Assignment 3 - DD2424

August Regnell 970712-9491

30 April, 2021

Contents

1 Introduction

In this assignment the task was to train and ConvNet to predict the language of surname from its spelling. The network was trained using mini-batch gradient descent with momentum applied to a loss function. The loss function was defined as sum of the cross-entropy loss of the classifier applied to the labelled training data divided by the size of the dataset.

2 Tests of analytical gradient

I managed to successfully write the functions to correctly compute the gradient analytically. I made sure of this by comparing the analytically computed gradient with the numerically computed gradient (code taken from canvas). More specifically, I compared with the given function *NumericalGradient*.

To make sure the computed gradients were correct, I checked that the relative error was small (`<1e-6`) using the following expression

$$r_e = \frac{|g_a - g_n|}{\max(\text{eps}, |g_a| - |g_n|)}$$

where g_a is the analytical gradient and g_n the numerical.
I tested for several different sizes of the dataset. The results can be seen in table 1. While the accuracies are not all 100%, they are close enough to be seen as correct. The errors are probably due to the inaccuracies of the numerical calculation.

#datapoints	$\mathbf{W}$	$\mathbf{F}_1$	$\mathbf{F}_2$
1	99.58%	100%	100%
100	95.69%	99.66%	100%
All	72.36%	87.24%	99.00%

Figure 1: Percentage of elements in the calculated gradients with a smaller relative error than `<1e-6`.

3 Compensating for imbalances in the dataset

I compensated for the unbalanced dataset in the following way: At the beginning of every epoch of training I randomly sampled the same number (the number of examples of the smallest class) of examples from each class and using that as the training set for the epoch.

4 Comparing results between training on the unbalanced and the balanced dataset

Graphs of the the loss and accuracy for the training and validation set as well as the final confusion matrix can be seen in figure 2 & 3 and 4 & 5, for the unbalanced and the balanced dataset respectively. The hyperparameters were $k_1 = 5$, $k_2 = 3$, $n_1 = 20$, $n_2 = 20$. The training was run for more than 20,000 update steps (the least number of epochs resulting in at least 20,00 update steps) and the learning parameters where $\eta = 0.001$ and $\rho = 0.9$. The values for the plots were recorded every 500th update step.

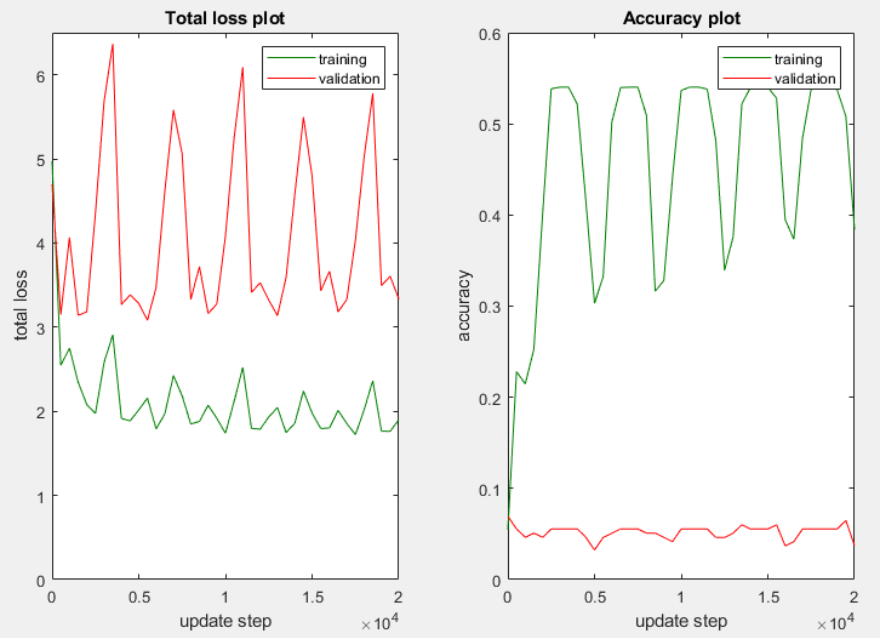

Figure 2: Loss and accuracy plots of the experiment on the unbalanced dataset.

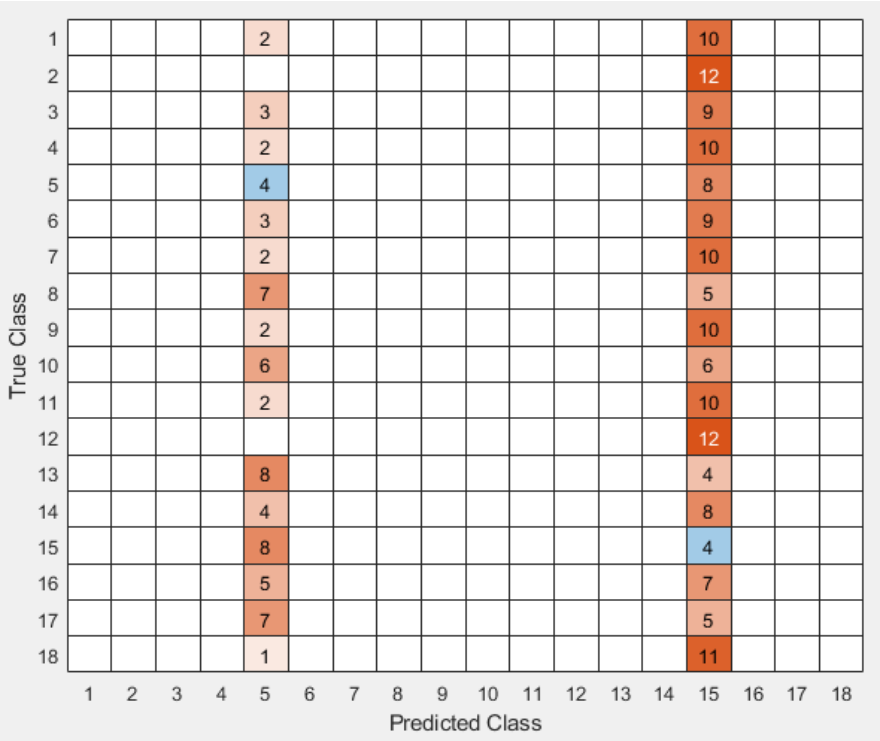

Figure 3: Final confusion matrix of the experiment on the unbalanced dataset.

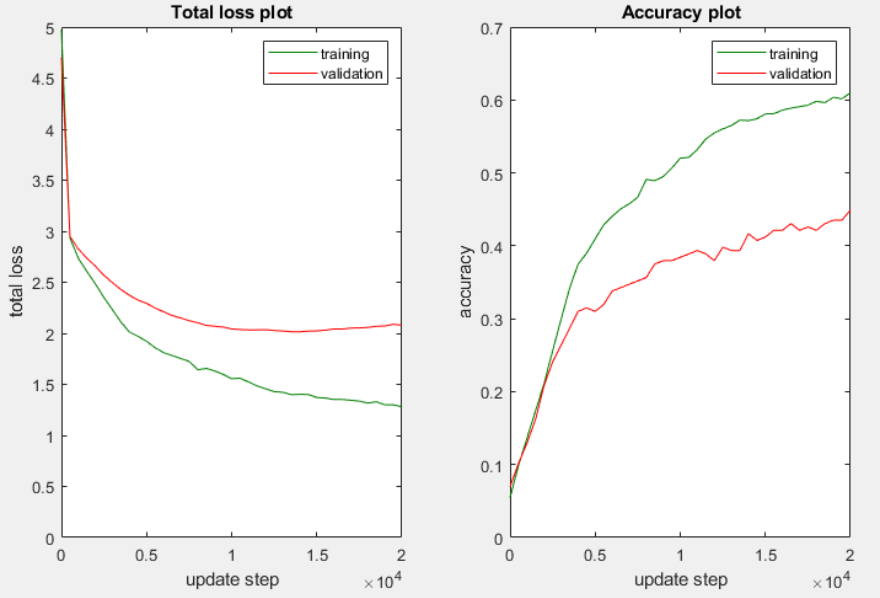

Figure 4: Loss and accuracy plots of the experiment on the balanced dataset.

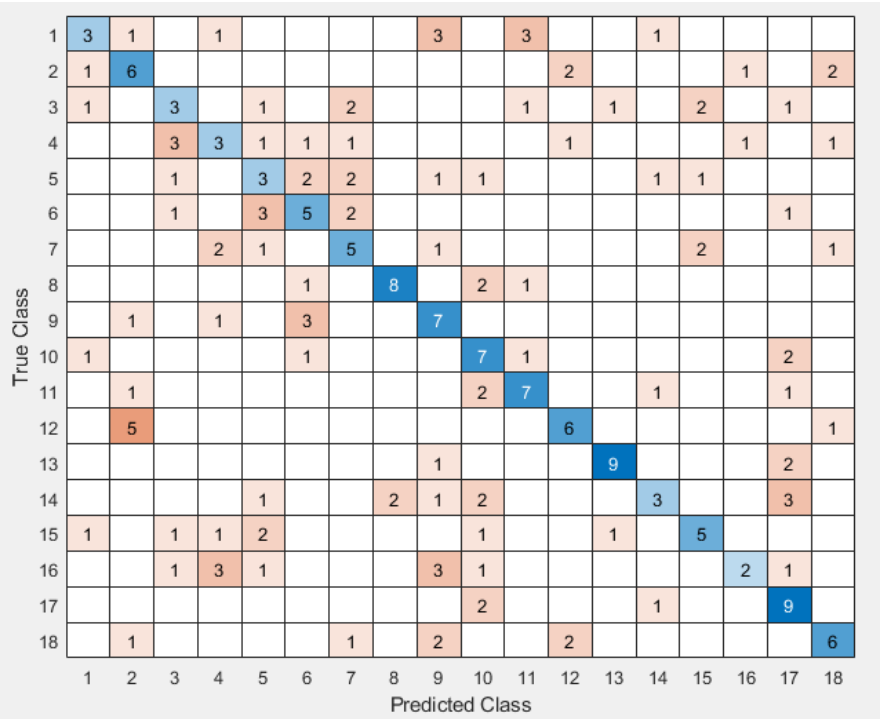

Figure 5: Final confusion matrix of the experiment on the balanced dataset.

Looking at the figures we clearly notice that the imbalance of the dataset in the first training affects the results. The first network always predicts the two classes with the most datapoints, which yields a very high accuracy on the training set. However, the accuracy on the validation set is much lower since it is balanced.

5 Best performing network

By doubling the number of filters at every layer I achieved my best performing network. Thus the hyperparameters were $k_1 = 5$, $k_2 = 3$, $n_1 = 20$, $n_2 = 20$ and the learning parameters were $\eta = 0.001$ and $\rho = 0.9$. The training was run for 20,016 update steps. Graphs of the the loss and accuracy for the training and validation set as well as the final confusion matrix can be seen in figure 6 & 7. We notice that the network starts to overfit at around 10,000 update steps and that the best validation accuracy was achieved at around 15,000 update steps. We would thus expect a even better performing network if we had stopped the training earlier.

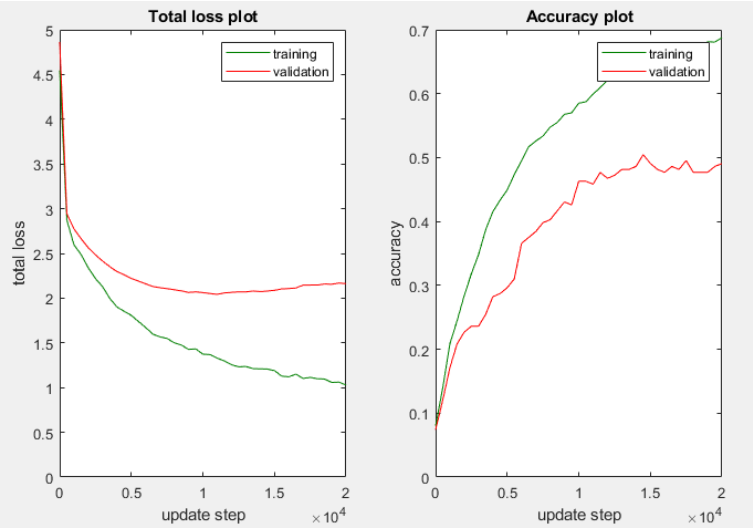

Figure 6: Loss and accuracy plots of the experiment on the best network.

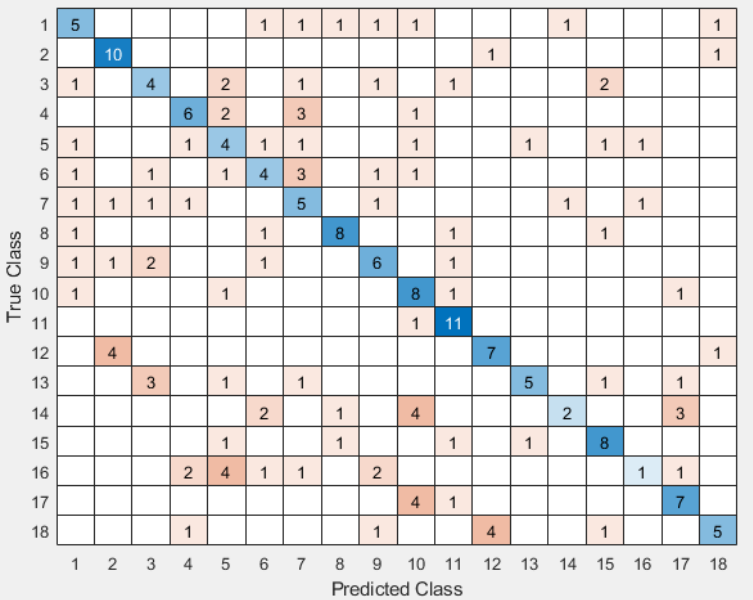

Figure 7: Final confusion matrix of the experiment on the best network.

The final validation accuracy was 47.69%, which probably could have been increased to ∼50% with a shortened training. The accuracies per class can be seen in figure 8.

Arabic	**Chinese**	**Czech**	**Dutch**	**English**	**French**	**German**	**Greek**	**Irish**
41.67%	83.33%	33.33%	50.00%	33.33%	33.33%	41.67%	66.67%	50.00%
Italian	**Japanese**	**Korean**	**Polish**	**Portuguese**	**Russian**	**Scottish**	**Spanish**	**Vietnamese**
58.33%	83.33%	58.33%	41.67%	16.67%	66.67%	8.33%	50.00%	41.67%

Figure 8: Accuracies per class for the best network

6 Efficiency improvements

I implemented the first improvement suggested in *Background 5*, that is pre-computing the $M^{input}_{x_j,k_1,n_1}$'s and making them spare. Since each $M^{input}_{x_j,k_1,n_1}$ only depends on the input data, it is unnecessary to re-compute them every batch.

To measure the efficiency gains I trained a network with the following hyper parameters, $k_1 = 5$, $k_2 = 3$, $n_1 = 20$, for 100 update steps and with batch size 39. The pre-computing itself took 37.5 seconds. The 100 update steps took 11.7 seconds when the $M^{input}_{x_j,k_1,n_1}$'s were pre-computed and sparse, and 182.5 seconds when they were not. For this case we thus get an improvement of 1.7 seconds per update step, which is a massive improvement that makes the time of the pre-computing insignificant.

7 Prediction on friends' surnames

	Regnell	**Haraldsson**	**Logren**	**Zhou**	**Kerakos**	**Ivinskiy**
Arabic	0.0000	0.0001	0.0006	0.0074	0.0000	0.0000
Chinese	0.0000	0.0000	0.0000	0.4475	0.0000	0.0000
Czech	0.0013	0.0238	0.0042	0.0003	0.0047	0.0063
Dutch	0.0172	0.0248	0.7213	0.0001	0.0004	0.0001
English	0.0670	0.2111	0.0445	0.0015	0.0001	0.0014
French	0.0196	0.1976	0.0898	0.0220	0.0000	0.0000
German	0.0091	0.0251	0.0709	0.0005	0.0000	0.0000
Greek	0.0000	0.0017	0.0001	0.0000	0.9895	0.0019
Irish	0.8808	0.0004	0.0586	0.0000	0.0000	0.0001
Italian	0.0001	0.0380	0.0006	0.0010	0.0000	0.0001
Japanese	0.0000	0.0076	0.0000	0.0007	0.0020	0.0241
Korean	0.0000	0.0000	0.0000	0.0810	0.0000	0.0000
Polish	0.0008	0.0001	0.0013	0.0000	0.0009	0.0626
Portuguese	0.0000	0.0041	0.0000	0.0018	0.0000	0.0000
Russian	0.0001	0.3643	0.0069	0.0017	0.0024	0.9035
Scottish	0.0007	0.0006	0.0002	0.0001	0.0000	0.0000
Spanish	0.0033	0.1008	0.0010	0.0006	0.0000	0.0000
Vietnamese	0.0000	0.0000	0.0000	0.4337	0.0000	0.0000

Figure 9: Table of probabilities of some friends' surname's classes.

Since the training data doesn't contain Swedish surnames it isn't surprising that the network's guesses on the first three surnames aren't very good. However, looking at the last two surnames we

notice that it is very precise, giving more then 90% probability to the right nationalities. Furthermore, the network seems to be split on Chinese and Vietnamese for Zhou, which isn't surprising since the two countries share some surnames. It does however predict the correct nationality for Zhou - Chinese.