Université d'Orléans
Graduate School Orléans Numérique (GSON)

Introduction to Deep Learning
Problem Sheet 1

1) Dataset

Take a look at the MNIST dataset:

1. The archive `mnist.zip` is shared on Google Drive. In a Colab notebook, copy it directly into your session (no need to download it to your computer):

   ```
   !gdown 1J91IPeGY3Yzv3D33dqsGVkRWsUqlfGuo
   ```

   Unzip it; you may use `shutil.unpack_archive`. Inspect a few images.

2. How is the dataset organized? How many images are there in total and per digit? What is their size in pixels?

3. Load one image into a NumPy array using `PIL.Image.open`. How is the image encoded? What are the minimum and maximum pixel values? How are white, black, and gray represented? Visualize the image using `matplotlib.pyplot.imshow`.

4. Write a function that loads the entire dataset into two NumPy arrays: `x` (images) and `y` (labels). Ensure that the grayscale values are floating-point numbers in $[0, 1]$ and that, as in the original MNIST and in the lecture, the **background is 0** and the strokes are close to 1. Shuffle the examples using the same permutation for both `x` and `y`.

5. Split both arrays into training (70%), validation (15%), and test (15%) sets. **Note:** any preprocessing (e.g., normalization/standardization) must be *fit on the training set only* and then applied to validation/test (avoid data leakage).

2) Small neural network

Create a small neural network in TensorFlow using `tensorflow.keras.models.Sequential`:

$$\text{Input} \to \text{Flatten} \to \text{Dense}(16, \text{ReLU}) \to \text{Dense}(10, \text{Softmax}).$$

1. Initialize your model (default initializer), *compile* it with `Adam` and `SparseCategoricalCrossentropy`, then *before training* evaluate it on the **validation** set:
   - Sparse categorical cross-entropy.
   - Accuracy.

2. If the softmax outputs are uniform over the 10 classes, what values do you expect for the cross-entropy and the accuracy? Why does your measured value differ slightly?

3. How many weights and biases does the model have?

4. Train the model for 20 epochs.

5. Plot training and validation loss, and report the best validation accuracy and the test accuracy.

6. Is this model sufficient? Train again, replacing the single hidden layer of 16 units with *two* hidden layers of 64 and 32 units. How do the curves and accuracies change?

7. *(Optional)* Train the 16-unit model for 5 epochs on the images *without* the conversion of question 1.4 (background 1, strokes 0). Compare with question 2.4 and explain.

3) Analyze the behaviour of your network

Model:

$$\text{Flatten} \rightarrow \text{Dense}(64, \text{ReLU}) \rightarrow \text{Dense}(32, \text{ReLU}) \rightarrow \text{Dense}(10, \text{Softmax}).$$

1. Create a confusion matrix using the test set. Comment on the most frequent confusions.
2. Visualize some of the misclassified images.
3. Create two images yourself, with a drawing program or as a photo of a drawing on paper: one with a digit and one with a cross. Convert them like the training data: 28×28 grayscale in $[0, 1]$, white strokes on a black background. To resemble MNIST, crop the drawing, scale it so that its longest side is 20 pixels, and centre it. (`Image.open(path).convert("L")` and `.resize((width, height))` help.) Classify both images with your network and comment.

4) Overfitting

Change the following:

- Data size: train on only 5,000 samples.
- Use a larger model (e.g., Dense(512)–Dense(512)–Softmax).

Train for 50 epochs and plot training and validation loss. Identify the epoch where validation loss increases while training loss continues to decrease. What happens to the validation *accuracy*?

5) Framing: Classification vs. Regression

Frame the problem as a regression problem.

1. Train the network with two hidden layers of 64 and 32 units, but use a *single scalar* output with a *linear* activation and use MSE as the loss function. Report the test RMSE and also compute a "rounded accuracy" by rounding predictions to the nearest integer (clipped to 0–9); compare with the classification baseline.
2. Explain why, for ambiguous inputs where $p(Y \mid X = x)$ is multi-modal, the regression predictor may yield values between classes that are not valid labels. When is regression an appropriate modeling choice, and when is classification necessary?