**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) LSTM network with 80% accuracy to 2-layer LSTM with the second layer in binary with 90.52% accuracy.
3. Made changes to the scaling factor of activation functions which improved the accuracy to 97.11%. 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 correcting 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.

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 eg., 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 .