TODO 1. Add a convolutional layer with the specifications defined in the code.

[1 point] TODO 2. Flatten the output of the last convolutional layer to add a dense layer.

[1 point] TODO 3. Add the dense layer (the one before the last output layer) with the specifications defined in the code.

2. REGULARIZATION

We are at 72% of validation accuracy, while at a 100% percent in training accuracy. Clearly, we are in an overfitting scenario. Too many free params (~10M!) for just a few samples (2000)

In the next notebook 02_cnn_template.ipynb, we are trying to cope with this, by using several regularization techniques.

EVALUATION QUESTIONS AND CODING EXERCISES

[1 point] TODO 1. Data Augmentation. Explain how it is working ImageDataGenerator. Specifically, explain what all the parameters, already set, and their respective values mean. One by one, from rotation_range to fill_mode.

[1,5 points] TODO 2. Dropout.

- Explain Dropout as a regularization technique.
- Add a Dropout layer with a dropout rate of 0.2.
- Try dropout of 0.9 and 0.1 rate. Explain the different behavior.

[0,5 points] TODO 3. Fit the final model following the specifications in the code

3. STAND ON THE SHOULDERS OF GIANTS

We did it very well, but could we do it better?

The answer is yes. How? By using a pre-trained net. In this case we will use Inception V3

For that we will use the notebook 03_cnn.ipynb and 04_cnn_template.ipynb

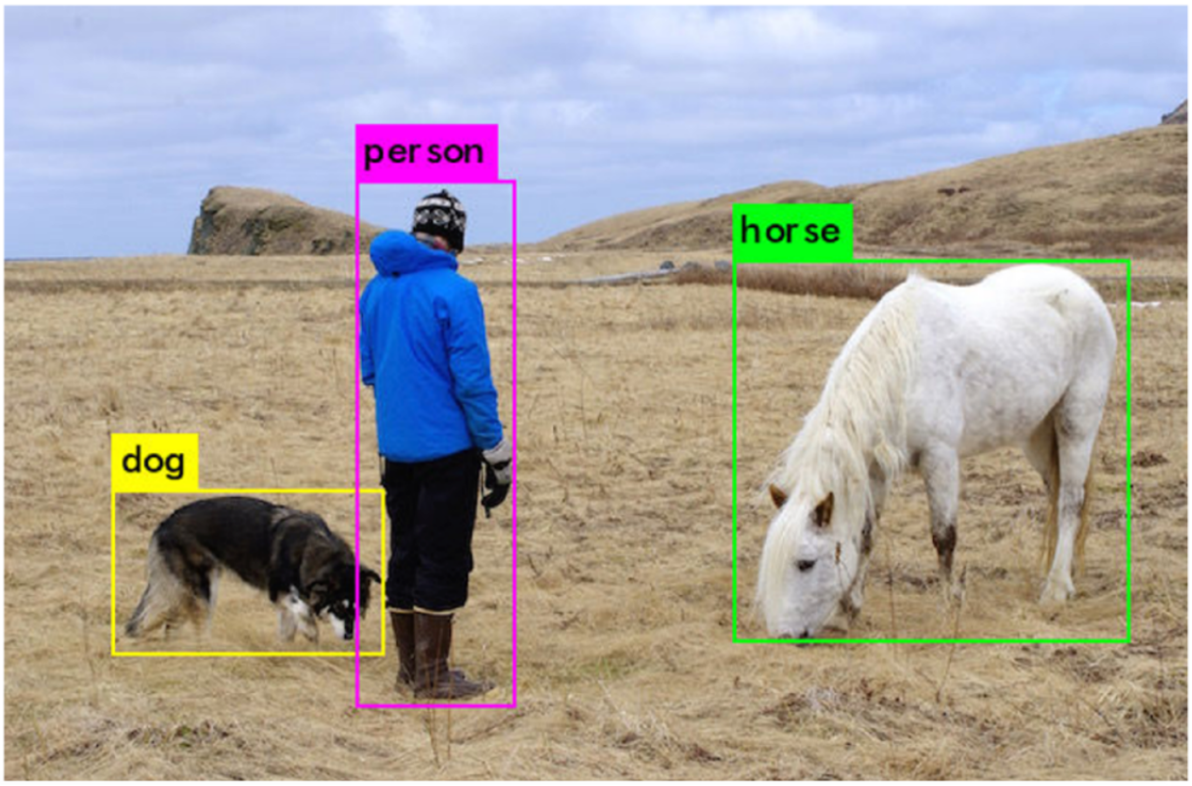
[1 point] TODO 1. Do some research and explain the inception V3 topology. Which database do they use for training it? How was trained?

[1 point] TODO 2. Inception V3 is set by default to admit input images of (299, 299, 3) dimensions; but we want (and we will use) inputs of (150, 150, 3). If the net is already trained, how is that even possible?

[2 points] TODO 3. Let's change the database. Will Inception work well with different objects? Use Inception V3 to outperform the results of a small convnet in a flower database. Complete the code in 04_cnn_template.ipynb to use Inception V3 in this database in the same way we did it for cats & dogs.

4. Object Detection. YOLO nets [Advanced & Optional] [1 extra point]

So far, we have been working on classification. A more complex task consists of locating & classifying different objects in a single image.



This problem needs powerful nets (more complex topologies) and some clever decisions.

In 2015, some smart guys published this paper https://arxiv.org/abs/1506.02640, where they presented a new approach for object detection based on a convolutional neural net topology.

Those nets, under the family name of YOLO (you only look once) – as they just need to see the image once to perform both location and classification -- has been evolving from its first version year after year.

YOLO is now in its version 5.

https://medium.com/towards-artificial-intelligence/yolo-v5-is-here-custom-object-detection-tutorial-with-yolo-v5-12666ee1774e

Out there, there are different implementations of those YOLO topologies, most of them written in PyTorch.

TODO 1. Describe YOLO neural net – no matter the version. I just need to know you know what you are doing ☺

TODO 2. Put to work YOLO_v5 in the database you want and send to me the notebooks with your comments.

Note 1. Note you need training images correctly labelled with the info of the location and objects in the images. Labelling a dataset is a time-consuming task so I recommend you use some datasets publicly available. In the tutorials below, people are using some datasets for Kaggle or Roboflow – if you use them is more than OK.

A tutorial of using yolo v5 to locate some cells in blood images.

https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/

https://colab.research.google.com/drive/1gDZ2xcTOgR39tGGs-EZ6i3RTs16wmzZQ

Yet another tutorial. This is interesting, cause it's a guy using Google Colab and doing what you should do ☺! Also you can see how to apply YOLO on videos (much more fun than images), and there are some links to pre-trained models.

https://www.youtube.com/watch?v=nBjod252PoY&feature=emb_logo

https://www.youtube.com/watch?v=bvzanMwIBaY