Image super-resolution exercise

Created by Nethanel Shimoni

The purpose of this assignment is to get familiar with construction and training of fully convolutional networks. we will specifically use the task of image super-resolution, and we'll construct several different architectures and compare the results achieved by each of them.

Step 1: Create dataset

We would like to create a dataset and a data-loader for the training of super-resolution network
We will use self-supervision to create our dataset

Our basic dataset for this task will be the PascalVOC 2007 dataset which is available here

Your first task was to create a dataset with images of 3 different sizes:
X - 72x72x3 y_mid – 144x144x3 y_large – 288x288x3

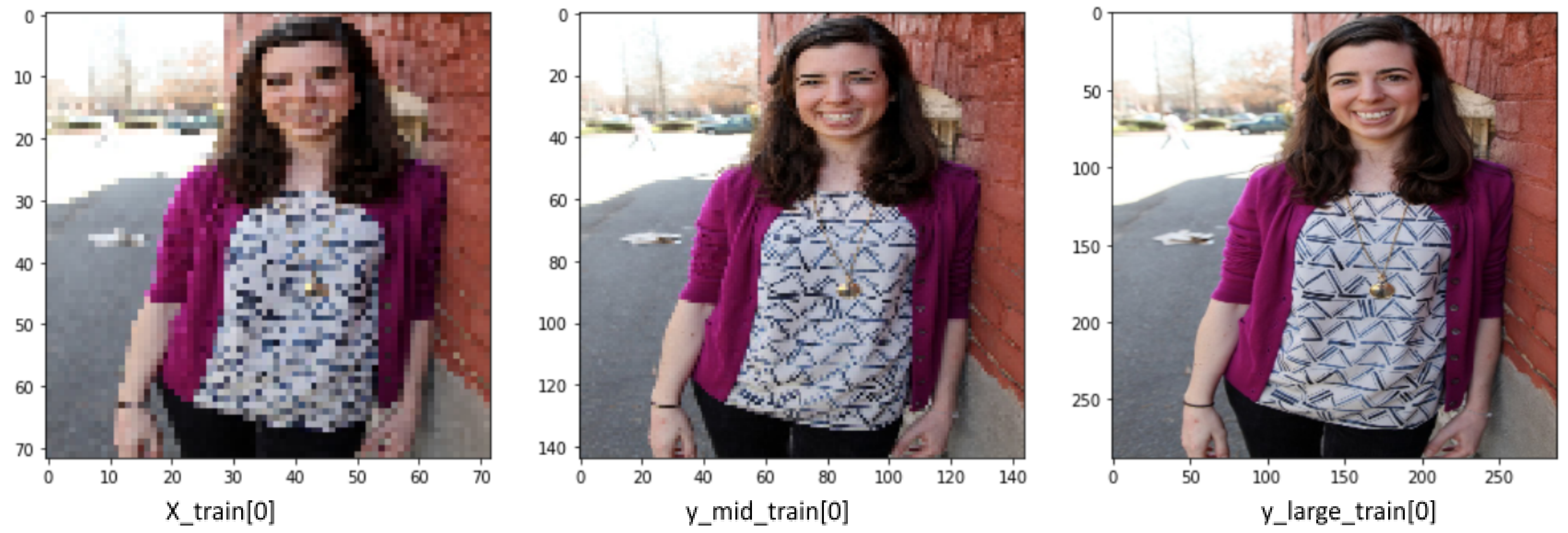
You may choose to either persist these different sizes to disk or create them on-the-fly within your data loader

Once you have loaded and generated the above arrays of input images, we would like to split them into training and validating our model, for simplicity, we will use the first 1000 images (~20%) for validation and the rest for training. (note you should have 5011 images in total but are welcome to add more if you wish to)

* note – I strongly recommend to only work on a sample of the data to make the process of loading and processing faster. This is a good practice for quick development of the loading pipeline and initial model creation. Once everything is working the way we expect it to work, we can increase the number of images we load and be sure that the process runs smoothly.

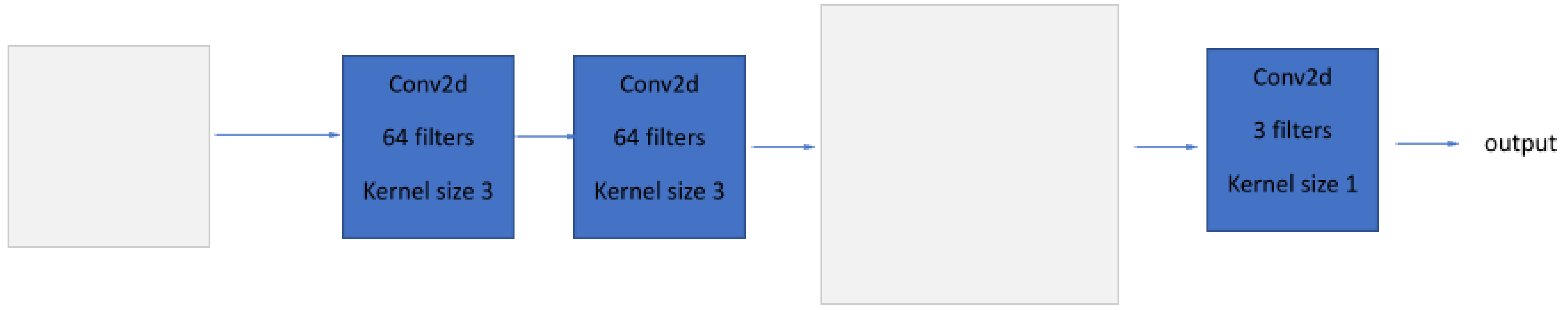
Next, display few images so that we can compare the training results with the desired output. This process is good for verifying that we have got the input we want and will also be useful for visual assessment of our model's results, so make sure you write it as a function for later reuse.

You should get something like this as an output (note the images dimensions):



Step 2: Create an initial model

In this step, create an initial fully convolutional model with the following architecture:

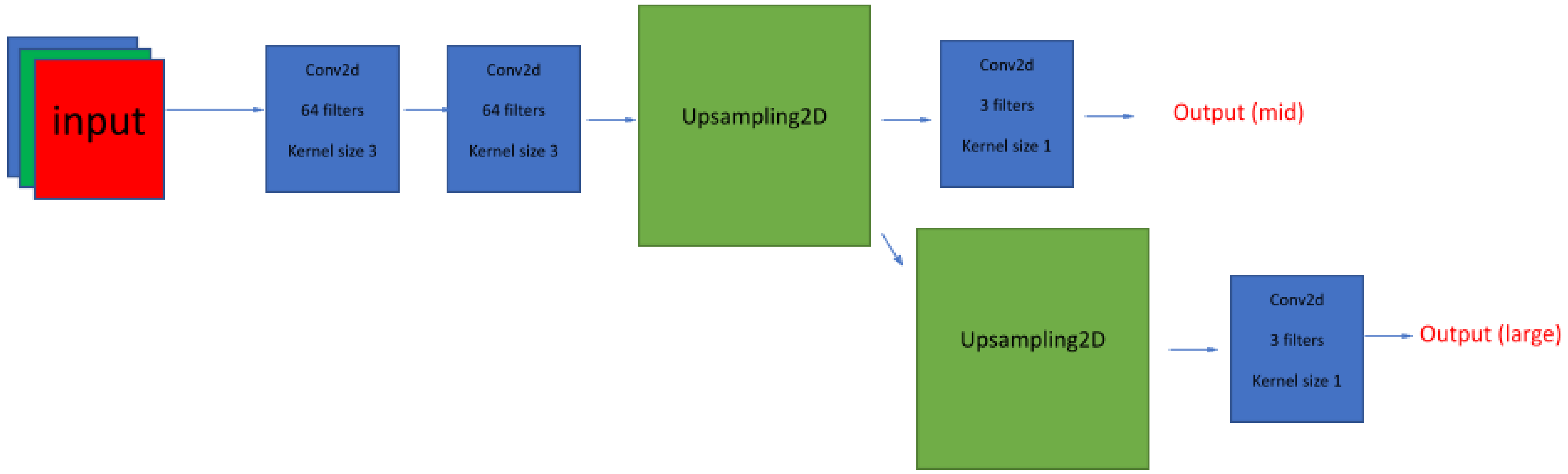


then fit your model to our data using X_train and y_mid_train only

Step 3: Add another upsampling path

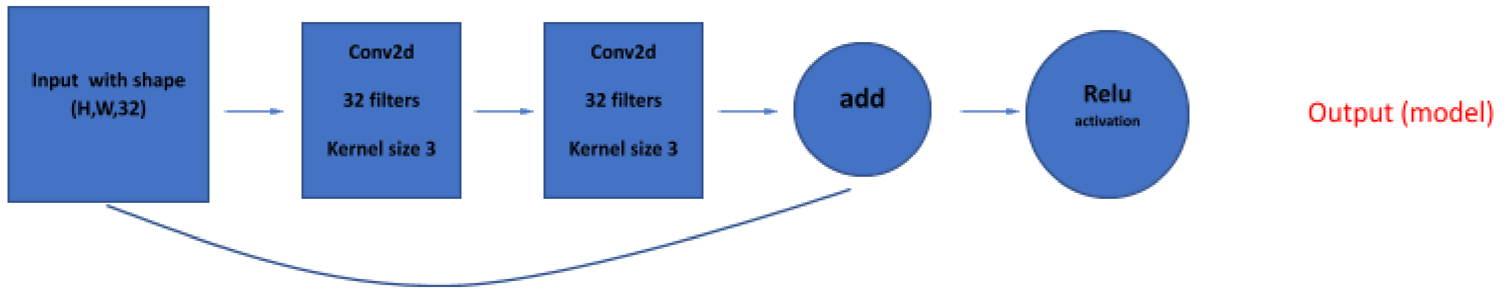
add another block to your model so that you'll have both 144x144x3 output along with 288x288x3 output as follows:

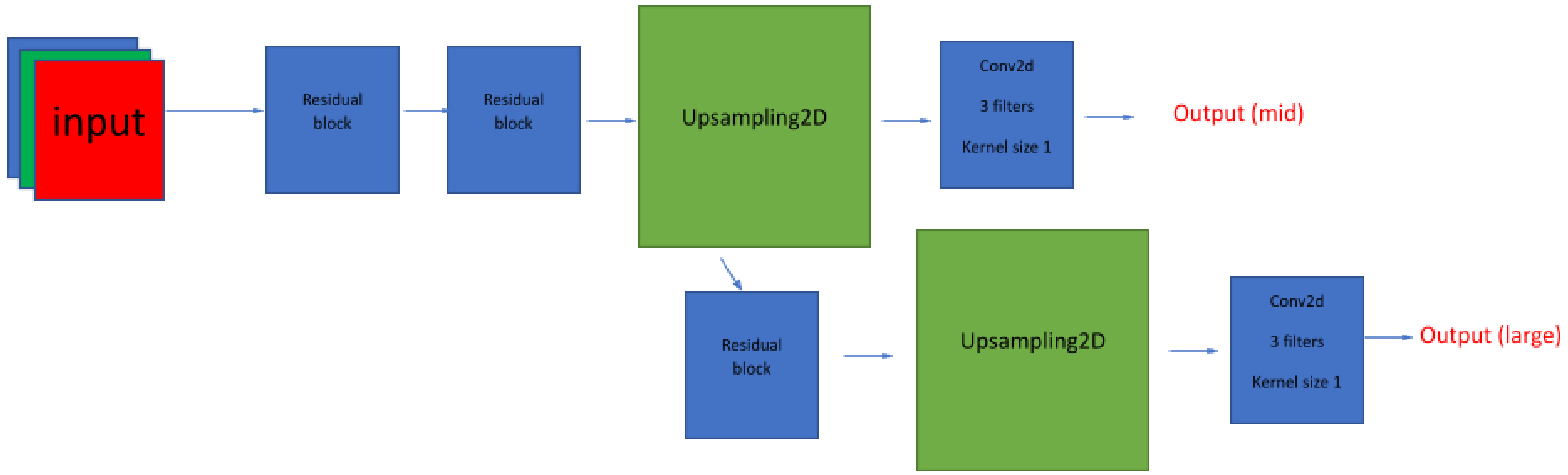
(make sure you understand the dimensions in each step of the process)



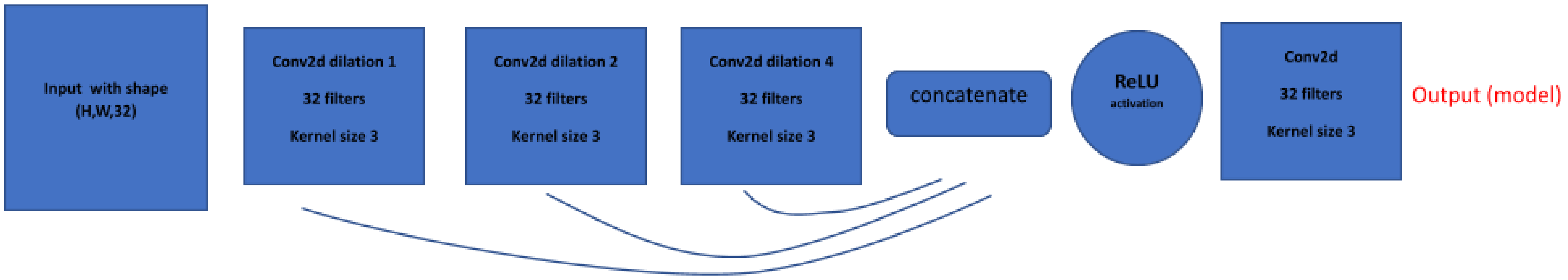
Step 4: Add residual blocks into the process

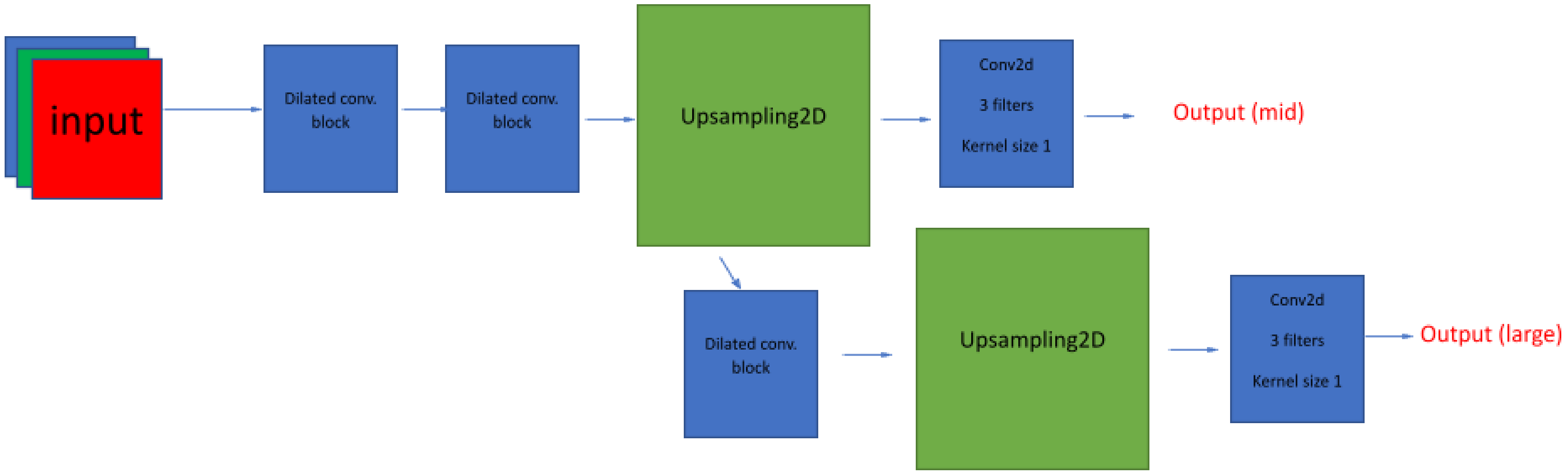
spend a moment to think, what should be the input and output of the residual-block model





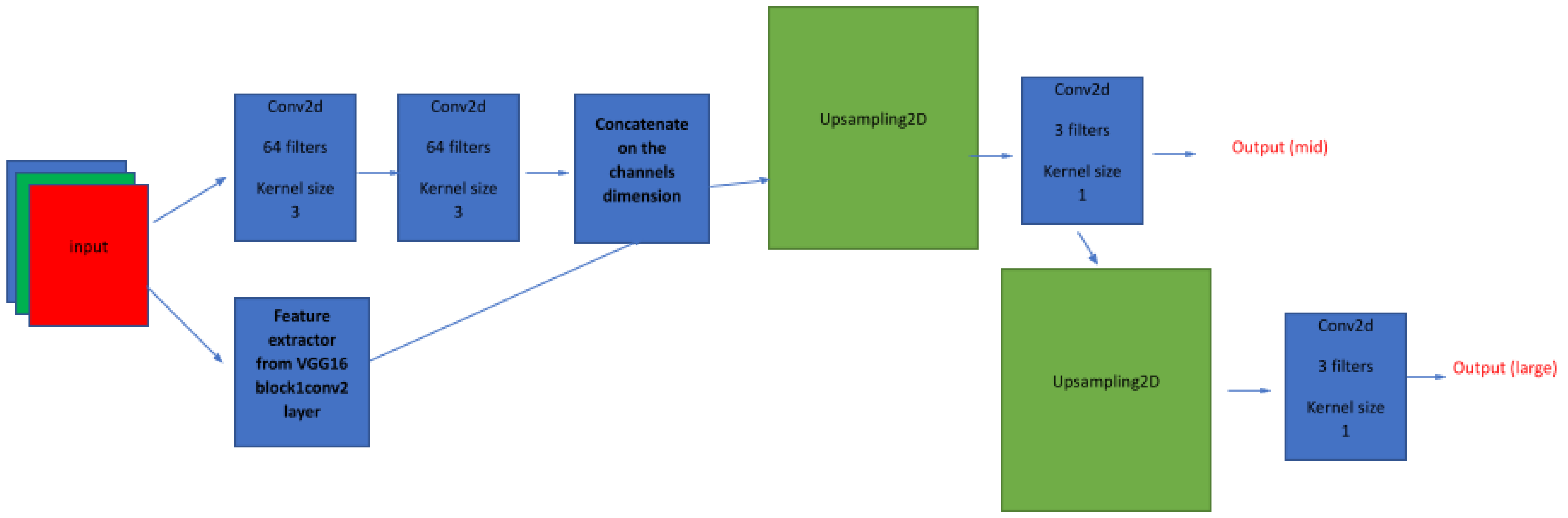
(optional) Step 5: replace the residual blocks we defined above with a dilated (Atrous) convolutional block as described below:





(optional) Step 6: Add a pretrained network

Add pretrained network (either efficientnet / resnet or any other) feature extractor to the network (note that the input to the network is only being read once)



Summary:

Create a report that describes your work and compares the different results you got from the various models that you have trained.

Use both visual examples of results you got in each stage and a relevant metrics table.

Use the MSE loss for the network loss.

Although the architecture and task is different, this blog post might be useful for code examples: https://debuggercafe.com/image-super-resolution-using-deep-learning-and-pytorch/