Deep Learning 2025 Assignment 2

This is an **individual assignment** and its deadline is **Friday, December 12, 2025, 22:00**. You must submit your solution electronically via the Absalon home page.

1 Convolutional neural networks

In this assignment, you are going to work on convolutional neural networks, and you are going to compare the structure and efficiency of a feed-forward and a convolutional neural network on a classification task.

In this assignment, you will work with the MNIST dataset which consists of thousands of images of handwritten digits (0–9). Examples are shown in Figure 2.

In the file, `feedForwardMNIST.ipynb`, accompanying this assignment, you will find a complete program for classifying digits using a feed-forward network. Its Pytorch print is as follows:

```
Net(
  (main): Sequential(
    (0): Linear(in_features=784, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=64, bias=True)
    (3): ReLU()
    (4): Linear(in_features=64, out_features=10, bias=True)
    (5): LogSoftmax(dim=1)
  )
)
```

That is, it consists of 3 linear layers, 2 ReLu, and 1 LogSoftmax activation function.

Figure 1: Examples of images in the MNIST dataset ["THE MNIST DATABASE of handwritten digits". Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond].

Your task is to:

1. Run `feedForwardMNIST.ipynb` and give a brief description of each part of the program. What does the transform function achieve? How many parameters does this model use, and how well is it able to correctly classify each digit class 0, 1, . . . , 9?

2. Often will use a linear output unit, that is remove line (5) in the network. We call the linear outputs logits. Making this change to the output also requires that we change the loss function. What is the current loss function used and which loss function should we use with logits? Why is it in general a good idea to work with log softmax/logits output rather than softmax output?

3. Based on `feedForwardMNIST.ipynb`, make a new program `CNNMNIST.ipynb` where the feed-forward neural network has been replaced with a convolutional neural network. The new network must also use:

```
Net(
  (main): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
        =1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
        =1, ceil_mode=False)
    (6): Flatten(start_dim=1, end_dim=-1)
    (7): Linear(in_features=800, out_features=10, bias=True)
    (8): LogSoftmax(dim=1)
  )
)
```

   Why must the Linear layer be preceded by Flatten, and why does it have 800 input features? How well does it classify digits as compared to the feed-forward network, and what are its number parameters relative to the feed-forward network?

2 Autoencoders

This assignment is about two distinct topics:

1. projecting high-dimensional to low-dimensional spaces to achieve a more compact representation (autoencoders and PCA) and

2. learning a latent variable generative model (variational autoencoders).

In the file, `autoencoders.ipynb`, accompanying this assignment, you will find a Jupyter notebook with a working autoencoder, with a model similar to:

```
class FF(nn.Module):
    def __init__(self,dim1,dim2,dim3):
        super().__init__()
        self.main = nn.Sequential(
            nn.Linear(in_features=dim1, out_features=dim2),
            nn.ReLU(),
            nn.Linear(in_features=dim2, out_features=dim3)
        )

    def forward(self, input):
```

Figure 2: Examples of images in the MNIST dataset ["THE MNIST DATABASE of handwritten digits". Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond].

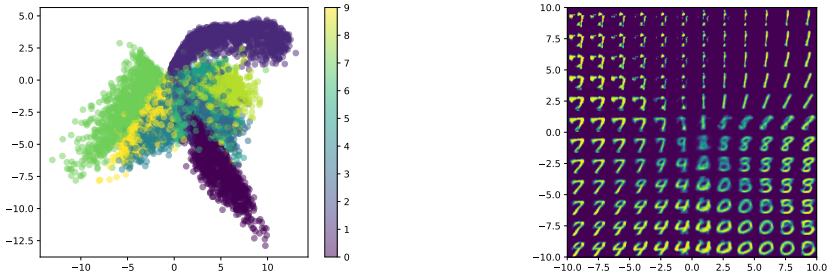

Figure 3: Example of the projection of images from MNIST to the latent space (left), and the resulting images from different latent space values (right).

```
        return self.main(input)
class Autoencoder(nn.Module):
    def __init__(self, dim1, dim2, dim3):
        super().__init__()
        self.encoder = FF(dim1, dim2, dim3)
        self.decoder = nn.Sequential(
            FF(dim3, dim2, dim1),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

The notebook analyzes the thousands of images of handwritten digits (0–9) in the MNIST dataset illustrated in Figure 2, and when executed, it produces 2 plots similar to those shown in Figure 3. Your task is to:

1. Run `autoencoders.ipynb` and give a brief description of each part of the program. What is the transform function doing? The output activation functions used in the encoder and decoder, what are they and why are the chosen to this? What is the loss function used and why this one? Discuss briefly how one could use the beta distribution as likelihood function.

2. Run `autoencoders.ipynb` at least 3 times and give a qualitative comparison between the resulting plots.

Principle Component Analysis (PCA) is a non-deep-learning method for encoding high-dimensional data in lower dimensions. Given a M samples of N-dimensional vectors, (μ, C) the mean vector and the covariance matrix of the samples, and (λ_i, v_i) the eigenvalues and -vectors of C, a new data-point x can be encoded as the vector-dot-product,

$$\alpha_i = (x - \mu) \cdot \sqrt{\lambda_i} v_i, \tag{1}$$

and new data can be generated by,

$$x = \mu + \sum_{i=0}^{K} \alpha_i \sqrt{\lambda_i} v_i, \tag{2}$$

for some $K < \min(M, N)$ and parameters α_i. Your task is to:

3. Make a baseline comparison of the autoencoder above with Principle Component Analysis (PCA), e.g., using the `decomposition.PCA` module in scikit-learn or your favorite PCA tool. You are to produce a 2-dimensional scatter plot and generate example images similar to `autoencoders.ipynb` based on PCA, and you are to give a qualitative comparison of the two models. Consider also, if you run your PCA program twice, will the result change? Note that in scikit-learn's PCA object, the mean vector μ is called `mean_`, the variance λ_i `explained_variance_[i]`, and the corresponding vector `components_.T[:,i]`. Make sure to scale the data appropriately before applying PCA.

4. For the PCA, the number of eigenvectors to use in the reconstruction is a model choice, as is the dimensionality of the latent space in the autoencoder. Define a range of values for the number of eigenvectors and latent dimensions respectively (dim3), e.g., from 2 to 20, and plot the reconstruction errors as a function of dim3. In your opinion, how does the number of eigenvectors used in the PCA model compare with the number of latent variables in the autoencoder, and is there a simple way to determine the optimal value of dim3 for each of the methods?

Autoencoders and PCA are dimensionality reduction techniques and not generative models. As we have in the autoencoder notebook, we can map points in the latent space to the data space using the decoder, but since there is no distribution associated with the latent space, we have no way to correctly sample the latent space.

Variational autoencoders are latent variable generative models. During training, they operate like a stochastic version of the autoencoder, thus their name. During generation of new data (called inference in generative AI lingo), they work by first drawing a random latent vector from the latent prior and then drawing a data point conditioned on the latent vector.

In the file, `vae.ipynb`, accompanying this assignment, you will find a Jupyter notebook with a working variational autoencoder, where the autoencoder class is replaced with a VAE class, which has an encoding method that outputs both the mean and the log variance of the latent distribution and samples from this distribution by reparameterization. The loss function is also adjusted to include both the reconstruction loss and the KL divergence term.

5. Run `vae.ipynb` and give a brief description of each part of the program. Explain each term in the training loss and how the encoder→latent→decoder flow works. Write the equation for the loss (can for example be found in the lecture slides) and check that the program implements it correctly.

6. Run `vae.ipynb` to train a variational autoencoder, sample from the latent space according to a standard normal distribution, and generate new images by passing the sampled latent vectors through the decoder. How does the result of the variational autoencoder qualitatively compare with the autoencoder from task 2 above? Next plot the latent representations of the variational autoencoder and qualitatively compare them with those obtained from the autoencoder. Finally, run `plot_generated_samples` to plot the generated samples. Do you see any differences in the generated images when running the function multiple times?