Exercise of CNN fine tuning with PyTorch

In this exercise, you will use the extracted features of a pretrained CNN architecture to classify images, and then you will evaluate the performance of the resulting deep network. You must use the provided example as a starting point. You must follow these steps:

1. First of all, you must load the SVHN dataset, which is already available for download from the virtual campus. The original dataset is from: http://ufldl.stanford.edu/housenumbers/
2. You must create a random subset of the original training set by using the torch.utils.data.Subset class, so that the resulting training subset contains exactly 50000 training samples.
3. You must load a pretrained model, modify its final layers to fit the classification problem at hand, and adapt the final layers to the training subset, while keeping the first section of the network frozen. The accuracy measured on the training and validation sets must be stored as the training progresses.
4. After the training is finished, you must plot the confusion matrix for the validation dataset.
5. You must generate a plot that shows the training and validation accuracy (vertical axis) versus the number of training epochs (horizontal axis). Please note that the plot must contain two data series: one for the training set accuracy, and another for the validation set accuracy.

In addition to this, you may complete one or more of the following optional tasks:

Optional task 1: You can plot the ROC curves for the one-versus-rest methodology in order to evaluate the performance of the model on the validation set. Please see:

https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#one-vs-rest-multiclass-roc

You should plot the micro average, macro average and per class ROC curves. Also, you should show the AUC values for all these ROC curves. All curves should be shown in the same plot for comparison purposes.

Optional task 2: You can perform the training procedure using various training subset sizes. Then you should generate a plot where the final training and validation accuracies (vertical axis) are depicted versus the training subset size (horizontal axis). Please note that the plot must contain two data series: one for the training set accuracy, and another for the validation set accuracy.

Optional task 3: You may try with several models. Then you should plot their respective validation set confusion matrices, and compare them.

Optional task 4: In order to generate a good quality PDF, you may do the following.

First, put the following code as the first cell of your notebook:

```
!apt-get install texlive texlive-xetex texlive-latex-extra
pandoc

!pip install pypandoc

from google.colab import drive

drive.mount('/content/drive')
```

Then put the following code as the last cell of your notebook:

```
!jupyter nbconvert --to PDF "/content/drive/MyDrive/Colab
Notebooks/NB.ipynb"
```

where `NB.ipynb` must be replaced by the file name of your notebook. Finally, run the notebook from the beginning. The pdf will be saved on your Google Drive.

For better quality plots use:

```
from IPython.display import set_matplotlib_formats

set_matplotlib_formats('pdf', 'svg')
```

as the first cell of the notebook.

Task submission: Once you have completed your exercise, you must download the .pdf version of your .ipynb file by using the File -> Print -> Save as PDF option of Google Colab (or the method stated as Optional Task 4). Then you must put both the .pdf and .ipynb files into a compressed .zip archive and submit it to the virtual campus task. If the .pdf or .ipynb files are missing, or they do not have the correct content, then the task will have a **fail grade**.