DD2424
Deep Learning in Data Science
Assignment 3 Bonus Points

Franco Ruggeri, fruggeri@kth.se

May 27, 2020

1 Stride and zero-padding

I made the convolution implementation more generic to include stride and zero-padding.

The stride requires to modify the functions *MakeMXMatrix* and *MakeMFMatrix*. Considering the example of the instructions, with stride 2 the equations (15) and (25) become:

$$M^{filter}_{\mathbf{F},4} = \begin{bmatrix} \leftarrow & V_{\mathbf{F}} & \rightarrow & 0_{nfx4} & 0_{nfx4} \\ 0_{nfx8} & \leftarrow & V_{\mathbf{F}} & \rightarrow & \end{bmatrix} \tag{1}$$

$$M^{input}_{X,2,3} = \begin{bmatrix} I_3 \otimes vec(X_{:,1:2})^T \\ I_3 \otimes vec(X_{:,3:4})^T \end{bmatrix} \tag{2}$$

where the differences with respect to the instructions are highlighted.

The zero-padding, instead, requires to modify both forward and backward pass:

- In the forward pass, $X^{(i)}_{batch}$ needs to be padded adding $d \cdot padding$ rows above and below, where d is the number of rows of a non-vectorized sample (i.e. number of filters in the previous layer, or one-hot-encoding length for the first layer).
- In the backward pass, the first and the last $d \cdot padding$ rows of G_{batch} need to be removed between equations (39) and (40) of the instructions, where

d has the same meaning as the previous bullet point. I omit the explicit derivation for brevity, but here I report my steps to find the solution:

1. I considered the computational graph between the $i{-}th$ and $(i{+}1){-}th$ convolutional layers for one sample X. The zero-padding is a new computation, so there is a new node $X_{pad}^{(i)}$ after $X^{(i)}$.
2. I computed $\frac{\partial vec(X_{pad}^{(i)})}{\partial vec(X^{(i)})}$. The result was a padded identity matrix.
3. I computed $\frac{\partial l}{\partial vec(X)} = g \cdot \frac{\partial vec(X^{pad})}{\partial vec(X)}$, where $g = \frac{\partial l}{\partial vec(X^{pad})}$. The result was g without the first and the last $d \cdot padding$ columns. This observation made the computation much easier for a batch.
4. I generalized to a batch, so g as column of G_{batch} and back-propagation as removal of first and last $d \cdot padding$ rows.

I modified the best ConvNet from exercise 1 by setting $stride = 1$ and $padding = \frac{k-1}{2}$, common choice to keep the same spatial dimension through the layers. Here the details:

- Parameters: $k_1 = 5$, $n_1 = 100$, $stride_1 = 1$, $padding_1 = 2$, $k_2 = 3$, $n_2 = 100$, $stride_2 = 1$, $padding_2 = 1$.
- Hyper-parameters: $batch\_size = 100$, $\eta = 0.001$, 20000 updates[1].
- Validation accuracy: 57.94% (without padding it was 55.56%).
- Validation accuracy per class: see table 1.
- Confusion matrix: see figure 1.
- Learning curves: see figure 2.

[1] As already mentioned in the report for the exercise 1, I used the validation set to keep a record of the best network parameters while training, to avoid overfitting. I will not repeat this in the following.

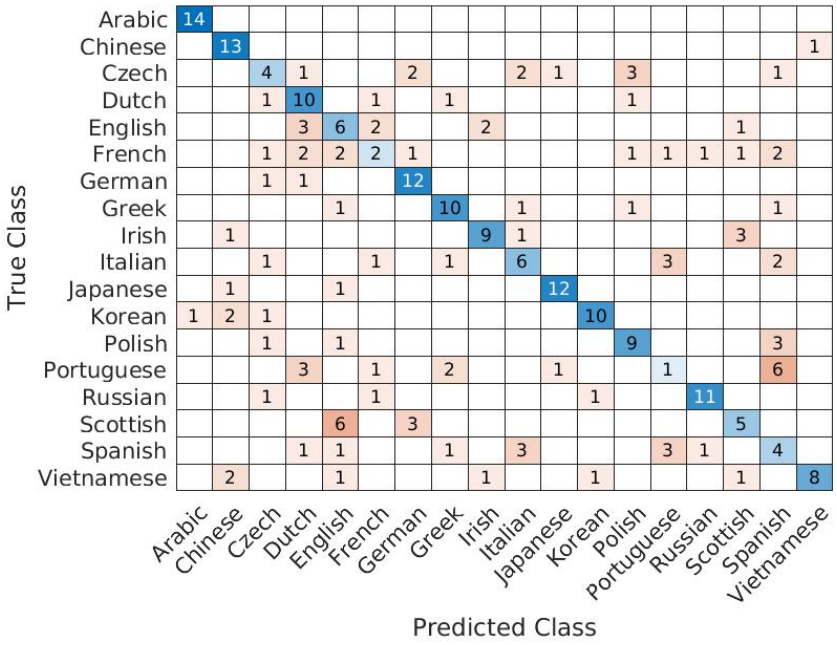

Figure 1: Confusion matrix on the validation set for the ConvNet with stride and padding.

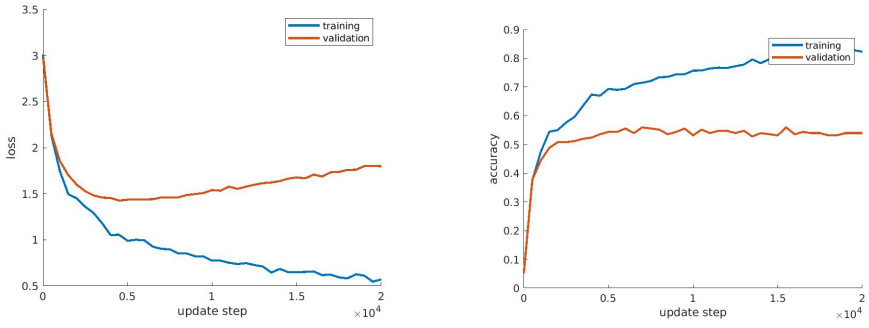

Figure 2: Learning curves for the ConvNet with stride and padding.

Class	Validation accuracy (%)
Arabic	100.00
Chinese	92.86
Czech	28.57
Dutch	71.43
English	42.86
French	14.29
German	85.71
Greek	71.43
Irish	64.29
Italian	42.86
Japanese	85.71
Korean	71.43
Polish	64.29
Portuguese	7.14
Russian	78.57
Scottish	35.71
Spanish	28.57
Vietnamese	57.14

Table 1: Validation accuracy per class for the ConvNet with stride and padding.

2 Adding convolutional layers

I made the code generic to define a L-layer ConvNet. After searching for good parameters, I trained a 3-layer ConvNet whose details are as follows:

- Parameters: $k_i = 3$, $n_i = 100$, $stride_i = 1$, $padding_i = 1$, $1 \leq i \leq 3$.
- Hyper-parameters: $batch\_size = 100$, $\eta = 0.0005$, 40000 updates. Note that I reduced the learning rate because with $\eta = 0.001$ the learning was unstable.
- Validation accuracy: 61.11% (with 2 layers it was 57.94%).
- Validation accuracy per class: see table 2.
- Confusion matrix: see figure 3.
- Learning curves: see figure 4.

I searched up to 5 layers, but with more than 3 layers the validation accuracy remained more or less the same, actually a bit worse. With batch normalization and more trainning data, especially for the smallest classes, more layers would most probably improve the generalization.

Class	Validation accuracy (%)
Arabic	100.00%
Chinese	92.86%
Czech	50.00%
Dutch	78.57%
English	64.29%
French	42.86%
German	92.86%
Greek	71.43%
Irish	50.00%
Italian	57.14%
Japanese	78.57%
Korean	50.00%
Polish	42.86%
Portuguese	14.29%
Russian	85.71%
Scottish	21.43%
Spanish	28.57%
Vietnamese	78.57%

Table 2: Validation accuracy per class for the 3-layer ConvNet.

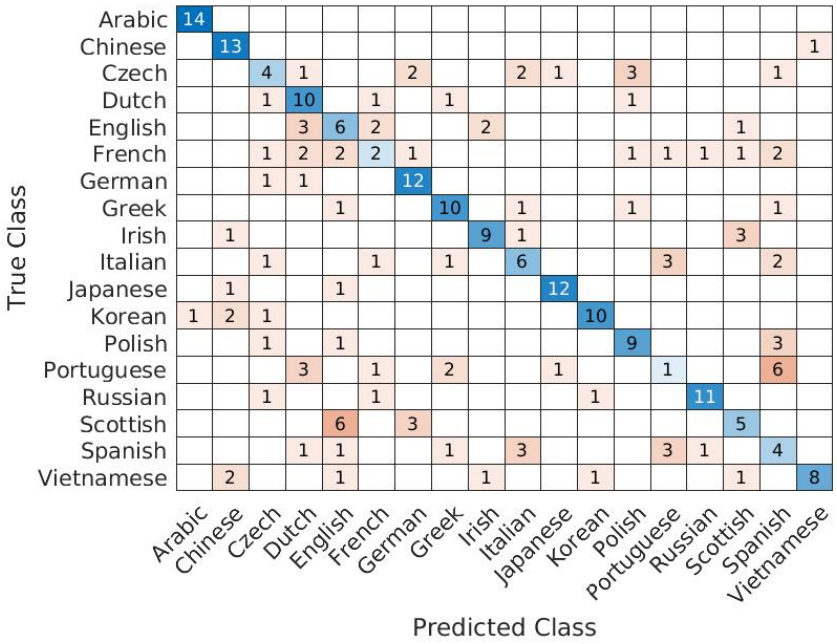

Figure 3: Confusion matrix on the validation set for the 3-layer ConvNet.

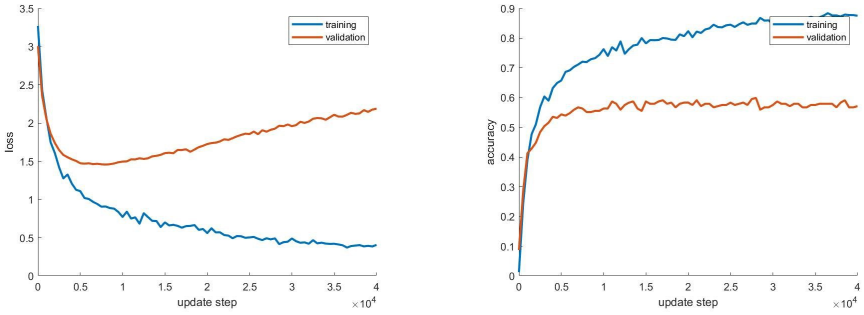

Figure 4: Learning curves for the 3-layer ConvNet.