Assignment 3 - option 1 - BONUS - DD2424

Alexandra Hotti

May 2019

1 Introduction

In this report 7 different methods to improve the results of a k-layer network with batch normalization are implemented. Finally a network combining several of the methods is implemented.

2 (a) - A more exhaustive random search to set the amount of regularization

Below are the results from a grid search performed to set the amount of regularization for the 3-layer network with batch normalization from the basic part of the assignment. In the basic part of the assignment this network achieved a test set accuracy of 53.64%.

Moreover, this network is trained using a cyclical learning and after each epoch the data points were shuffled. The network was initialized using HE-initialization. The following parameter setting was used for 2 cycles of training:

$$epochs = 20, \lambda = 0.005, n_s = 2250, m_1 = 50, m_2 = 50, n_{cycles} = 2$$

Were m_1 and m_2 are the number of hidden nodes in the first and second layers.

2.1 Coarse Search

The λ values were sampled during the coarse search using a broader range and a larger number of runs of 40 compared to in the basic part of the assignment. The following setting was used:

$$l = l_{min} + (l_{max} - l_{min}) \cdot rand(1,1)$$
$$\lambda = 10^l, l_{min} = -10, l_{max} = -1$$

Table 1: Coarse λ grid search.

Run	λ	Test set Accuracy
1	1.086384902337103e-12	52.25 %
2	5.445546995868439e-10	52.41 %
3	2.001428288314980e-10	52.52 %
4	0.002556854087961	**53.93** %
5	4.245813231864976e-07	52.54 %
6	0.015463150956827	52.66 %
7	0.020530541312151	51.93 %
8	4.384325852848999e-10	52.35 %
9	9.547265161850072e-06	53.08 %
10	1.496598309916386e-12	53.11 %
11	6.111227466032562e-08	52.49 %
12	7.372648328588670e-09	52.89 %
13	8.647180868038567e-08	52.65 %
14	3.910625692063962e-05	52.95 %
15	0.014469117905438	52.80 %
16	1.683346043195703e-04	53.25 %
17	3.249559199252779e-13	52.56 %
18	3.932897753151636e-05	52.79 %
19	8.173711466292847e-07	53.10 %
20	1.785593440943595e-13	52.75 %
21	4.235778632342755e-13	53.20 %
22	8.682009364127300e-08	52.31 %
23	2.939684903529485e-04	52.61 %
24	1.694079046917877e-13	52.71 %
25	1.061254192754272e-08	53.41 %
26	3.936920083204932e-10	52.85 %
27	2.298065636416008e-13	52.97 %
28	0.030040952494308	51.34 %
29	0.020090338465906	52.59 %
30	3.581953098862835e-09	53.10 %
31	8.736903471902508e-07	52.66 %
32	1.436099370550052e-14	53.01 %
33	5.911839661424570e-05	52.42 %
34	1.479426143311886e-04	52.96 %
35	1.470514050360108e-09	52.71 %
36	1.263307240386780e-06	52.63 %
37	6.600345885536658e-15	52.91 %
38	2.994022450031828e-11	53.08 %
39	5.874594118270877e-15	53.39 %
40	7.391864658917797e-08	52.64 %

2.2 Fine Search

The coarse search range was set using the best performing λ in the coarse search for run 4 in table 1. Which corresponds to a l value of -2.592294055. Next a finer search was performed using this value to set the upper and lower range for $l +/- 0.1$ i.e: $l_{min} = -2.692294055, l_{max} =$

-2.492294055.

Table 2: Finer λ grid search.

Run	λ	Test set Accuracy
1	0.002452728034992	53.63 %
2	0.002417905783433	53.68 %
3	0.003054631929675	53.77 %
4	0.002697506681627	53.22 %
5	0.003134183989792	53.68 %
6	0.003146901095234	53.59 %
7	0.002445144653046	53.25 %
8	0.002820172314203	53.94 %
9	0.002254574503941	52.96 %
10	0.002623833975837	52.95 %
11	0.002545744724213	53.48 %
12	0.002636876969129	53.41 %
13	0.002877555764044	53.59 %
14	0.003131210467042	53.24 %
15	0.002938189827127	53.20 %
16	0.002205916918018	53.76 %
17	0.002877789230074	53.77 %
18	0.002722865697557	53.49 %
19	0.002187128337739	53.19 %
20	0.002214285228775	53.33 %
21	0.002637028391949	52.74 %
22	0.002961684642175	53.73 %
23	0.002185485123362	53.22 %
24	0.002559026562438	53.65 %
25	0.002441387707542	53.18 %
26	0.002195026127219	53.56 %
27	0.003164060012395	53.61 %
28	0.003145926849895	53.41 %
29	0.002519626858832	52.96 %
30	0.002725458817587	53.74 %
31	0.002109779907595	53.14 %
32	0.002894594263555	53.25 %
33	0.002810001983332	53.64 %
34	0.002932774734710	53.48 %
35	0.002487783869334	53.00 %
36	0.002739854510798	**54.02 %**
37	0.002086479108871	53.48 %
38	0.002353165354915	53.45 %
39	0.002083009943393	53.39 %
40	0.002630974970691	52.85 %

2.3 Results

The regularization term λ resulting in the highest test set accuracy of 54.02% was equal to: 0.002739854510798. This was 0.38 percentage points higher than the highest accuracy achieved using the default regularization term in the basic part of the assignment.

3 (b) - A deeper Network

Here the performance of the network was attempted to be increased by increasing the number of layers and changing the number of hidden nodes in the networks. Each network was trained for 2 cycles, had batch normalization, had HE-initialization and the data points were shuffled after each epoch of training. The following common parameter were used:

$$epochs = 20, \lambda = 0.005, n_s = 2250, n_{cycles} = 2$$

3.1 Results

The different architectures implemented and their corresponding test set accuracies can be found in table 3.

Table 3: Architectures tried

Number of layers	Hidden layers	Test set Accuracy
3	[50, 50]	53.64 %
4	[50, 50, 30]	54.07 %
4	[50, 50, 40]	54.16 %
4	[60, 50, 40]	**55.28 %**
4	[70, 50, 40]	54.78 %
4	[60, 60, 40]	54.56 %
4	[60, 50, 50]	54.26 %
5	[50, 50, 30, 25]	53.60 %
5	[50, 50, 30, 20]	53.55 %
5	[60, 50, 40, 20]	54.77 %
5	[60, 50, 40, 10]	55.10 %

3.2 Discussion

Increasing the number of hidden nodes and layers allows the network to fit its parameters more closely to the data and increases the training accuracy. However, this could result in overfitting to the training data set such that the performance on unseen data worsens and the gap between the validation and training data loss and accuracy plots increase. Therefore, it was not the most complex network which achieved the highest accuracy on the unseen test set data. Instead the best performing network had 4 layers and achieved a test set accuracy of **55.28 %**. Also, as the complexity of the network increases it seems appropriate to increase the amount of regularization. Therefore, adding dropout to the network activations is explored in the next section.

4 (d) - Dropout

Here the performance of the network was attempted to be increased by adding dropout to several relatively deep networks. Each network was trained for 2 cycles, had batch normalization, had HE-initialization and the data points were shuffled after each epoch of training. The following common parameter were used:

$$epochs = 20, \lambda = 0.005, n_s = 2250, n_{cycles} = 2$$

Inverted dropout was implemented during the training of the network using the following implementation:

$$\begin{aligned} x^l &= max(0, W_l \cdot X^{l-1} + b_l) \\ u^l &= (rand(size(x^l)) < p)/p \\ x^l &= x^l \odot u^l \end{aligned}$$

4.1 Results

The different architectures implemented and the dropouts used can be found in table 4.

Table 4: Architectures with dropout

Number of layers	Hidden layers	Drop Out	Test set Accuracy
4	[60, 50, 40]	0.001	54.19 %
5	[60, 50, 40, 10]	0.001	54.53%
6	[60, 50, 40, 10, 10]	0.001	54.23%
6	[60, 50, 40, 10, 10]	0.01	53.29 %
6	[70, 60, 50, 30, 20]	0.01	54.31 %
6	[70, 70, 50, 30, 20]	0.01	54.38 %
9	[50, 30, 20, 20, 10, 10, 10, 10]	0.01	48.83 %

4.2 Discussion

Adding dropout to the network did not improve its performance. A potential explanation could be that it was combined with batch normalization [2].

5 One Sided Label Smoothing

A method used for regularization commonly used when training Generative Adversarial Networks is one sided label smoothing [3]. Thus, in this multi-classification task instead of using ordinary one-hot encoding a value is uniformly sampled between a range, such as $[0.9, 1.0]$, for each label encoding. The purpose being to reduce the confidence of the classifier. The implementation can be found in the function: *ComputeGradients*.

The hyperparameters used in this section of the assignment are found below. Also HE-initialization, batch normalization and shuffling after every epoch was used. For reference, the 3 layer network optimized in this assignment achieved a test accuracy of 53.64% and the 9-layer network achieved 51.58% in the basic part of the assignment.

$$\lambda = 0.005, n_s = 2250$$

5.1 Results

The results were label smoothing was used during training can be found in table 5.

Table 5: Label smoothing results

Sampling Range	Hidden Layers	n_{cycles}	Test set Accuracy
$[0.9, 1.0]$	[50, 50]	2	49.07 %
$[0.95, 1.0]$	[50, 50]	2	53.43 %
$[0.9, 1.0]$	[50, 30, 20, 20, 10, 10, 10, 10]	2	47.65 %
$[0.95, 1.0]$	[50, 30, 20, 20, 10, 10, 10, 10]	2	**52.51 %**
$[0.9, 1.0]$	[50, 50]	3	49.76 %
$[0.95, 1.0]$	[50, 50]	3	52.72 %
$[0.9, 1.0]$	[50, 30, 20, 20, 10, 10, 10, 10]	3	46.61 %
$[0.95, 1.0]$	[50, 30, 20, 20, 10, 10, 10, 10]	3	**53.07 %**

5.2 Discussion

The results from using one-sided label smoothing, found in table 5 show that the final test set accuracy worsened for the 3-layer network. This was also the case for the 9-layer network when the labels were smoothed within the range $[0.9, 1.0]$. However, when a smaller range of $[0.95, 1.0]$ was used to smooth the training images the results improved. When the network was trained for 2 cycles the accuracy improved by almost 1 percentage point. From 51.58% to 52.51%. Also, by increasing the number of cycles trained into 3 cycles, the test accuracy increased to 53.07%.

6 (e) Augmenting training data by applying random jitter

Next, two different networks were trained with augmented training data. This was performed by uniformly sampling a factor to either increase or decrease each element in the training data within a certain range given in the table 6. This implementation can be found in the function *applyJitter*. For reference, the 3 layer network optimized in this assignment achieved a test accuracy of 53.64% and the 9-layer network achieved 51.58% in the basic part of the assignment. The following common parameters were used:

$$\lambda = 0.005, n_s = 2250, n_{cycles} = 2$$

6.1 Results

The results were jitter was applied to the training data can be found in table 6.

Table 6: Results from augmenting training data with random jitter.

Sampling Range	Hidden Layers	Test set Accuracy
$+/-1.5\%$	[50, 50]	53.61 %
$+/-2\%$	[50, 50]	**53.77 %**
$+/-2.5\%$	[50, 50]	53.38%
$+/-1.5\%$	[50, 30, 20, 20, 10, 10, 10, 10]	51.67 %
$+/-2\%$	[50, 30, 20, 20, 10, 10, 10, 10]	51.75 %
$+/-2.5\%$	[50, 30, 20, 20, 10, 10, 10, 10]	52.10 %
$+/-3\%$	[50, 30, 20, 20, 10, 10, 10, 10]	**52.59 %**
$+/-3.5\%$	[50, 30, 20, 20, 10, 10, 10, 10]	52.26 %

6.2 Discussion

Augmenting the training data by applying random jitter improved the results for both the 3-layer and the 9-layer network. However, at best the 3-layer networks test set accuracy increased by 0.13 percentage points. The improvement of the 9-layer network was better as it increased by about 1 percentage point. This seems reasonable as the risk of overfitting is higher for the more complex 9-layer network compared to the 3-layer network. Thus, regularization is more useful in the complex network.

7 Augmenting training data via transformations

One approach to augment the training data used for object recognition was used by the runner up in the 2016 *NOAA Right Whale Recognition challenge* [1] and involved applying affine transformations such as translation and rotation on the training data. The intuition is to make sure that objects can be recognized from different viewing angles and of different sizes.

In this report, at every second epoch of training every individual training image was randomly mirrored with a certain probability. This implementation can be found in the function: *mirrorBatch*.

The following setting was used for the network:

$$\lambda = 0.005, n_s = 2250, n_{cycles} = 2$$

For reference, the 3 layer network optimized in this section achieved a test accuracy of 53.64% and the 9-layer network achieved 51.58% in the basic part of the assignment.

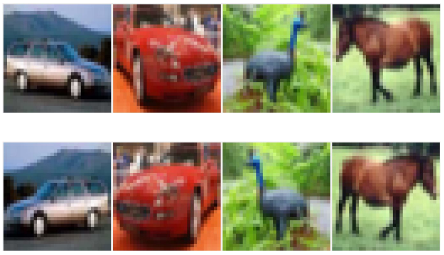

Figure 1: The first row contains four images from the training data set and the second row contains the corresponding mirrored images.

7.1 Results

The results with augmented training data via mirroring can be found in table 7.

Table 7: Results from augmenting training data via mirroring.

Mirror probability	Hidden Layers	Test set Accuracy
1%	[50, 30, 20, 20, 10, 10, 10, 10]	51.99 %
2%	[50, 30, 20, 20, 10, 10, 10, 10]	**52.11 %**
5%	[50, 30, 20, 20, 10, 10, 10, 10]	48.70 %
1%	[50, 50]	53.44 %
2%	[50, 50]	52.81 %
5%	[50, 50]	50.85 %

7.2 Discussion

Mirroring the training data did not improve the results of the less complex, 3-layer network. However, it did slightly improve the more complex 9-layer network which test set accuracy increase by approximately 0.5 percentage points. Again regularization is expected to be more efficient for a more complex network.

8 Combining methods

Several of the methods tried above were combined in order to improve the performance of the network even further. These results can be found in table 8.

Table 8: Results from the final networks with combined improvement attempts.

Hidden Layers	jitter	λ	Mirroring probability	Label smoothing range	Test set Accuracy
$[60, 50, 40]$	-	0.005	-	-	55.28 %
$[60, 50, 40]$	$+/-1\%$	0.005	-	-	54.70 %
$[60, 50, 40]$	-	0.005	-	[0.95, 1.00]	54.99 %
$[60, 50, 40]$	-	0.005	1 %	-	54.06 %

8.1 Results

The results for the final networks trained in this section can be found in table 8. Unfortunately combining the methods above did not improve the performance of the network with the best performing architecture from section 3. Perhaps this was because the best performing architecture was relatively shallow and the amount of overfitting. relatively small.

References

[1] *NOAA Right Whale Recognition, Winner's Interview: 2nd place, Felix Lau*, 2016 (Accessed May 20, 2019).

[2] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. *arXiv preprint arXiv:1801.05134*, 2018.

[3] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.