Université d'Orléans
Graduate School Orléans Numérique (GSON)

Introduction to Deep Learning
Problem Sheet 3

Exercises 1 and 2 are on paper, exercises 3 and 4 in a Colab notebook with TensorFlow and Keras. The multiple-choice test at the end of the session covers practical 2 and lecture 3: exercises 1 and 2 are a good preparation.

As in lecture 3, a convolution does **not** flip the kernel (mathematicians call this a cross-correlation). In 1D and in 2D, with a bias b:

$$y_i = \sum_{k=1}^{K} w_k \, x_{i+k-1} + b, \qquad y_{ij} = \sum_{m=1}^{k} \sum_{n=1}^{k} w_{mn} \, x_{i+m-1,\, j+n-1} + b, \qquad n_{\text{out}} = \left\lfloor \frac{n_{\text{in}} + 2p - k}{s} \right\rfloor + 1.$$

1) Convolution by hand

(a) 1D convolution. Take the input $x = (1, 2, 0, 1, 5, 0, 4)$ and the kernel $w = (1, 0, -1)$, without bias. Compute the output

1. without padding, stride 1;
2. with one zero added at each end (padding $p = 1$), stride 1;
3. without padding, stride 2.

Check the output sizes with the formula above. Which case corresponds to `padding="same"` in Keras, which one to `padding="valid"`?

(b) 2D convolution, pooling and shifts. Take the 6×6 grey image I and the 3×3 kernel K:

$$I = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}, \qquad K = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}.$$

1. What is the size of the output (stride 1, no padding, no bias)? Compute the feature map. Which kind of feature does this kernel detect? Where is the output positive, where negative?
2. Apply a 2×2 max pooling to the feature map.
3. The image is shifted by one pixel to the right: a column of zeros enters on the left, the last column is lost. Without redoing the whole calculation, how does the feature map change? Compute the new pooled output. Is the feature map *equivariant* to the shift? Is the pooled output *invariant*?

(c) Receptive field. Convolutional layers with 3×3 kernels and stride 1 are stacked. On how many pixels of the input does one output of the first layer depend? Of the second? Of the third? How does the answer change if a 2×2 max pooling is inserted after the first layer?

(d) Questions.

- Why does a convolutional layer have far fewer parameters than a dense layer acting on the same image? Does its number of parameters depend on the size of the image?
- What is the difference between translation equivariance and invariance? Which operations of a CNN provide which?

2) Shapes and parameters

A convolutional layer with C_{out} filters of size $k \times k$, acting on C_{in} input channels, has $(k^2 C_{\text{in}} + 1)\, C_{\text{out}}$ parameters.
(a) Complete the table for the network used in this practical, for 28×28 grey images (`padding="valid"`, the Keras default). Which layer holds most of the parameters?

Layer	Output shape	Parameters
`Input(shape=(28, 28, 1))`	$28 \times 28 \times 1$	0
`Conv2D(32, 3, activation="relu")`		
`MaxPooling2D(2)`		
`Conv2D(64, 3, activation="relu")`		
`MaxPooling2D(2)`		
`Flatten()`		
`Dense(128, activation="relu")`		
`Dense(10, activation="softmax")`		
Total		

(b) For each variant, give the shape that enters `Flatten`, the number of parameters of the layer that changes, and the total:

1. 5×5 kernels in both convolutional layers;
2. `padding="same"` in both convolutional layers;
3. without the second `MaxPooling2D`;
4. colour images of 32×32 pixels (3 channels, as in the CIFAR-10 dataset) instead of MNIST.

Which variant would you expect to overfit most when only a few thousand training images are available?
(c) The small CNN of lecture 3 is `Conv2D(16, 3)` → `MaxPooling2D(2)` → `Conv2D(32, 3)` → `MaxPooling2D(2)` → `Flatten` → `Dense(10)`. Check that it has 12 810 parameters. How many of them are in the output layer?
(d) Two ideas from lecture 3.

- A residual block computes $\mathbf{y} = \mathbf{x} + F(\mathbf{x})$. Compute $\partial \mathbf{y} / \partial \mathbf{x}$. Why does this help to train networks with more than a hundred layers?
- Transfer learning: MobileNetV2, pretrained on ImageNet, is loaded without its output layer, frozen (`base.trainable = False`), and followed by `Dense(10, activation="softmax")`. Its output (with `pooling="avg"`) is a vector of 1 280 numbers. Which weights does `fit` change, and how many are there?

3) A convolutional network for MNIST

Select a GPU in Colab (*Runtime* → *Change runtime type* → *T4 GPU*). Without a GPU, train on the first 10 000 training images only.
(a) Load the data. The archive stores black digits on white; as in practical 1, we invert it so that the background is 0:

```
!gdown 1J91IPeGY3Yzv3D33dqsGVkRWsUqlfGuo
import glob, os, shutil
import numpy as np
from PIL import Image
shutil.unpack_archive("mnist.zip", "mnist")
paths = sorted(glob.glob("mnist/*/*.png"))
x = np.array([1 - np.asarray(Image.open(p)) / 255 for p in paths], dtype="float32")
y = np.array([int(os.path.basename(os.path.dirname(p))) for p in paths])
idx = np.random.default_rng(0).permutation(len(x))
x, y = x[idx], y[idx]
```

What is the shape of `x`? A `Conv2D` layer expects one more axis, the channels: reshape `x` to $(n, 28, 28, 1)$. Split the data into training (70 %), validation (15 %) and test sets (15 %).
(b) Build the network of exercise 2(a) with `keras.Sequential`. Compare `model.summary()` with your table.
(c) Compile it with `Adam`, the sparse categorical cross-entropy and the accuracy, and train it for 10 epochs with batches of 64, with the validation set. Plot the training and validation loss, and the training and validation accuracy, against the epoch. Evaluate the model on the test set, once, at the end of the practical.

(d) Experiments. Change one thing at a time, and compare the validation accuracy and the curves with those of (c):

1. add `Dropout(0.25)` between `Flatten` and `Dense(128)`. Why can the training accuracy be lower than the validation accuracy?
2. add `BatchNormalization()` after each convolutional layer. How many parameters does each of these layers add? Why are some of them reported as non-trainable?
3. use 5×5 kernels, then `padding="same"`: check `model.summary()` against exercise 2(b).

Train the network of (c) a second time: how much does the validation accuracy change from one run to the next? Are the differences between the variants larger than this?
(e) Overfitting. Using your plots: how can you tell that a model overfits? Which techniques of lectures 2 and 3 reduce overfitting?

4) Inside the network

(a) Filters and feature maps. The weights of the first layer, `model.layers[0].get_weights()[0]`, have the shape $(3, 3, 1, 32)$. Plot the 32 filters as 3×3 images (`cmap="bwr"`). Then plot the 32 feature maps of one test digit, computed by

```
first = keras.Model(model.inputs, model.layers[0].output)
maps = first.predict(x_test[:1])          # shape (1, 26, 26, 32)
```

What do the learned filters look like? Compare with the kernel of exercise 1(b).
(b) Shifts and data augmentation. Shift the test images to the right by $s = 0, 1, \ldots, 6$ pixels and plot the test accuracy against s:

```
def shift_right(images, s):               # zeros enter on the left
    out = np.zeros_like(images)
    out[:, :, s:] = images[:, :, :28 - s]
    return out
```

Then train a new network whose first layer, right after the input, is

```
layers.RandomTranslation(0.2, 0.2, fill_mode="constant")
```

which shifts every training image randomly by up to 20 % of its size (it does nothing at test time). Compare. Which part of the network is not translation invariant?
(c) Adversarial examples (if you have time). Compute the gradient of the loss with respect to the *pixels* of 1 000 test images, the weights staying fixed:

```
x0 = tf.constant(x_test[:1000])
with tf.GradientTape() as tape:
    tape.watch(x0)
    loss = keras.losses.sparse_categorical_crossentropy(y_test[:1000], model(x0))
x_adv = tf.clip_by_value(x0 + 0.15 * tf.sign(tape.gradient(loss, x0)), 0, 1)
```

Compare the accuracy on `x_adv` with the accuracy on the original images and on images with random noise of the same size, ± 0.15 on every pixel. Look at a few adversarial images: would you still recognise the digits?