Course Maths for Data Sciences
TP 1. Image denoising.

Frédéric Richard.

Master 2 Mathématiques Appliquées Statistique,
parcours Data Sciences,
Aix-Marseille Université, 2025.

This practical session focuses on image denoising, which is a well-known inverse problem of image processing. To deal with this problem, we will investigate convolutional neural networks, a residual neural network called DnCNN (Zhang et al., 2016) and convolutional networks inspired from the U-Net architecture (Couturier et al., 2019). We will make experiments on the MNIST database. We will code with Pytorch.

Useful libraries and methods.

[ ]:
```
# Put here the librairies or methods that are needed in your code.

from torch.nn import Module
```

Data Preparation.

- Download the dataset MNIST and prepare learning, validation and test sets for pytorch applications.
- Set noisy images from the clean images of the dataset.

1 DnCNN architecture.

The architecture of the DnCNN is described in (Zhang et al., 2016) and illustrated in Figure xxx.

The architecture includes of a series of blocks composed of convolution (conv), batch normalisation (BN) and ReLU activation. As for VGG, convolutions are defined with kernels of size 3×3. Specific to the architecture, a skip connection links the first and last layers.

Exercise 1

1.1. Using the following template, create a module for the block of a DnCNN. For the convolution, use zero-padding to keep image dimensions throughout the layers.

[ ]:
```
class DnCNN_Block(Module):
    """A Block of a DnCNN.
    """
    def __init__(self, num_of_features=64):
```

```
        """
        Parameters
        ----------
        num_of_features : int, optional
            Number of convolution filters. The default is 16.
        """
        super(DnCNN_Block, self).__init__()
        # To be completed.
        # ...

    def forward(self, x):
        # To be completed.
        return y
```

1.2. Using the block module, create a module for the complete DnCNN architecture, letting the number of layers in arguments, and the skip connection as an option. It is suggested to use the method Sequential.

[ ]:
```
class DnCNN(Module):
    """A  DnCNN.
    """
    def __init__(self, num_of_features=64, num_of_layers=10,␣
↪skip_connection=True):
        """
        Parameters
        ----------
        num_of_features : int, optional
            Number of filters in each block. The default is 64.
        num_of_layers : int, optional
            Number of layers. The default is 10.
        skip_connection : boolean, optional
            True if skip_connection. The default is True.
        """
        super(DnCNN, self).__init__()
        # To be completed.
        # ...


    def forward(self, x):
        # To be completed.
        # ...
        return y
```

1.3. Instanciate the model specifying the class parameters and account for its complexity in terms of number of parameters.

1.4. Learn the model by minimizing the mean square error (MSE) criterion and using the Adam's method.

1.5. On the test set, visualize some results and compute MSE and the peak signal-to-noise ratio, defined for image ranging in $(-1, 1)$ as

$$PSNR = 10 \log_{10} \frac{4n}{|x - \hat{x}|^2}$$

where x and $\hat{x}$ are the clean image and its estimate, respectively, and n is the number of image pixels.

1.6. Compare different specifications of the model and discuss. In particular, test the effect of the skip connection.

2 3. U-Net architecture.

The U-Net architecture was originally proposed in (Ronnenberger et al., 2015) for an application to image segmentation. Similar architecture were further adapted to the image denoising; see for instance (Couturier et al., 2019)

The architecture has a U-shape formed by two symmetric parts, a contracting part and an expansive one. The contracting part is made of a series of consecutive blocks involving - two convolutions steps which increase the input channels by a factor two, - a batch normalization, - a ReLu activation, - a max-pooling with downsampling of a factor two along each image direction.

After each bloch of the contracting part, the image size is reduced by a factor two along each direction whereas the image channels are multiplied by a factor two.

The expansive part composed of consecutive blocks including - a transposed convolution with upsampling of a factor 2, - a concatenation with the symmetric output of the contracting part - a convolution which decrease the input channels by a factor two, - a batch normalization, - a ReLU activation.

After each block of the expansive part, the image size is multiplied by a factor two along each direction whereas the image channels are reduced by a factor two. In that way, the size of the output tensors are the same in the symmetric layers of contracting and expansive parts.

Exercise 2

2.1. Write modules defining the contracting and expansive blocks of the U-Net architecture, and the complete U-Net architecture, keeping the number of layers as an argument of the module.

2.2. Specify the U-Net model and compare its complexity to the one of the DnCNN.

2.3. Learn the model.

2.4. Visualize and test it on some test data. Compare the results to those of a DnCNN. Discuss.