# Artificial Neural Networks - Torch + Lua

Arulkumar S (CS15S023)

September 16, 2015

## 1 Introduction to Torch

### 1.1 Setup

The startup code[1] is downloaded and the main file (qlua 1_data_SVHN_dataset.lua) is executed in the PC to train the already defined model upto 15 epochs.

The required SVHN dataset[3], MNIST dataset[4] are downloaded from it's source websites.

The time taken to complete one epoch with full dataset (73257 examples) = approx. 45 minutes

The time taken to complete one epoch with reduced (small) dataset(10000 examples) = approx. 10 minutes

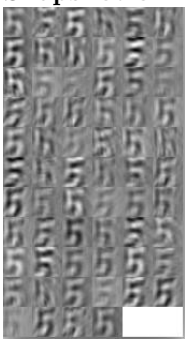
### 1.2 Testing and Visualization

The trained model (with SVHN dataset upto 15 epochs) is loaded independently & tested with one of the images of numbers.

The convolution layer outputs are shown below:

**Test image**

**Snapshot of filtered images from 1st convolution layer**



**Snapshot of filtered images from 2nd convolution layer**

### 1.3 Transferring the knowledge

The saved model (trained in SVHN dataset) can then be loaded independently of the training/test data & can be used to predict the numbers in image.

Since, the previous SVHN model is trained using RGB images, we will not be able to use the model for MNIST dataset, as MNIST contains images are of only Gray color.

So, as a first step, all the images in SVHN training dataset are converted to Gray image and used to train the model again.

I observed that there is a minor deviation between the notation of labels used in SVHN & MNIST datasets. The label settings in these two datasets are as below:

| Number in the image | SVHN label | MNIST label |
|---|---|---|
| 0 | 10 | 1 |
| 1 | 1 | 2 |
| 2 | 2 | 3 |
| 3 | 3 | 4 |
| 4 | 4 | 5 |
| 5 | 5 | 6 |
| 6 | 6 | 7 |
| 7 | 7 | 8 |
| 8 | 8 | 9 |
| 9 | 9 | 10 |

To equalize the labels between SVHN & MNIST datasets, the labels in MNIST datasets are subtracted by 1 & if the label is 0, then it is filled by 10.

Now, As per the requirement, the model is trained with SVHN dataset (for 10 epochs)) and tested with MNIST test dataset.

The observations are:

* Even after 10 epochs, the MNIST test set accuracy (68.1%) is less compared to the SVHN training set accuracy (96.375%)
* Though the SVHN training dataset error reduces fast during training, the MNIST test set performance is not good

The detailed log can be found here.

After getting a trained model from SVHN dataset, the same trained model is provided with the training data from MNIST dataset and trained for about 10 epochs.

The observations are:

* In the $1^{st}$ epoch, The test set accuracy is dramatically increased to 96% due to parameter fine-tuning.
* At the end of $3^{rd}$ epoch, the training set accuracy is reached 99.2%

The detailed log can be found here.

**during epoch#10 with SVHN training dataset & MNIST test dataset**

```
==> online epoch # 10 [batchSize = 1]
 [================================== 4000/4000 ==================================>] ETA: 0ms | Step: 36ms

==> time to learn 1 sample = 36.970651745796ms
ConfusionMatrix:
[[     763       5       0       3       1       1       5       0       2       1]   97.695%   [class: 1]
 [       5     523       1       4       1       0       1       0       2       1]   97.212%   [class: 2]
 [       5       2     449       2       3       0       1       2       1       1]   96.352%   [class: 3]
 [       6       2       1     406       0       2       0       2       1       0]   96.667%   [class: 4]
 [       2       0       5       2     374       3       0       2       2       0]   95.897%   [class: 5]
 [       1       0       2       4       6     273       0       1       0       2]   94.464%   [class: 6]
 [       5       4       2       0       0       0     277       0       0       0]   96.181%   [class: 7]
 [       3       2       3       2       1       7       1     272       1       1]   92.833%   [class: 8]
 [       1       1       1       2       2       1       0       2     239       2]   95.219%   [class: 9]
 [       2       1       0       1       0       1       0       0       0     279]]  98.239%   [class: 0]
 + average row correct: 96.075873970985%
 + average rowUcol correct (VOC measure): 92.743074893951%
 + global correct: 96.375%
==> saving model to /home/arul/workspace/Assignment2/Assignment2Starter/MNIST_test_with_SVHNmodel/results/model.net
==> testing on test set:
 [================================== 2000/2000 ==================================>] ETA: 0ms | Step: 23ms

==> time to test 1 sample = 15.297873020172ms
ConfusionMatrix:
[[     184       2       0       9       0       0      39       0       0       0]   78.632%   [class: 1]
 [       0     160      14      12       4       1       5      13      10       0]   73.059%   [class: 2]
 [       0       5     185       0      12       1       2       1       1       0]   89.372%   [class: 3]
 [      11      13       0      96       3       0      13       0      81       0]   44.240%   [class: 4]
 [       1       1       9       2     159       7       0       0       0       0]   88.827%   [class: 5]
 [       0       6       0      54       4      95       0       0       7      12]   53.371%   [class: 6]
 [       5      19       5       3      12       0     161       0       0       0]   78.537%   [class: 7]
 [       4      12      14       4      18      19       7     104      10       0]   54.167%   [class: 8]
 [       3      11       1      14      26       0      43       1      94       1]   48.454%   [class: 9]
 [       0      10       0      20       0      20       1       0       0     124]]  70.857%   [class: 0]
 + average row correct: 67.951506376266%
 + average rowUcol correct (VOC measure): 52.966233193874%
 + global correct: 68.1%
```

**during epoch#10 with MNIST training dataset & MNIST test dataset**

```
==> online epoch # 10 [batchSize = 1]
 [================================== 5000/5000 ==================================>] ETA: 0ms | Step: 52ms

==> time to learn 1 sample = 46.477428627014ms
ConfusionMatrix:
[[     563       0       0       0       0       0       0       0       0       0]   100.000%  [class: 1]
 [       0     488       0       0       0       0       0       0       0       0]   100.000%  [class: 2]
 [       0       0     493       0       0       0       0       0       0       0]   100.000%  [class: 3]
 [       0       0       0     534       0       0       0       0       1       0]   99.813%   [class: 4]
 [       0       0       0       0     434       0       0       0       0       0]   100.000%  [class: 5]
 [       0       0       0       0       0     501       0       0       0       0]   100.000%  [class: 6]
 [       0       0       0       0       0       0     550       0       0       0]   100.000%  [class: 7]
 [       0       0       0       0       0       0       0     462       0       0]   100.000%  [class: 8]
 [       0       0       0       0       0       0       0       1     493       1]   99.596%   [class: 9]
 [       0       0       0       0       0       0       0       0       0     479]]  100.000%  [class: 0]
 + average row correct: 99.940904378891%
 + average rowUcol correct (VOC measure): 99.87839281559%
 + global correct: 99.94%
==> saving model to /home/arul/workspace/Assignment2/Assignment2Starter/MNIST_test_with_SVHNmodel/results/model.net
==> testing on test set:
 [================================== 2000/2000 ==================================>] ETA: 0ms | Step: 18ms

==> time to test 1 sample = 18.096981048584ms
ConfusionMatrix:
[[     232       0       0       2       0       0       0       0       0       0]   99.145%   [class: 1]
 [       1     216       0       0       0       0       1       0       0       1]   98.630%   [class: 2]
 [       0       0     202       0       4       0       1       0       0       0]   97.585%   [class: 3]
 [       0       0       0     214       0       2       0       0       1       0]   98.618%   [class: 4]
 [       0       0       1       1     175       0       1       1       0       0]   97.765%   [class: 5]
 [       0       0       0       0       1     177       0       0       0       0]   99.438%   [class: 6]
 [       1       1       0       2       0       0     200       0       1       0]   97.561%   [class: 7]
 [       0       2       1       0       0       1       1     184       1       2]   95.833%   [class: 8]
 [       0       0       0       2       0       0       1       1     190       0]   97.938%   [class: 9]
 [       0       0       0       0       0       1       0       0       0     174]]  99.429%   [class: 0]
 + average row correct: 98.194207549095%
 + average rowUcol correct (VOC measure): 96.443321704865%
 + global correct: 98.2%
```

# 2 Own layer in Neural network

## 2.1 'ReQU' unit

source code

As per the given definition of *ReQU* unit, the implementation is done in *updateOutput*() & *updateGradInput*() function.

ReQU function output

$$z_i = \begin{cases} 0, & x_i \leq 0 \\ x^2, & x > 0 \end{cases}$$

The unit testing of the new ReQU layer is done using the unit test file. To test the model of ReQU, a new object of ReQU is created & the data is passed to the function $model : forward()$ & the output is verified.

The test log can be found here.

## 2.2 Testing the module on full network

The simple network used for testing ReQU is having the layered form as:

```
linear->ReQU->linear->softmax
```

For testing the gradient calculation done on the ReQU function *updateGradInput*(), an approximation method as shown below, is followed to estimate the expected Gradient.

$$\frac{\partial E}{\partial w_i} = \frac{f(w_1, w_2, \ldots, w_i + \epsilon, \ldots, w_n) - f(w_1, w_2, \ldots, w_i - \epsilon, \ldots, w_n)}{(2 * \epsilon)}$$

The $\epsilon$ is chosen as $10^{-7}$.

The measure of Symmetric relative error[1][2] is found between the Actual gradient ($g1$) & Estimated gradient ($g2$) as,

Relative error = $\frac{||g1-g2||}{2*||g1+g2||}$

It is observed that the symmetric relative error is close to the chosen $\epsilon$.

Also, the cosine similarity measure is calculated as,

cosine-similarity(actualGradient($g1$), estimatedGradient($g2$)) = $\frac{{g1}^T g2}{|g1|.|g2|}$

The calculated cosine similarity is very close to 1.

Hence, the results show that the *updateGradInput*() function in new module 'ReQU' is performing as expected.

The measures taken can be seen in the Log file

# References


[1] http://10.6.4.152/cnncv/Assignment_2.pdf.

[2] https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/practicals/practical4.pdf.

[3] http://torch7.s3-website-us-east-1.amazonaws.com/data/housenumbers/train_32x32.t7. *SVHN dataset.*

[4] http://torch7.s3-website-us-east-1.amazonaws.com/data/mnist.t7.tgz. *MNIST dataset.*