Université d'Orléans **2025/2026**
Graduate School Orléans Numérique (GSON)

Introduction to Deep Learning
Problem Sheet 3

Exercise 1: The Convolution Operation

(a) 1D Convolution

Consider a 1D input signal

$$x = [1, 2, 0, 1, 5, 0, 4]$$

and a kernel

$$w = [1, 0, -1].$$

Compute the convolution $y = x * w$:

1. without padding, stride = 1
2. with zero-padding of one element on each side, stride = 1
3. with stride = 2

Comment on how padding and stride affect the output size.

(b) 2D Convolution

Given a $6 \times -$ grayscale image

$$I = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

and a 3×3 kernel

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}.$$

Compute the output feature map (stride = 1, no padding). What type of feature is this kernel detecting?

(c) Conceptual Questions

- Why do CNNs require fewer parameters than fully connected networks for image data?
- What is the difference between *translation equivariance* and *invariance*?
- How can pooling layers help reduce overfitting?

Exercise 2: Building a Convolutional Neural Network

(a) Constructing the Model

Using `Keras` and `TensorFlow`, build a small CNN to classify the MNIST dataset. A suggested architecture:

```
Conv2D(32, kernel_size=(3,3), activation='relu')
MaxPooling2D(pool_size=(2,2))
Conv2D(64, kernel_size=(3,3), activation='relu')
Flatten()
Dense(128, activation='relu')
Dense(10, activation='softmax')
```

Train your network for 10 epochs. If you don't want to wait as long, use only 10% of the images. Plot the training and validation accuracy and loss.

(b) Experimentation

Perform the following modifications, and briefly discuss the impact on performance:

1. Add a dropout layer (`Dropout(0.25)`) before the dense layer.
2. Add batch normalization after the convolutional layers.
3. Increase kernel size to $(5, 5)$ and compare validation accuracy.

(c) Overfitting Diagnostics

Explain, using your plots:

- How can you tell if your model is overfitting?
- Which techniques could mitigate this?

Exercise 3: Autoencoders

(a) Constructing a Simple Autoencoder

Use the MNIST dataset again. Build an autoencoder with the following architecture:
Encoder:

```
Flatten()
Dense(128, activation='relu')
Dense(64, activation='relu')
Dense(16, activation='relu')
```

Decoder:

```
Dense(64, activation='relu')
Dense(128, activation='relu')
Dense(784, activation='sigmoid')
Reshape((28,28))
```

Train for 10 epochs using MSE loss. Plot some examples of reconstructed digits.

(b) Denoising Autoencoder

Add Gaussian noise to your input images:

$$x_{\text{noisy}} = x + \mathcal{N}(0, 0.1)$$

Train the same autoencoder to reconstruct the clean image from the noisy one. Compare the result visually.

(c) Latent Space Visualization

Reduce the latent space to only two dimensions for visualisation. Plot the result and color points by their digit labels. Do you observe clustering according to digit class?

(d) Anomaly Detection

Train the autoencoder only on digits `0-8`, then test it on the entire MNIST dataset including `9`. Plot the reconstruction error distribution. How could you set a threshold to detect anomalous samples?

(e) CNN Autoencoder

Replace the dense layers in your autoencoder with convolutional and transposed convolutional layers. Train this convolutional autoencoder on MNIST. Visualize feature maps in the encoder and reconstructions from the decoder.