DD2424 A3

Harsha HN harshahn@kth.se

1. State how you checked your analytic gradient computations and whether you think that your gradient computations are bug free for your ConvNet.

   Gradient for the network parameters gradConvNet: *F1, F2* and *W* were implemented based on the analytical solution. Correctness of the computed analytical gradients is validated against numerically computed gradients measured in terms of relative error. Relative error is computed from the *rerr(ga, gn)* function. *ga*: Analytical gradient & *gn*: Numerical gradient.

   $$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)}$$ where eps is a small number;

   Results were obtained as follows:
   For ConvNet filter size at both layers = 5, h = 1e-5;

Relative error	$\sum_{1}^{5}$ F1	$\sum_{1}^{5}$ F2	W
Batchsize = 2	2.54e-11	5.90e-10	6.61e-09
Batchsize = 18	6.89e-10	3.99e-09	1.05e-07
Batchsize = 118	2.07e-04	9.21e-06	2.11e-07
Batchsize = 1062	7.77e-05	7.48e-08	1.87e-06

   Relative error values are very low. Thus, the correctness of the implementation of analytical gradient is ensured. Even for large values of batch size the relative error remained low.

DD2424 A3

Harsha HN harshahn@kth.se

2. Comment on how you compensated for the imbalanced dataset
   Method: implemented the other strategy of normalizing loss and gradient w.r.t class using the variable *py*. Reason for this strategy is that we have large data samples for each epoch and also, the problem of imbalance is countered by simple weight approach. Also, implemented randomly sampled 59 (number of examples from the smallest class) from each of the 18 classes and this becomes the effective training set for a particular epoch. Thus, 59*18 = 1062 names were picked for an epoch. But commented.

3. Include a graph of the validation loss, for a longish training 20,000 update steps for a network with k1 = 5; k2 = 3 and n1 = 20, n2 = 20 when you train without compensating for the unbalanced dataset and when you do. Include also the final confusion matrix on the validation set in both cases. Comment on the qualitative differences between the two different training regimes.
   For GDparams as follows: Number of epochs = 20, batch size = 19, batches = 1042, GDparams.eta = 0.001; GDparams.rho = 0.9. Total updates = 20840.

   a. Unbalanced dataset (258s)

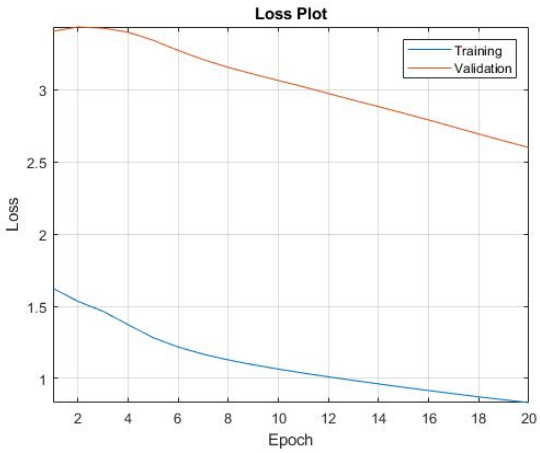

Figure a
Loss

   b. Balanced dataset (240s)

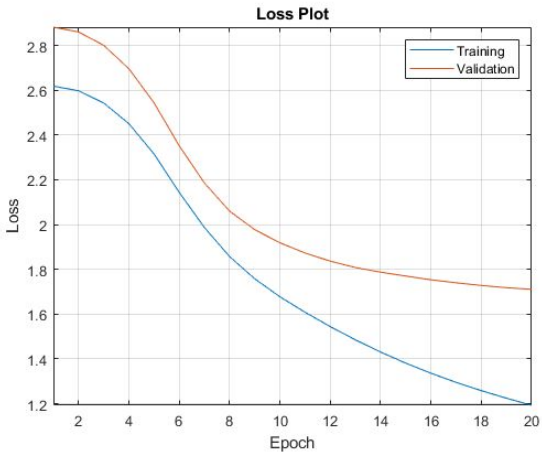

Figure b
Loss

DD2424 A3

Harsha HN harshahn@kth.se

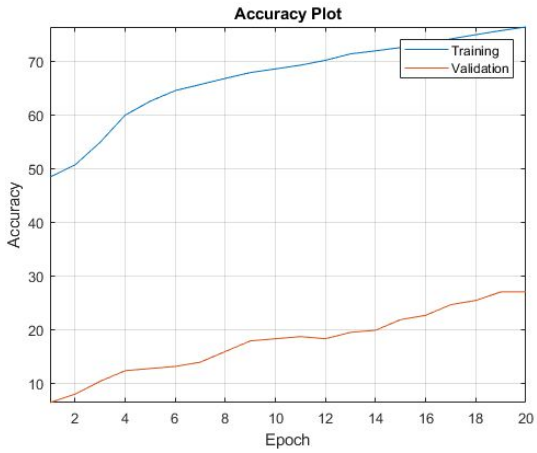

Figure c
Accuracy

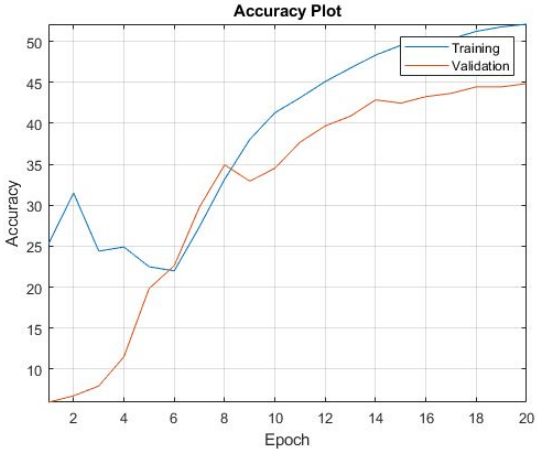

Figure d
Accuracy

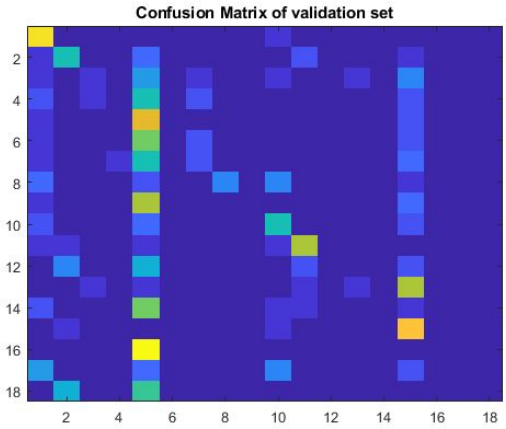

Figure e
Confusion chart

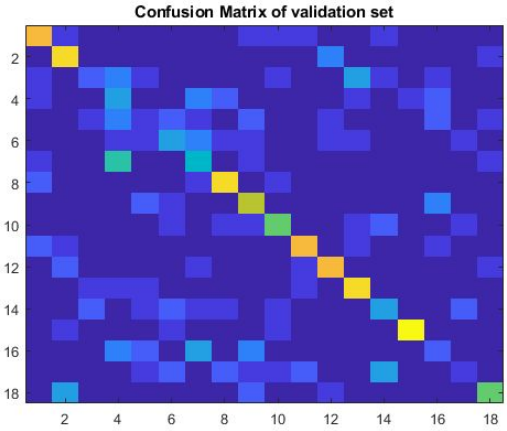

Figure f
Confusion chart

Firstly, the loss plot, we observe consistent gap between validation and training set which indicates lack of regularization. Evidently, the imbalanced dataset has led to skewed parameters' tuning, thus, performing poorly for the classes which had lesser samples. After normalizing each loss and gradient, we observe the gap has significantly reduced offering better regularization.

Secondly, the accuracy plot, similar observation can be noted. Further, over-tuning of parameters has led to accuracy of >70% on training set, consequently performing poorly <30% for validation set. In the balanced regularization, although it has brought down the accuracy on training, it is giving consistent accuracy of > 45% for both.

Finally, the confusion matrix, we observe that the trained network predicting the selected classes 5 and 15 which had the most samples in the training set leading to biasing tuned. However, with balancing dataset, we see consistent distribution which is better.

DD2424 A3

Harsha HN harshahn@kth.se

4. Include the validation loss function, for your best performing ConvNet. State the final validation accuracy of this network and its accuracy for each class. Detail the parameter settings and how trained the network and the training parameters.

$$L_{upsampled}(trainX, F_1, F_2, W) = \frac{1}{|19|} \sum_{train(X,y) \in ß} p_y \, l(train(X,y), F_1, F_2, W)$$

ConvNet: F1(28,5,30), F2(30,5,30), W(18,330) | $train(X,y) \in ß$, Total size = 19798. Achieved validation accuracy of 51.19% for parameter settings as follows:
n1 = 30; k1 = 5; n2 = 30; k2 = 5; eta = 0.001; rho = 0.9, Epochs = 29, Batch size = 19, Batches = 1042, Num of update steps = 30218, Training time = 567 seconds. Balanced data set using normalize strategy.

Class Precision	**1** 71.43%	**2** 52.38%	**3** 42.86%
4 53.33%	**5** 30.77%	**6** 35.71%	**7** 50.00%
8 55.56%	**9** 39.13%	**10** 42.11%	**11** 73.34%
12 66.67%	**13** 78.57%	**14** 12.50%	**15** 81.82%
16 33.33%	**17** 20.00%	**18** 60.00%	**49.97%**

5. State whether you implemented the efficiency gains in Background 5 and by how much they helped speed up training. You can quote how long it took to perform a fixed number of update steps (100) with mini-batches of size 100 before and after the upgrade.
   Yes, both the optimizations were made. For 100 updates involving mini-batch size of 521, 38 batches, 3 epochs.
   a. Optimized: 35 seconds in total.
   b. Without optimization: 2700 seconds in total.
      Profiler info shows that *VXtoMX, kron, MakeMXMatrix* consumed 75% of the overall time load.Thus, we achieved significant speed-up in training!

DD2424 A3

Harsha HN harshahn@kth.se

6. Show the probability vector output by your best network when applied to your surname and those of 5 of your friends. Comment on the results!

testX	huang	maria	akeel	andreas	gonzales	woods
testY	2	15	1	7	17	5
Ypred	2	17	4	8	17	5
Prob						
Top 1	02@ 0.46	17@ 0.43	04@ 0.23	08@ 0.25	17@0.64	05@0.3
Top 2	18@ 0.34	10@ 0.21	05@ 0.18	04@0.18	08@0.11	16@0.19
Top 3	12@ 0.18	06@ 0.12	03@ 0.11	14@0.17	06@0.08	03@.11

Surprisingly, the CNN has got 50% of the names rightly at significant confidence/probability value. Interestingly, the wrong predictions had minimal probability allocation towards the right class, however, often the probability values for the classes are close enough depicting the uncertainty and diffidence for its prediction.

Note: This assignment took incredibly lot of time and intellectual efforts.

Thank you!