**Record of minor details during thesis**

**LSTM MNIST NN (Binary ad SC):**

1. Initializing weights which are drawn from a uniform distribution between -1/√hidden\_layers and 1/√hidden\_layers (resembles Xavier or Glorot initialization) improves the accuracy of the SC network a lot. This happens because it prevents gradients from vanishing or exploding which in turn prevents values saturating at 0 or 1 and hence doesn’t result in errors in SC operations resulting from correlation. Furthermore, this also keeps the weights between -1 and 1.
2. Moved from 1-layer (SC activation) LSTM network with 80% accuracy to 2-layer LSTM with both layers using SC which had an inference accuracy of 88%. Then, the second layer was implemented in binary which resulted in 90.52% accuracy. (the scalings were not adjusted until this point)
3. Made changes to the scaling factor of activation functions which improved the accuracy to 97.11% from 90.52% for the model with second layer in binary. The network was trained with σ(2x) and tanh(2x). Considering state machine-based activations, 8 state tanh function was designed. Hence,

Where, k = number of states. Hence,

And,

Earlier scaling of sigmoid function was incorrect where the input of Stanh function was only divided by 2 instead of 4. Another reason for dividing in SC domain and not entirely compensating the scaling by multiplying in the binary domain is because it keeps the values between [-1,1] and no additional steps are needed to normalize the values.

1. Up until this point the LSTM layers had 400 hidden layers. The gate computations of SC layer were performed in binary and the actions, hidden and cell state calculations were done in SC.
2. The 2nd LSTM layer i.e. the binary layer was implemented using lstm() function which was corrected and implemented.
3. Dividing cx by max value of cx compensated for the scaling by 2 during SC addition.
4. First LSTM layer returns sequence of hidden states for all time steps to be fed into the second LSTM layer. This isn’t the case for the last LSTM layer as we only need the last time step sequence (or all inputs of the given batch size).
5. A minor error was noticed in the forward() function of Class CustomLSTM(). If init\_states is None, the hidden and cell states are defined as 2 dimensional matrices but it should be 3 dimensional with the first one missing i.e. the sequence length dimension. Although this is redundant as everywhere in the code the states are initialized but corrected it nonetheless.
6. Refer to Table 1 for accuracy of the LSTM network with one SC and one binary LSTM layers with varying size of hidden layers. The training accuracy of the model was 99% and inference accuracy of fully binary model was close to 98.5% in all cases. (98.85% for 400 layer architecture)

The SC LSTM additions were performed using MUX in these cases.

1. The SC network breaks down below a certain number of LSTM layers. Checked for 32 hidden layers and the inference accuracy reduced drastically to only 46%. Possible reason is that more layers can represent the information better because it is averaged out between larger number of bitstreams.
2. Made changes in the code to make it possible to have different hidden layer number for different LSTM layers.

|  |  |  |
| --- | --- | --- |
| **No. of hidden layers** | | **Accuracy (%)** |
| **Layer 1 (SC)** | **Layer 2 (Binary)** |
| 400 | 400 | 97.11 |
| 200 | 200 | 97.39 |
| 128 | 128 | 96.2 |
| 200 | 100 | 97.24 |
| 100 | 200 | 97.49 |

Table : Comparing accuracies of model with varying hidden layer sizes

**Accumulative Parallel Counter (APC):**

1. MUX based adder was replaced with APC based adder because of its better reliability
2. An APC consists of 2 parts: a CPC and a ripple carry adder.

The size of the CPC depends on how many bit streams need to be compared parallelly. A larger CPC can be constructed using multiple smaller CPCs using divide and conquer strategy. Given two CPCs each with inputs, a -input CPC is obtained by the use of a ripple-carry adder of length . Similarly, two such -input CPCs can be combined with a -bit ripple-carry adder to produce a -input CPC. This procedure is repeated until a CPC of size is obtained. (Don’t confuse ripple carry adder used to construct CPCs with the 2nd part of the APC which is also a ripple carry adder).

The 2nd part of the APC i.e. the RCA has number of stages equal to the number of bits required to represent the maximum sum. For ex., if the bit stream length of a number is 5 and we need counter for two such numbers in parallel, the max counter value can be 10 (when all bits of both numbers are 1). Hence the number of stages of RCA = 4 as a minimum of 4-bits are needed to represent 10.

The CPC calculates the current count and the RCA stores the total count of all previous time steps.

1. In our case, since the SC addition is being performed only for cell state calculation, it requires only a 2 input CPC to add the 2 terms of cell state. The SC addition resembles a counter since what we eventually want is the total number of 1s in the bit stream.

For a 2 parallel inputs, only 1 FA is required for the CPC and n-stage ripple carry adder where n =

1. If the number of additions is a lot, make use of Approximate Parallel Counter which first passes a pair of 2 inputs through AND and OR gates alternatively before passing it on to CPC to reduce the inputs to the CPC. The weight of the outputs of the gates is instead of .
2. Accuracy of the network improved by ~0.3% for 100/200 and 200/100 hidden layer architectures when APC was implemented with second LSTM layer in binary. The accuracy when both LSTM layers were using SC activations with APC came out to be 97.47%.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Layer 1**  **(SC/Binary - no. of layers)** | **Layer 2**  **(SC/binary - no. of layers)** | **Adder type**  **(MUX/APC)** | **Inference**  **Accuracy** | **Binary Model Inference Accuracy (%)** | **Training Accuracy (%)** |
| S – 400 | S – 400 | MUX | 96.65 | 98.85 | ~99% |
| S – 400 | B – 400 | MUX | 97.11 | 98.85 | ~99% |
| S – 200 | B – 200 | MUX | 97.39 | ~98.5 | ~99% |
| S – 128 | B – 128 | MUX | 96.2 | ~98.5 | ~99% |
| S – 200 | B – 100 | MUX | 97.24 | 98.52 | ~99% |
| S – 100 | B – 200 | MUX | 97.49 | 98.5 | ~99% |
| S – 400 | S – 400 | APC | 97.47 | 98.85 | ~99% |
| S – 200 | B – 100 | APC | 97.5 | 98.52 | ~99% |
| S – 100 | B – 200 | APC | 97.73 | 98.5 | ~99% |

Table : Summary of implementations

**14.07.25  
Note: The above accuracies might need to be changed if the normalisation of cell state is performed sample wise instead of batch dependant normalization. This means that earlier the max value was being searched across all the samples of a batch given at once, and this value was being used to normalize cx of all samples. But the correct way should be to do this sample wise to avoid the dependency on batch size.**

**The normalization was changed from batch dependant to sample-wise batch independent normalization. The above table still hold valid since they give a clear picture of overall accuracy of SC architecture.**

**CW Attack (Untargeted):**

1. The initial attacks were carried out by using ART library for ML security. It contains the code by Carlini and Wagner to carry out the L0, l2 and L∞ attacks. But this library accepts only Numpy arrays and not tensors. Hence, the simulation cannot be run on GPUs and required much longer processing time. So, Torchattacks library was used instead to reduce the processing times. L2 norm was used for the attacks.
2. Table 3 elaborates on the different parameters used to carry out the attacks. These attacks were always carried out on a batch size of 100 and learning rate of 0.01.

These attacks were carried out on the model with batch dependant cell state normalization. It was observed that the binary accuracy of the attack model didn’t match the binary accuracy of the main model only because the main model had a different test batch size. This is when it was observed that the cell state normalization was batch dependant. This was later fixed and hence the following results become invalid.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Batch** | **HL1** | **HL2** | **C** | **K** | **Steps** | **Cw\_bin** | **bin** | **1 SC** | **2 SC** |
| 1 | 200 | 100 | 10-4 | 0 | 1000 | 99% |  |  |  |
| 1 | 200 | 100 | 1 | 0.5 | 1000 | 60% |  |  |  |
| 1 | 200 | 100 | 0.1 | 0.5 | 1000 | 96% |  |  |  |
| 1 | 200 | 100 | 10 | 0.5 | 1000 | 14% |  |  |  |
| 1 | 200 | 100 | 1 | 1 | 1000 | 59% |  |  |  |
| 1 | 200 | 100 | 10 | 1 | 1000 | 13% |  |  |  |
| 1-5 | 200 | 100 | 10 | 1 | 1000 | 14.6% | 12.6% | 69.2% | 70.4% |
| 1-5 | 200 | 100 | 10 | 1.6 | 1000 | 14.2% | 12% | 65.6% | 60% |
| 1-5 | 200 | 100 | 10 | 1.62 | 1000 | 14.2% | 12.2% |  |  |
| 1-7 | 200 | 100 | 10 | 1.6 | 1000 | 13% | 10.57% | 65.86% | 63.57% |
| 1-7 | 200 | 200 | 10 | 1.6 | 1000 | 7.57% | 6% | 64% | 60.57% |
| 1 | 200 | 100 | 10 | 1.6 | 1000 | 10% | 12% |  |  |
| 1-5 | 200 | 100 | 20 | 1.6 | 1000 | 10.6% | 9.4% | 65% | 61.2% |
| 1-5 | 200 | 100 | 30 | 1.6 | 1000 | 9.6% |  |  |  |
| 1-5 | 200 | 100 | 40 | 1.6 | 1000 | 10% |  |  |  |
| 1-5 | 200 | 100 | 30 | 2 | 1000 | 10.4% |  |  |  |
| 1-5 | 200 | 100 | 30 | 1.6 | 1100 | 9.4% | 8.6% | 61.2% | 60.2% |
| 61-65 | 200 | 100 | 10 | 1 | 1000 | 18.2% | 14% | 74% | 71% |

Table : CW attack parameters

It was observed that CW binary accuracies above 14-15% resulted in SC accuracy of more than 68% and very close to 70%. So, the attack accuracy was decided to be kept around 14-15%. Also an architecture of 200 and 100 hidden layers was selected since 200 and 200 was not giving any exceptionally better SC results.

1. After observing the inconsistency in accuracy due to batch size variation, the normalization of the cell state was changed to sample wise. But it was observed that even though binary accuracy was quite consistent in this case the SC accuracy reduced from the 97% range to 92% range.

Hence, cx/2 normalization was used which is also consistent with the APC implementation. Both the binary and SC accuracies in this case remained consistent with the original normalization technique.

1. Table 4 elaborates on different parameters used to carry out the attacks. These attacks were always carried out on a batch size of 100 and learning rate of 0.01 with cx/2 normalization.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Batch** | **HL1** | **HL2** | **C** | **K** | **Steps** | **Bin accu** | **1 SC** | **2SC** |
| 1-5 | 200 | 100 | 10 | 1.6 | 1000 | 9.4% | 66.4% |  |
| 1-5 | 200 | 100 | 10 | 1 | 1000 | 9.4% |  |  |
| 1-5 | 200 | 100 | 10 | 1.4 | 1000 | 9.4% |  |  |
| 1-5 | 200 | 100 | 10 | 2 | 1000 | 9.4% |  |  |
| 1-5 | 200 | 100 | 10 | 0 | 1000 | 9.4% |  |  |
| 1-5 | 200 | 100 | 100 | 0 | 1000 | 3.8% | 66.4% | 69% |
| 1-5 | 200 | 100 | 200 | 0 | 1000 | 3.6% | 61% |  |
| 1-5 | 200 | 100 | 1000 | 0 | 1000 | 3.6% |  |  |
| 6-10 | 200 | 100 | 100 | 0 | 1000 | 3.92% | 66.2% | 66% |
| 1-50 | 200 | 100 | 100 | 0 | 1000 | 2.6% | 67.4% |  |
| 51-100 | 200 | 100 | 100 | 0 | 1000 | 3.92% |  |  |

It was observed that changing the kappa values in this case did not have any influence on the attack accuracy. Accuracy of bathes from 1-5 didn’t go below 3.6% for any further changes made. So, a lot of other batches like 6-10 and 11-15 were tested on c value equal to both 100 and 200 as the attacks were quite successful on these values to finalize a value for c. After running these tests, it was observed that most batches didn’t have any significant difference between c=100 or c=200, hence c=100 was selected as the parameter value for carrying out the untargeted CW attacks.

Accuracy of the model from batch 1-50 in batch groups of 5 can be checked in GIT repo saved files which mention these values. Also, for further insights into which classes were the most resistant to the attacks or most vulnerable to attacks can be found in the excel file ‘CW\_attack\_outputs.ods’.

**CW Attack (Targeted):**

1. Started by running targeted attacks with target class as 9 as it was easiest to attack for the untargeted attack with the least number of correct predictions. Only because of the above reason class 9 was chosen but for the ‘Best Case’ attack the class would be chosen based on the perturbation distance and not how easy it was to carry out the attack during untargeted attack. It has been observed that targeted attack is much more computationally intensive as compared to untargeted and requires much larger c values and number of steps for it to be a nearly successful attack.