Homework Assignment #1

The one with the CNNs

Description

This homework assignment focuses on developing your skills to work with convolutional neural networks.
It has three main tasks:

1. To **implement the forward pass** of a convolutional filter by hand.
2. To **develop, train and test** your own **small convolutional neural network** for **image classification.**
3. To **fine tune** an existing convolutional neural network for image classification on a similar task.

Task 1 [2p]

The objective of this task is the implementation of **the forward pass** of **your own Convolutional Layer.** It focuses on the understanding of convolutional layer arithmetics in applying **stride**, **dilation** and **grouping.**

Proposed scheme:

- Examine the provided file **conv.py.** Fill in the *forward pass method* for:
    - **MyConvStub**: implements a convolution operation using parameters that govern
        - Kernel size
        - Number of input channels
        - Number of output channels
        - The use of a bias term
        - Stride
        - Dilation
        - Number of groups (for grouped convolution)

    - **MyFilterStub:** the forward pass must **correctly apply** the given *blur filter* (a two dimensional tensor) across a volume having *input_channels* channels. Application of the blur filter is done **channelwise** (i.e. each input channel is blurred using the same filter). The blur filter is provided during testing.

1. **Objective 1:** Use the convolution unit tests in **test_conv.py** to verify your convolution forward pass.
2. **Objective 2:** Use the filter test in **test_conv.py** to verify the correct application of a blur filter to a given input volume.

Task 2 [4p]

This task focuses on developing your skills to write, train and test a simple network model for an image classification problem.

The dataset

The dataset you are using is ***Imagenette***, the 160 x 160 px version.
Imagenette is a subset of 10 easily classified classes from Imagenet (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute).

Proposed schema:

- Design and implement a neural network model containing **Convolutional Layers** and **Linear Layers**, which is **at most 5 layers** deep (pooling, normalization and non-linear activation functions are not counted). An example would be a schema of 3 Convolutional Layers and 2 Linear Layers.
    - Specify and justify the chosen non-linear activation functions and pooling layers (if used)
- Develop your base model by following the Build Model and Classifying CIFAR10 basic tutorials from PyTorch.

- Perform the following experimental training:
    - Train your model **with** and **without batch normalization** applied after the convolution layers
    - Train your model **with** and **without Dropout regularization** in the final Linear Layers
    - Train your model **with** and **without any *Data Augmentation*** methods (e.g. Horizontal Flips, Color Jitter)
    - Train your model **with batch normalization, dropout in final Linear Layers and Data Augmentation methods**

For each experiment, present your results by means of:

- Training / Test **Loss curves +** Training / Test **Accuracy curves**:
    - Plot **with** and **without** normalization results on the same graph
    - Plot **with** and **without dropout** results on the same graph
    - Plot **with** and **without data augmentation** results on the same graph. Specify in the homework report **what kind** of augmentations you used.
    - Plot results of training with all the suggested methods (normalization, regularization, data augmentation)
- Confusion Matrix on the 10 classes

Note!
Aim for training your models for at least 20 epochs.
Aim for exceeding a 60% accuracy metric on the TEST data.

Task 3 [4p]

The objective of this task is to showcase the typical procedure for fine tuning a convolutional neural network architecture.

Proposed schema:

- Use the ResNet-18 torchvision model pre-trained on ImageNet as a **backbone**
- Follow and implement the PyTorch transfer learning tutorial to use the ResNet-18 model as a **feature extractor**
- Modify the tutorial to explore with **unfreezing** the BatchNormalization layers, i.e. adapt the normalization statistics (mean and standard deviation per batch) to the new dataset

Present your results by means of:

- Training and Test Loss curves
- Accuracy curve
- Confusion Matrix
- Compare accuracy and confusion matrix to the results of your simple model from Task 2
- Does **unfreezing** the BN layers help or hinder the performance?