Assignment 1 - DL

Section 4

Hyper parameters :

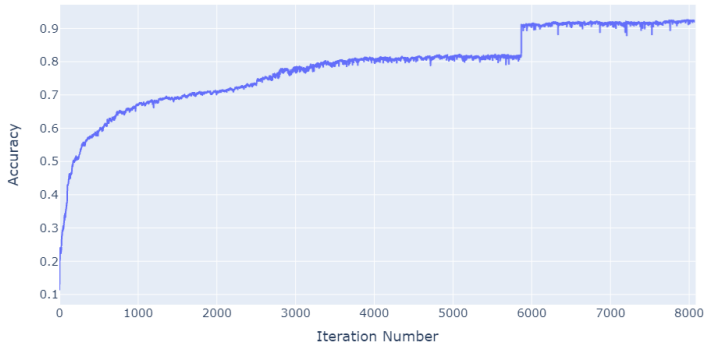
Convergence epsilon = 1e-7
Learning rate = 0.009
Train iterations= 12000
Batch size = 32

Convergence criteria -
We trained the network till the average accuracy on the validation set on the latest 100 iterations wasn't different (smaller than convergence epsilon) from the average accuracy in the 100 iterations before that.

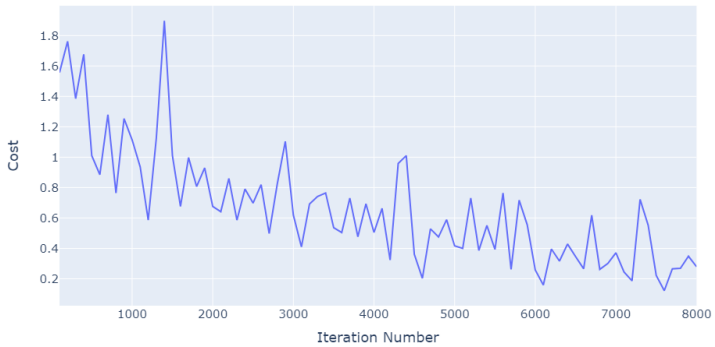
Results:

The model converges in iteration number 8083, epoch number 6, after 311.262 seconds.
.
final model accuracy for the training set: 0.928
final model accuracy for the validation set: 0.920
final model accuracy for the test set: 0.916

Accuracy on the validation set:



Costs on the training set:



The cost value for each 100 training steps can be found in the appendix.

Section 5

Hyper parameters : (same as in section 4)

Convergence epsilon = 1e-7
Learning rate = 0.009
Train iterations= 12000
Batch size = 32

Implementation:

We applied the batch normalization on each layer activation output. During training we kept for each layer the mean and standard deviation of the layer activation output (for each batch) and during prediction we used the average values of the last 100 batches. We kept in the parameters dictionary auxiliary variables for the implementation, such as:

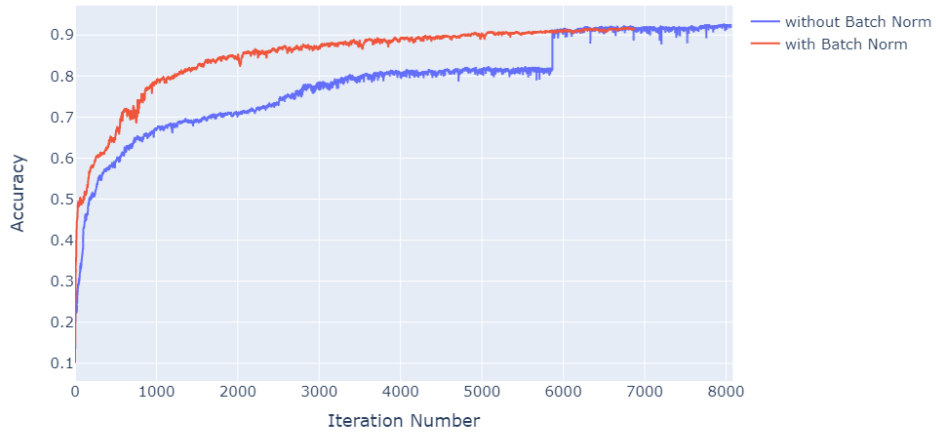
1. Boolean indicator that batchNormalization is required (batchnorm).
2. String indicator for the mode of the process (mode). can be set to “train” or to “predict”.
3. Dictionaries with activation mean and standard deviation values for each layer and batch.

Results:

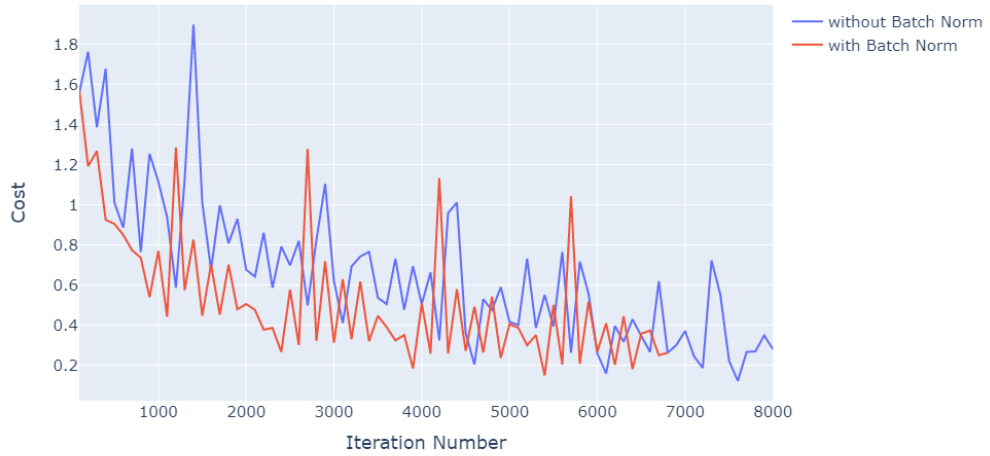
The model converges in iteration number 6886, epoch number 5, after 326.196 seconds.

final model accuracy for the training set: 0.920
final model accuracy for the validation set: 0.915
final model accuracy for the test set: 0.915

Accuracy on the validation set:



Costs on the training set:



From those results, we can infer that the learning process is much more efficient (as in accuracy) and fast using batch normalization.
The cost value for each 100 training steps can be found in the appendix.

Bonus - Dropout functionality

Hyper parameters : (same as in section 4)
Convergence epsilon = 1e-7
Learning rate = 0.009
Train iterations= 12000
Batch size = 32
Dropout probability (keep_prob)= 0.95 (the probability to keep a node)

Implementation:
We updated the function “linear_forward”, if the function is used during the training session, then a dropout is performed, otherwise (during the prediction session) a dropout is not performed. In addition, we added a boolean variable as a parameter to this function (and to higher hierarchy functions) which will indicate if the function is used in a training session or a prediction session.

```
def linear_forward(A, W, b, mode):
    """
    Description: Implement the linear part of a layer's forward propagation.

    input:
    A - the activations of the previous layer
    W - the weight matrix of the current layer (of shape [size of current layer, size of previous layer])
    B - the bias vector of the current layer (of shape [size of current layer, 1])
    mode - string value, indicate if the dropout is required

    Output:
    Z - the linear component of the activation function (i.e., the value before applying the non-linear function)
    linear_cache - a dictionary containing A, W, b (stored for making the backpropagation easier to compute)
    """
    if mode == 'dropout':
        drop_W = (np.random.rand(W.shape[0], W.shape[1]) < keep_prob) / keep_prob
        W = np.multiply(W, drop_W)
    linear_cache = {'W':W, 'A':A, 'b':b}
    Z = np.matmul(W, A)+b
    return Z, linear_cache
```

We set the dropout probability (keep_prob) to 0.95 because lower values didn't lead to model convergence. We used the dropout mode only during the train process and not during the predict process.

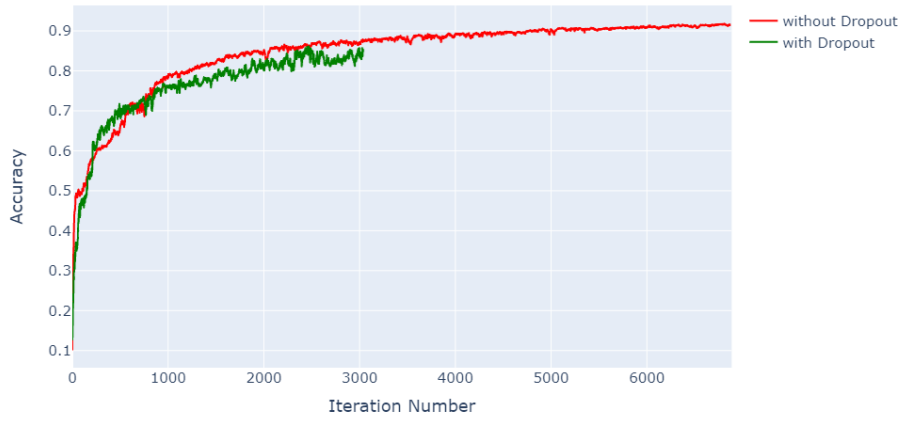
Results:

We compared results between the dropout model and the models in the previous sections.
The original model (without batch norm, presented in section 4) didn't converge with dropout technique.
The model with batch norm (presented in section 5) did converge with dropout technique.
For those reasons we will compare the results between two models, the first without dropout (presented in section 5) and the second with dropout, both of them include batch normalization.

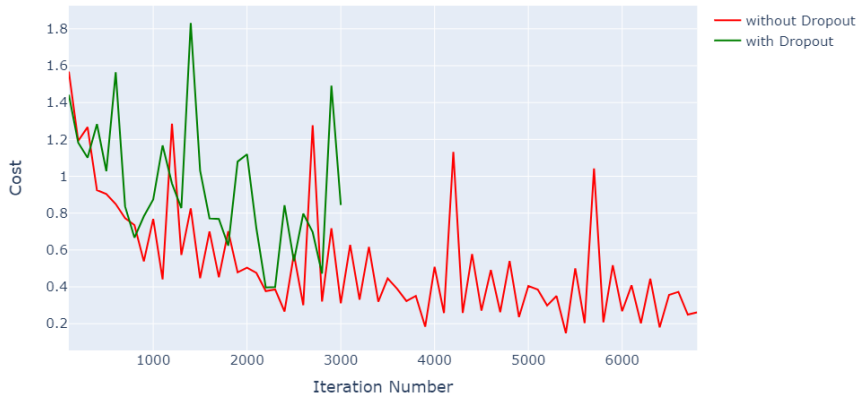
Results for dropout model:

The model converges in iteration number 3042, epoch number 3, after 126.576 seconds.
.
final model accuracy for the training set: 0.862
final model accuracy for the validation set: 0.853
final model accuracy for the test set: 0.853

Accuracy on the validation set:



Costs on the training set:



From those results we can infer that the learning process (for our implementation and for the MNIST dataset) is more stable without dropout technique. In addition, we can see that the accuracy is higher without using dropout. Our hypotheses that these results are because our network is quite small ([784, 20, 7, 5, 10]) and dropout regularization adds noise to the learning process.
The cost value for each 100 training steps can be found in the appendix.

New auxiliary functions explanations:

1. generateBatches(X, Y, batch_size) -
   Creating random batches from the entire data.
   Returns list of X batches and Y batches
2. create_one_hot_vector_y(y) -
   Creates a one hot vector from ground truth labels
3. standard_scale(X) -
   Scale the input data by standard scale
4. load_mnist() -
   Loads the MNIST dataset using keras.datasets and flatten the input data to a matrix of [m,784]
5. train_val_split(X, Y, split_size) -
   Split the training set to train set and validation set

Appendix:

<u>Section 4- The cost value for each 100 training steps:</u>
cost for iter:100, 1.5571389367368065
cost for iter:200, 1.7621070423263852
cost for iter:300, 1.3852695483212603
cost for iter:400, 1.676282964382402
cost for iter:500, 1.0097650037568282

cost for iter:600, 0.8850055092256315
cost for iter:700, 1.2798891567720205
cost for iter:800, 0.7640302573502371
cost for iter:900, 1.253304083706972
cost for iter:1000, 1.114315280036174
cost for iter:1100, 0.9370645083466538
cost for iter:1200, 0.5857775558990236
cost for iter:1300, 1.1234669090921965
cost for iter:1400, 1.897537571323799
cost for iter:1500, 1.010464749914552
cost for iter:1600, 0.6756508935548353
cost for iter:1700, 0.9973975322455091
cost for iter:1800, 0.8069391194364576
cost for iter:1900, 0.9290432674736695
cost for iter:2000, 0.6759812525790582
cost for iter:2100, 0.6413180600640846
cost for iter:2200, 0.8598802450293713
cost for iter:2300, 0.5858166366875133
cost for iter:2400, 0.7910172271055075
cost for iter:2500, 0.6978382815326316
cost for iter:2600, 0.8193107341986613
cost for iter:2700, 0.4972286526183185
cost for iter:2800, 0.8225671775892671
cost for iter:2900, 1.1036657984008815
cost for iter:3000, 0.618270034489023
cost for iter:3100, 0.40947583377544405
cost for iter:3200, 0.6916834021366638
cost for iter:3300, 0.7416847463135543
cost for iter:3400, 0.7645376809550114
cost for iter:3500, 0.5361774915688392
cost for iter:3600, 0.5042206882687588
cost for iter:3700, 0.7307041042696284
cost for iter:3800, 0.4766429145866039
cost for iter:3900, 0.6926530379191591
cost for iter:4000, 0.5044401194494182
cost for iter:4100, 0.6628197833463035
cost for iter:4200, 0.3234322738581391
cost for iter:4300, 0.9583680247108837
cost for iter:4400, 1.0102131181973033
cost for iter:4500, 0.36038210363114953
cost for iter:4600, 0.202933793747291
cost for iter:4700, 0.5283451937384683
cost for iter:4800, 0.4748620137146513
cost for iter:4900, 0.5886972501814889
cost for iter:5000, 0.4162443636159259
cost for iter:5100, 0.39997700706600564
cost for iter:5200, 0.7312476635472346
cost for iter:5300, 0.3860532377082554

cost for iter:5400, 0.5503213596462606
cost for iter:5500, 0.3930096127660574
cost for iter:5600, 0.7645002649834794
cost for iter:5700, 0.2616641070403921
cost for iter:5800, 0.716580179404414
cost for iter:5900, 0.5554361649965358
cost for iter:6000, 0.2575418481832642
cost for iter:6100, 0.1584780779612736
cost for iter:6200, 0.39569739193335757
cost for iter:6300, 0.31628315936972407
cost for iter:6400, 0.4285530728320172
cost for iter:6500, 0.3459121856824433
cost for iter:6600, 0.26572652432755994
cost for iter:6700, 0.6183779290151404
cost for iter:6800, 0.26196814772469845
cost for iter:6900, 0.3003963701067403
cost for iter:7000, 0.3703434968885928
cost for iter:7100, 0.24516793062242584
cost for iter:7200, 0.18609629890876717
cost for iter:7300, 0.7219592465597169
cost for iter:7400, 0.5526590915161318
cost for iter:7500, 0.22222948927372502
cost for iter:7600, 0.12158460646117726
cost for iter:7700, 0.26566463743805396
cost for iter:7800, 0.2687806774320233
cost for iter:7900, 0.3482730837643891
cost for iter:8000, 0.27922968124352415

Section 5- The cost value for each 100 training steps:

cost for iter:100, 1.567824928434261
cost for iter:200, 1.1920848282024887
cost for iter:300, 1.2664980665828398
cost for iter:400, 0.9245975806187143
cost for iter:500, 0.9039488405369299
cost for iter:600, 0.849350445104416
cost for iter:700, 0.7731691081398471
cost for iter:800, 0.7352663843854725
cost for iter:900, 0.5388716218684078
cost for iter:1000, 0.7688241109224527
cost for iter:1100, 0.440777552132822
cost for iter:1200, 1.2846643655528243
cost for iter:1300, 0.5730726452533473

cost for iter:1400, 0.8256304801543047
cost for iter:1500, 0.447059231970179
cost for iter:1600, 0.7007203312751049
cost for iter:1700, 0.45177488349557515
cost for iter:1800, 0.7010745925873805
cost for iter:1900, 0.4784655886640098
cost for iter:2000, 0.5044219197650627
cost for iter:2100, 0.4756590786547249
cost for iter:2200, 0.3768339218364827
cost for iter:2300, 0.3857091833899173
cost for iter:2400, 0.2664995321620006
cost for iter:2500, 0.5762958086958866
cost for iter:2600, 0.3003311328736591
cost for iter:2700, 1.2772309020016106
cost for iter:2800, 0.3203150474556513
cost for iter:2900, 0.7180701590778511
cost for iter:3000, 0.3114527054988875
cost for iter:3100, 0.6280659188994432
cost for iter:3200, 0.3302702498024633
cost for iter:3300, 0.616634871381823
cost for iter:3400, 0.31984881687784855
cost for iter:3500, 0.44567476233761394
cost for iter:3600, 0.3894694374031821
cost for iter:3700, 0.3231374076642386
cost for iter:3800, 0.35044581298920463
cost for iter:3900, 0.18383003510998405
cost for iter:4000, 0.5083618387304413
cost for iter:4100, 0.258302720232367
cost for iter:4200, 1.1327190046045685
cost for iter:4300, 0.25835482471961635
cost for iter:4400, 0.5784314401980135
cost for iter:4500, 0.27091880752127845
cost for iter:4600, 0.49141942636861136
cost for iter:4700, 0.26311891879553845
cost for iter:4800, 0.5405524855064968
cost for iter:4900, 0.23605598333183925
cost for iter:5000, 0.40466209179821655
cost for iter:5100, 0.38584023438114734
cost for iter:5200, 0.29889774846599226
cost for iter:5300, 0.3500960175466271
cost for iter:5400, 0.14876118955237572
cost for iter:5500, 0.5000476951135921
cost for iter:5600, 0.2034500298769948
cost for iter:5700, 1.0422989109511205
cost for iter:5800, 0.20734390727857116
cost for iter:5900, 0.5171404330729021
cost for iter:6000, 0.2676039578152732
cost for iter:6100, 0.4083341756573616

cost for iter:6200, 0.20189220541642555
cost for iter:6300, 0.4437255924468268
cost for iter:6400, 0.18041224898497538
cost for iter:6500, 0.35661701546556146
cost for iter:6600, 0.37272829051202466
cost for iter:6700, 0.2494622989750065
cost for iter:6800, 0.26154894574978804

Bonus Dropout- The cost value for each 100 training steps:

cost for iter:100, 1.4423526342278903
cost for iter:200, 1.1825730706798128
cost for iter:300, 1.1008961494031133
cost for iter:400, 1.2831250040020556
cost for iter:500, 1.0277102556093007
cost for iter:600, 1.5639002194851328
cost for iter:700, 0.8364090859110866
cost for iter:800, 0.6671154219770918
cost for iter:900, 0.7824327637416562
cost for iter:1000, 0.8744534274903866
cost for iter:1100, 1.1672681253856996
cost for iter:1200, 0.960691413000577
cost for iter:1300, 0.8274541982850716
cost for iter:1400, 1.8319005418338958
cost for iter:1500, 1.0311839758958938
cost for iter:1600, 0.7714298192964726
cost for iter:1700, 0.7685553473996565
cost for iter:1800, 0.6234884273431434
cost for iter:1900, 1.0795412357977145
cost for iter:2000, 1.1190876159060408
cost for iter:2100, 0.7169666823496088
cost for iter:2200, 0.3968513361872138
cost for iter:2300, 0.3991585158149564
cost for iter:2400, 0.8427211224207307
cost for iter:2500, 0.5397257486490388
cost for iter:2600, 0.7969790836478325
cost for iter:2700, 0.6968699302083436
cost for iter:2800, 0.4722103118686744
cost for iter:2900, 1.4914535578169326
cost for iter:3000, 0.8438494747887813