Ex5 - Super Resolution

The purpose of this assignment is to get you familiar with construction of your own dataset (creating the ground-truth images), building a fully-convolutional network and using perceptual loss in order to boost the performance.
We will specifically use the task of image super-resolution to enlarge image resolution by a factor of 2.
Originally, the idea for adding perceptual loss to the regular super resolution loss was suggested in the paper *"Perceptual Losses for Real-Time Style Transfer and Super-Resolution"* (Johnson et al. 2016, http://svl.stanford.edu/assets/papers/JohnsonECCV16.pdf).
Usually this type of task takes a long time to train, because of the complex networks used for it. In this exercise we take the ideas from this paper, but make it manageable, with a rather small network, a small number of training images and working with small images (dimensions). Even with these changes, you must work on a GPU (~30 min for training).

Step 1: Create dataset

We would like to create a dataset for the training of a super-resolution network.
Our basic dataset for this task is the PascalVOC 2011 dataset which is available online (https://pytorch.org/vision/0.8/datasets.html#voc). This dataset contains quality images, which is very important for this task (in comparison to MNIST or CIFAR). Personally, I worked with a subset of this dataset in order to see results faster, without overfitting (1000-200 images for training and validation respectively).

** Pay attention that in this task you do not need the labels and they are actually interrupting in loading a batch (the labels are different in their size). Attached is a function you can use in order to change the labels to have the same size.

For the training, we need pairs of images: low-resolution and its high-resolution version. In this exercise we will apply super resolution from 120x120x3 to 240x240x3.

In order to create the dataset, you can take one of the following options:

- Save these different sizes to disk
- create them on-the-fly in the training process. The DataLoader loads only the high-resolution-images, and at every iteration you create the low-resolution one by resizing it.
- Create your own CustomDataLoader that loads two images at the same time (train,gt)->(low-resolution-image, high-resolution-image).
-

We encourage you to present some of the input and output images. For that, you can use plt.figure(figsize=(12,12),dpi=300)

Step 2: Create an initial model

In this step, we create the fully convolutional model with the architecture described below.

Our network works on the image that has the same size as the output. In other words, the input image (120x120x3) is rescaled to be in the output size (240x240x3, use ‘bilinear’ flag) before it is feeded to the network (or as the first layer of the network).

This results in a blurry 240x240x3 image that does not have the details as the ground truth has. We can say that the purpose of the network is to de-blur the input image and add the details the blurry image doesn't have. This is not an easy task.

Architecture:
We will follow a modified version of the SRCNN architecture - *"Image Super-Resolution Using Deep Convolutional Networks"* (2015, https://arxiv.org/pdf/1501.00092.pdf) with 3 layers:

(1) Conv (9x9 @ 64),
(2) ReLU, (2) Conv (5x5 @ 32), ReLU,
(3) Conv (5x5 @ 3)

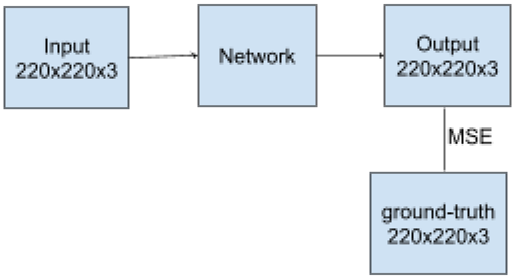
** @ is the number of filters.
** Make sure the image size does not change during the process (use padding)

Step 3: Training the model

In this step we finally train the model.

(1) We train the model using your favorite optimizer and MSE-loss.

MSE-loss, MSE(output,ground-truth):



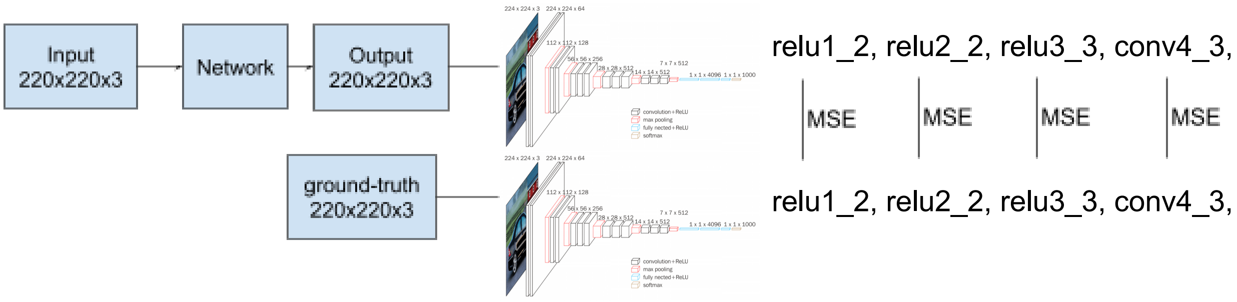
(2) We will train the model with the addition of perceptual loss to the regular MSE-loss

By feeding a pre-trained vgg-16 network with the network output (240x240x3), you get the image activations. Usually the activations that are taking into account for the perceptual loss are: *relu1_2, relu2_2, relu3_3, relu4_3*. This should be done also for the ground-truth image.

Then, the perceptual loss is defined as the MSE-loss between the activations of the output and the ground-truth.

A visualization of the process is described below.

Perceptual loss:
MSE(vgg(output)_relu1_2,vgg(ground-truth)_relu2_2) +…+
MSE(vgg(output)_relu4_3,vgg(ground-truth)_relu4_3)



This task has multiple losses and you should balance them. Play with the weights of the losses and consider even putting a weights=0 if you see that for example the relu4_3 part of the loss does not help for the results.

Evaluating your model:
There is no good metric to evaluate the quality of the super resolution output image (so far…). There are some metrics such as SSIM and PSNR but not something that is worth checking for this exercise. Please assess your network and output by printing the loss(es) and presenting the output images.

More comments:

1) The *colors* of the output image might be a bit different. This is a common issue that papers report. As a post process usually they match the output histogram to the input. You should not worry about it - do not put emphasis on the colors and focus on the details and the sharpness of your output images.
2) This is a very hard task to solve, and we use a very very small and basic network in order to make it applicable for you to train in minutes. Do not expect to have great results (not even close to it). Compare your results to the input image which should be blurry and not to the ground-truth image :)

Step 4: Summary
You should report your MSE and the perceptual loss in two different graphs. Visualize also a subplot of input, output and ground-truth images.