DD2424
Deep Learning in Data Science
Assignment 3

Franco Ruggeri, fruggeri@kth.se

May 27, 2020

1 Gradients computation

I successfully managed to write the functions to correctly compute the gradients analytically. To check it, I compared the results with the numerical estimations of the gradients given by the function *NumericalGradient* (provided) for one mini-batch using the following setting: $k_1 = 5$, $n_1 = 5$, $k_2 = 5$, $n_2 = 5$, $batch\_size = 3$. The results are shown in table 1 and prove the correctness of the analytical computations, given that the differences are really small.

Gradient	Max absolute difference	Max relative difference
$\frac{\partial L}{\partial F_1}$	4.48e-10	4.09e-07
$\frac{\partial L}{\partial F_2}$	4.38e-10	1.46e-06
$\frac{\partial L}{\partial W}$	5.65e-10	2.65e-06

Table 1: Tests on gradients.

2 Optimizations

I implemented the proposed efficiency gains:

(a) Pre-compute $M^{input}_{x_j,k_1,n_1}$.

(b) Compute $M^{input}_{x_j,k_i}$ instead of $M^{input}_{x_j,k_i,n_i}$ (also for the first layer).

To analyze the speedup, I compared the execution times with and without the optimizations for the following setting: $k_1 = 5$, $n_1 = 10$, $k_2 = 5$, $n_2 = 10$, $batch\_size = 100$, $\eta = 0.001$, $epochs = 5$. The results, reported in table 2, clearly show that the speedup is very high. A further observation is that the optimization (a) gives a higher gain over time, because the pre-computation is done once at the beginning.

Optimizations	Execution time (s)	Speedup
-	176.715	-
(a)	143.063	1.24x
(b)	30.873	5.72x
(a)+(b)	25.501	6.93x

Table 2: Tests for optimizations.

3 Imbalanced dataset

The provided dataset is quite imbalanced and is shown in figure 1. To compensate this fact, I implemented the second proposed strategy, i.e. using a randomly sampled balanced sub-set of the original training set as effective training set at each epoch. The balanced effective training set is composed of n_{small} samples from each class, where n_{small} is the number of samples in the smallest class.

To analyze the effect I trained two ConvNets, without and with compensation, using the proposed setting ($k_1 = 5$, $n_1 = 20$, $k_2 = 3$, $n_2 = 20$, $\eta = 0.001$, $batch\_size = 100$) and about[1] 50000 updates. I also used the validation set to keep a record of the best network parameters while training, to avoid overfitting.

Figure 2 shows the confusion matrices. Without compensation the network does not learn at all to classify some small classes (blank columns, e.g. *Vietnamese*) and there are a lot of wrong predictions to the dominating classes (red cells in columns *English* and *Russian*). With compensation, instead, all the classes are considered and learnt (more or less accurately, but here the focus is not on this).

The learning curves are shown in figure 3. We can note that loss and accuracy are better on the training set and worse on the validation set when not using the compensation. The former assertion is rather obvious, because by compensating we alter the training process to consider the samples of the smallest classes more times than the samples of the dominating classes, in order to fit all the data at

[1] *About* because the algorithm runs complete epochs.

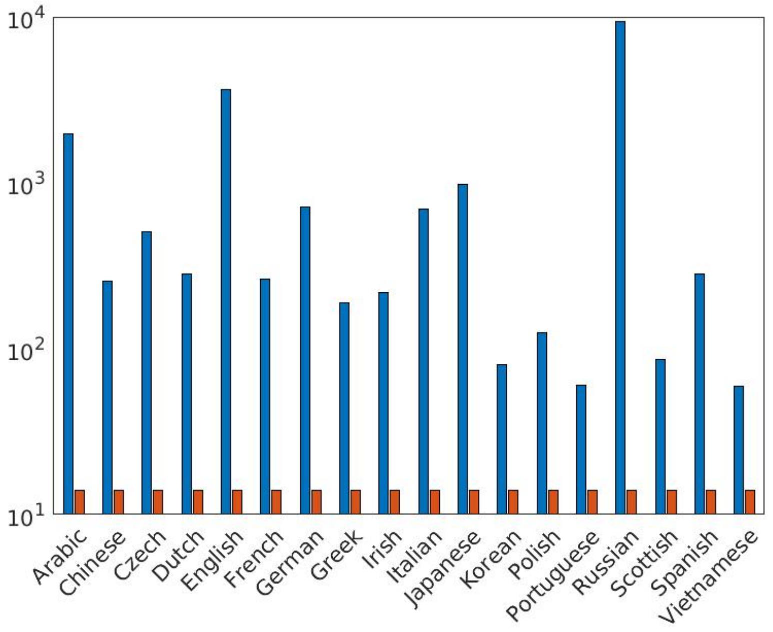
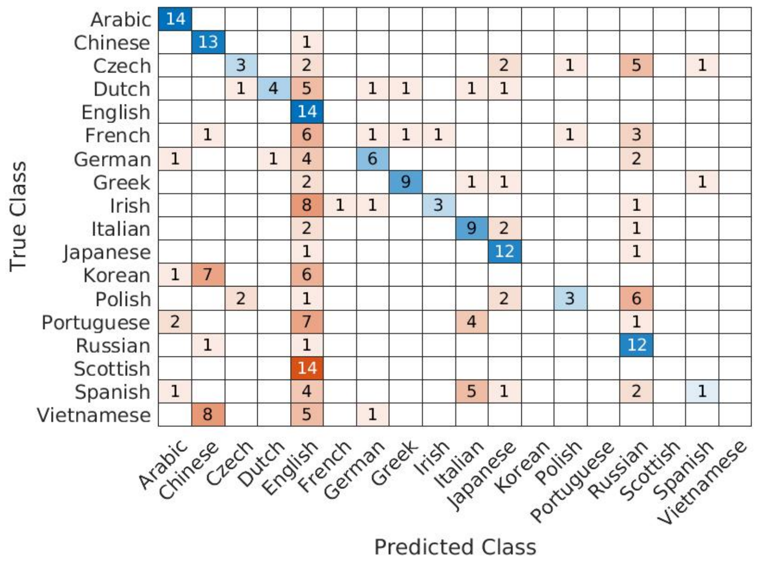
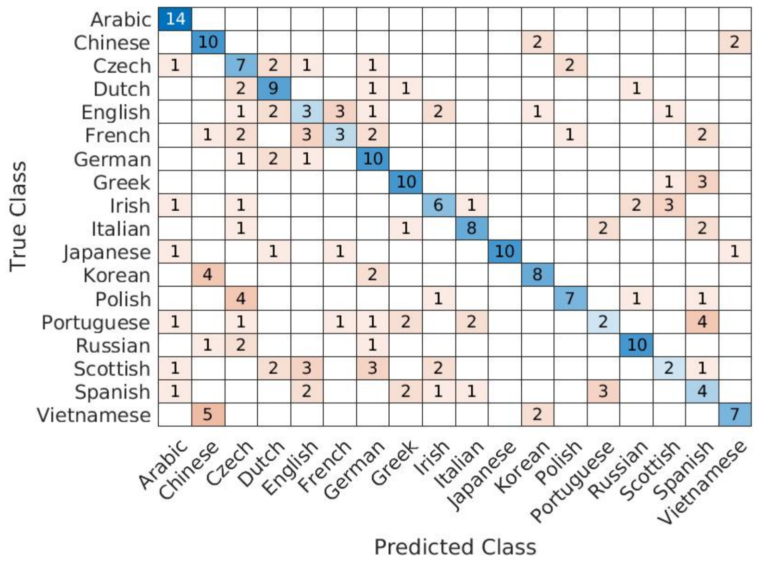


Figure 1: Bar plot of training (blue) and validation (red) sets. The y-axis is in log-scale.

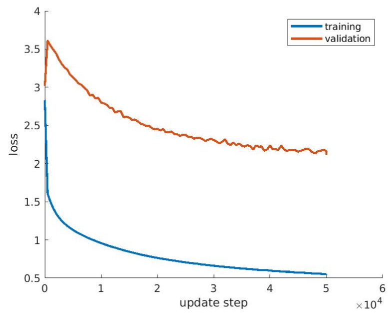


(a)

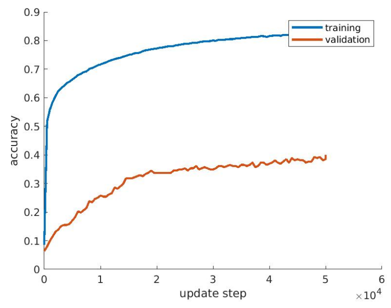


(b)

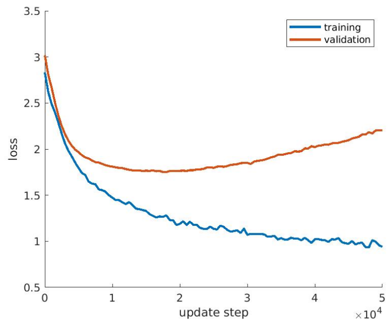
Figure 2: Confusion matrices on the validation set without (a) and with (b) compensation.



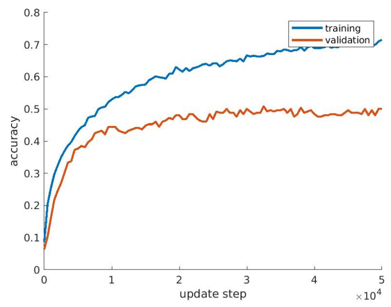
(a)



(b)



(c)



(d)

Figure 3: Learning curves without (a-b) and with (c-d) compensation.

the cost of fitting worse overall. The latter is true because the validation set is balanced.

The validation accuracy is 40.87% without and 51.59% with compensation. Again, the compensation in this case helps to increase the validation accuracy because the validation set is balanced (see figure 1), but this would not be true with an unbalanced test set (in such case, we would have to consider other metrics).

4 Final network

For the final network, following the advice in the instructions and searching for good parameter settings, I incremented the number of filters in both layers to 100. The details are listed in the following:

- Parameters: $k_1 = 5$, $n_1 = 100$, $k_2 = 3$, $n_2 = 100$.
- Hyper-parameters: $batch\_size = 100$, $\eta = 0.001$, 20000 updates (once again, using the validation set to keep a record of the best network parameters while training).
- Validation accuracy: 55.56%.
- Validation accuracy per class: see table 3.
- Learning curves: see figure 5.
- Confusion matrix: see figure 4.

Table 3 reports also the probabilities predicted for some new surnames. The network performs quite good for the given samples. Even though the prediction for *Lee* is wrong, the true class has a high probability.

Class	Val. acc.	Ruggeri	Salmeri	Lee	Scofield	Fdfr	Gonzales
Arabic	100.00	0.0002	0.0139	0.0010	0.0000	0.0040	0.0000
Chinese	92.86	0.0001	0.0000	0.3103	0.0000	0.0004	0.0000
Czech	35.71	0.0004	0.0290	0.0017	0.0004	0.1123	0.0021
Dutch	64.29	0.0050	0.0229	0.2415	0.0058	0.0262	0.0399
English	57.14	0.1008	0.0225	0.0094	0.7785	0.3394	0.0128
French	21.43	0.0625	0.0400	0.0197	0.0611	0.0193	0.0041
German	85.71	0.0104	0.0434	0.0062	0.0288	0.4107	0.0139
Greek	71.43	0.0012	0.0138	0.0006	0.0000	0.0046	0.0202
Irish	57.14	0.0015	0.0003	0.0014	0.1121	0.0016	0.0001
Italian	57.14	0.7985	0.7321	0.0022	0.0083	0.0473	0.1133
Japanese	78.57	0.0015	0.0005	0.0004	0.0000	0.0029	0.0001
Korean	57.14	0.0000	0.0000	0.3934	0.0000	0.0001	0.0000
Polish	50.00	0.0006	0.0452	0.0003	0.0000	0.0006	0.0001
Portuguese	14.29	0.0000	0.0000	0.0016	0.0000	0.0001	0.0008
Russian	78.57	0.0157	0.0294	0.0035	0.0003	0.0124	0.0033
Scottish	0.00	0.0014	0.0058	0.0001	0.0047	0.0023	0.0000
Spanish	28.57	0.0003	0.0012	0.0040	0.0001	0.0158	0.7893
Vietnamese	50.00	0.0000	0.0000	0.0028	0.0000	0.0000	0.0000

Table 3: Validation accuracy per class and probabilities for some surnames predicted by the final ConvNet. The probabilities for the true and predicted class are highlighted in yellow and red, respectively, if different, or green if equal.

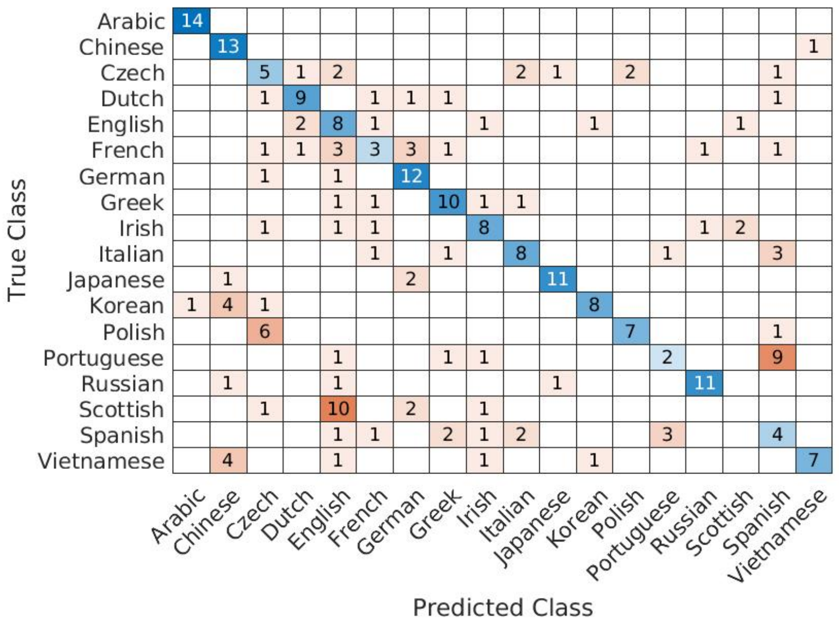


Figure 4: Confusion matrix on the validation set for the final ConvNet.

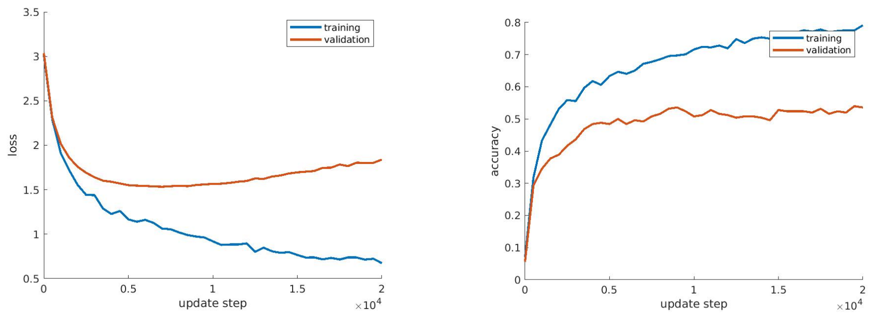


Figure 5: Learning curves for the final ConvNet.