DD 2424: Assignment 2 option 3

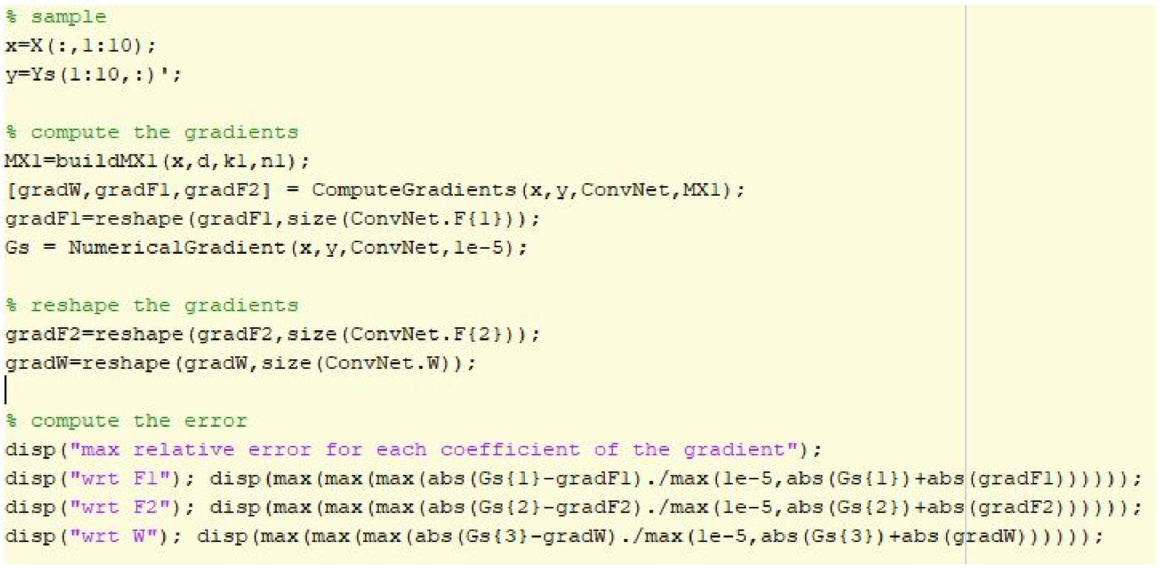
Clément Gousseau

June 30, 2018

1 Checking the gradients

To check my gradients' calculations I computed the gradients on the first 10 inputs of the given datasets, with the function I implemented (analytical gradients) and the function provided (numerical gradients, with h=1e-5).

Then for each coefficient I calculated the quantity $\frac{|g_a - g_n|}{max(1e{-}5, |g_a| + |g_n|)}$ where a refers to the analytical gradient and n to the numerical gradient. Then I took the maximum of these quantities for each layer.



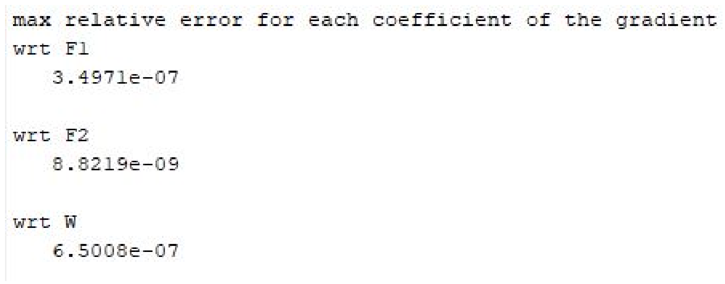
```
% sample
x=X(:,1:10);
y=Ys(1:10,:)';

% compute the gradients
MX1=buildMX1(x,d,k1,n1);
[gradW,gradF1,gradF2] = ComputeGradients(x,y,ConvNet,MX1);
gradF1=reshape(gradF1,size(ConvNet.F{1}));
Gs = NumericalGradient(x,y,ConvNet,1e-5);

% reshape the gradients
gradF2=reshape(gradF2,size(ConvNet.F{2}));
gradW=reshape(gradW,size(ConvNet.W));

% compute the error
disp("max relative error for each coefficient of the gradient");
disp("wrt F1"); disp(max(max(max(abs(Gs{1}-gradF1)./max(1e-5,abs(Gs{1})+abs(gradF1))))));
disp("wrt F2"); disp(max(max(max(abs(Gs{2}-gradF2)./max(1e-5,abs(Gs{2})+abs(gradF2))))));
disp("wrt W"); disp(max(max(max(abs(Gs{3}-gradW)./max(1e-5,abs(Gs{3})+abs(gradW))))));
```

The resulting errors were very small so I considered my gradient computations were bug free.



```
max relative error for each coefficient of the gradient
wrt F1
   3.4971e-07

wrt F2
   8.8219e-09

wrt W
   6.5008e-07
```

2 Balanced/Unbalanced datasets

I implemented the second option that was mentioned in the text. For each update, I created a sample from the training set. In each of these samples, there are the same number of inputs of each class. In total there are *n_ batch* inputs in the sample, with *n_ batch* a multiple of 18 (number of classes).

Figure 1 shows a training with 25000 updates, n1=n2=20, k1=5, k2=3, eta=0.001, rho=0.9. Looking at the loss (figure 1), the training is much better with unbalanced data. The final accuracy is 75.4% for the unbalanced training and 59.0% for the balanced training.

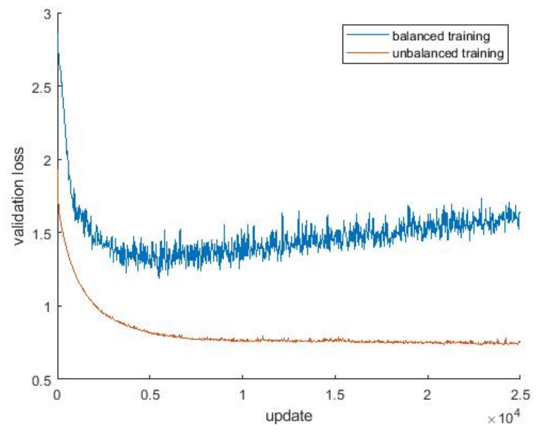
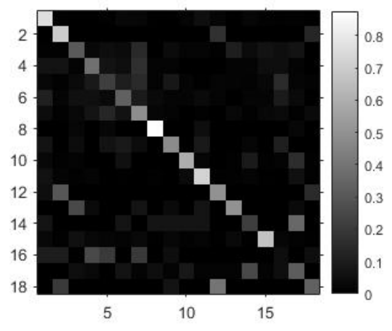
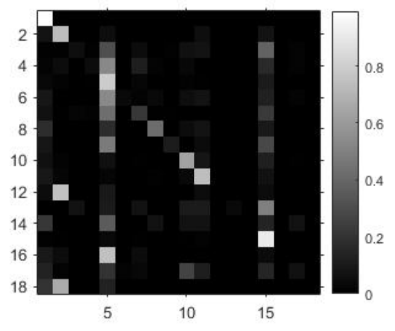


Figure 1: Evolution of the loss with balanced and unbalanced training

However the conclusion is different when we look at the confusion matrix. The matrix for the balanced training has in average higher values on the diagonal (which corresponds to the well classified inputs).



(a) with balanced data.



(b) without balanced data

Figure 2: Confusion matrix on the test set.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
balanced	76	70	31	38	22	33	49	88	48	59	72	50	50	21	67	5	32	33
unbalanced	99	73	6	5	79	4	22	43	11	63	74	0	4	0	91	0	7	0
#elements	400	53	104	60	734	54	146	40	46	142	198	20	28	14	1876	20	60	15

Table 1: For each class, accuracy with balanced training (%), accuracy with unbalanced training (%), number of elements

The balanced training is for most of the classes better but the unbalanced training has very good performance on the most prominent class. For example, there is an accuracy higher than 90% for the class 1 and 15, which represent more than 50% of the dataset.

So there is not a best training technique. Unbalanced training enables a low loss on the whole dataset. Balance training enables a better accuracy for most of the classes.

3 Best performing ConvNet

I tried different settings to find the best performing network. All these settings have been trained through 25000 updates. I choose unbalanced training since it gave better accuracy. The first layer includes filter of width 5, the second layer filters of width 3.

n1=n2 \ eta	0.0001	0.001	0.01	0.1
5	51.2	70.3	76.7	74.4
10	53.3	73.5	78.8	77.6
20	58.9	75.4	76.5	78.1

Table 2: Accuracy for the different settings

So the best results were obtained with 10 filters at each layer and a learning rate of 0.01. The corresponding final accuracy is 78.8%.

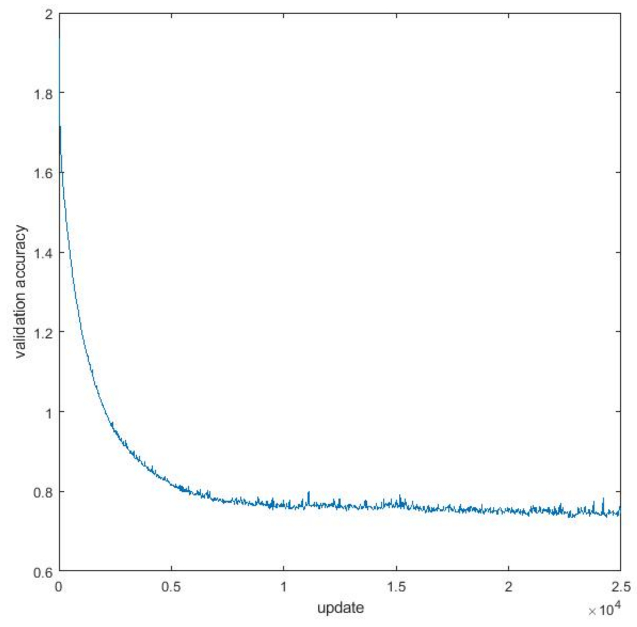


Figure 3: Evolution of the loss for the best network

class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
accuracy (%)	100	66	16	20	80	13	34	48	17	70	86	15	25	0	92	0	20	0

Table 3: Accuracy for each class for the best performing network

4 Efficiency gains

I first implemented a version without any specific efficiency gain. For a network with n1=20, k1=5, n2=20, k2=5 the training took 753.4s for 100 updates. Then I used sparse matrices for the matrices with many zeros (input data, MX-matrices, MF-matrices), the training time dropped at 33.1s. Then I precomputed the matrix MX1 and the training took 11.8s. These 11.8s do not include the pre-computation of MX1 which took 544s.

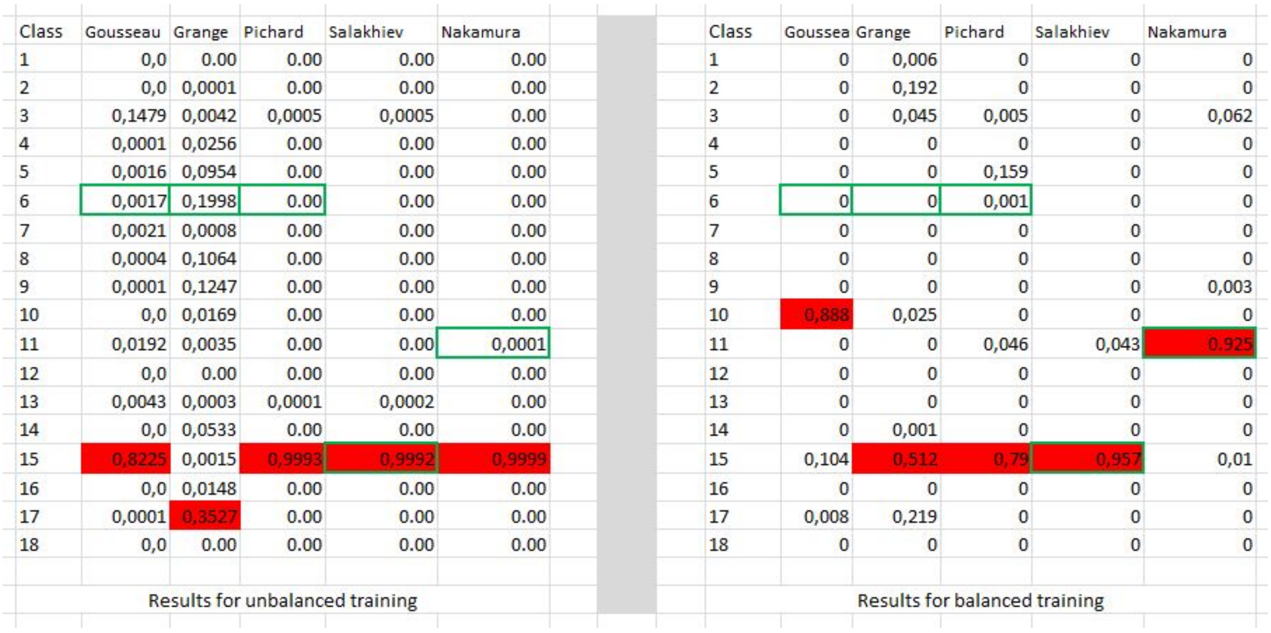
So for a long training (25000 updates):

the first version would have taken 25000*753.4/100=188350s (52h)
the second version would have taken 25000*33.1/100=8275s (2.3h)
the third version would have taken 544+25000*11.8/100=3494s (1h)

5 Let's have a try

I tested my best network with 5 names: Gousseau, Grange, Pichard are French names, Salakhiev is Russian and Nakamura is Japanese. In the table below, the red value corresponds to the highest value, and the green box corresponds to the value that should be the highest.

For the unbalanced training, only the Russian name is well classified. For the balanced training, the Russian and Japanese name are well classified. The French names are never well classified. It is not surprising since this class is not prominent in the training set.



Class	Gousseau	Grange	Pichard	Salakhiev	Nakamura
1	0,0	0.00	0.00	0.00	0.00
2	0,0	0,0001	0.00	0.00	0.00
3	0,1479	0,0042	0,0005	0,0005	0.00
4	0,0001	0,0256	0.00	0.00	0.00
5	0,0016	0,0954	0.00	0.00	0.00
6	0,0017	0,1998	0.00	0.00	0.00
7	0,0021	0,0008	0.00	0.00	0.00
8	0,0004	0,1064	0.00	0.00	0.00
9	0,0001	0,1247	0.00	0.00	0.00
10	0,0	0,0169	0.00	0.00	0.00
11	0,0192	0,0035	0.00	0.00	0,0001
12	0,0	0.00	0.00	0.00	0.00
13	0,0043	0,0003	0,0001	0,0002	0.00
14	0,0	0,0533	0.00	0.00	0.00
15	0,8225	0,0015	0,9993	0,9992	0,9999
16	0,0	0,0148	0.00	0.00	0.00
17	0,0001	0,3527	0.00	0.00	0.00
18	0,0	0.00	0.00	0.00	0.00

Results for unbalanced training

Class	Goussea	Grange	Pichard	Salakhiev	Nakamura
1	0	0,006	0	0	0
2	0	0,192	0	0	0
3	0	0,045	0,005	0	0,062
4	0	0	0	0	0
5	0	0	0,159	0	0
6	0	0	0,001	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0,003
10	0,888	0,025	0	0	0
11	0	0	0,046	0,043	0.925
12	0	0	0	0	0
13	0	0	0	0	0
14	0	0,001	0	0	0
15	0,104	0,512	0,79	0,957	0,01
16	0	0	0	0	0
17	0,008	0,219	0	0	0
18	0	0	0	0	0

Results for balanced training

Figure 4: Results for balanced and unbalanced training