PROPOSAL

PARALLEL LSTM TRAINING FOR SEQUENCE PREDICTION FROM SEQUENTIAL DATA

Group name: LSTM
List of members:
- 19120454 - Bùi Quang Bảo

Keywords: Parallel, Machine Learning, Sequence Prediction, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM)

List of references:
- Long Short-Term Memory:
  - Original paper: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory
  - Understanding: https://arxiv.org/pdf/1909.09586.pdf
  - Explanation: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
- RNN numpy implementation (not using Tensorflow, Keras, PyTorch, etc):
  - https://quantdare.com/implementing-a-rnn-with-numpy/
  - https://github.com/gy910210/rnn-from-scratch
  - https://peterroelants.github.io/posts/rnn-implementation-part01/
- LSTM numpy implementation (not using Tensorflow, Keras, PyTorch, etc):
  - https://www.analyticsvidhya.com/blog/2022/01/the-complete-lstm-tutorial-with-implementation/
  - https://mattgorb.github.io/lstm_numpy
  - https://github.com/nicklashansen/rnn_lstm_from_scratch/blob/master/RNN_LSTM_from_scratch.ipynb
  - https://blog.varunajayasiri.com/numpy_lstm.html

Content

1. Summary:

In this project, our group will analyze and parallel the LSTM model (a RNN - Recurrent Neural Network) in order to improve its training speed and efficiency. By utilizing parallel processing, the model can handle larger datasets and have shorter training duration. The specific task that we want to apply using the LSTM model in this project is time-series prediction - sequence prediction from sequential data. We will implement a raw LSTM model using Numpy library, analyze, parallelize and measure the efficiency.

2. Background:
LSTM stands for Long Short-Term Memory. It is a type of recurrent neural network (RNN) that is designed to handle the problem of vanishing gradients in traditional RNNs. LSTM networks have the ability to selectively remember or forget information over long periods of time, making them particularly effective for tasks that involve sequential data, such as speech recognition, language translation, and time series prediction. The architecture of an LSTM network includes memory cells, input and output gates, and forget gates, which work together to regulate the flow of information through the network.

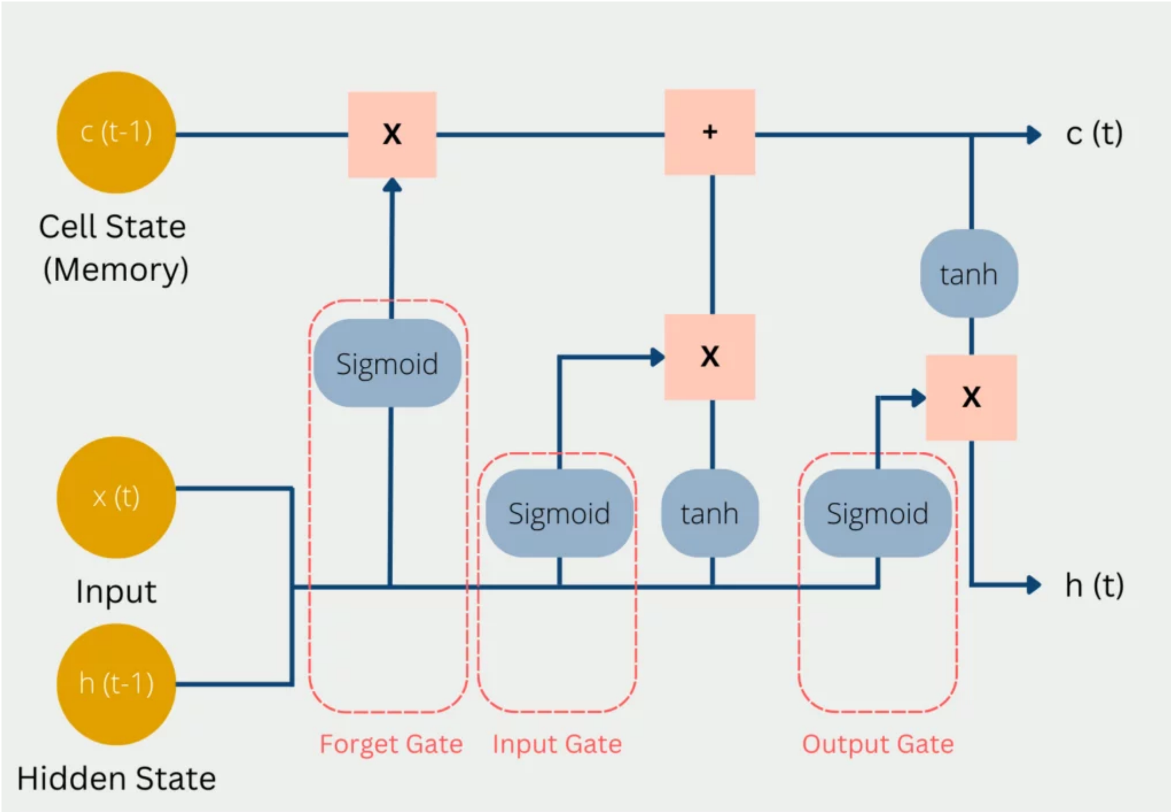

Image from https://databasecamp.de/en/ml/lstms

The compute-intensive process is the training phase of the LSTM model. For a simply explanation, the training will include some processes:

- Data one-hot encoding
- Forward pass
- Backward pass
- Losses calculation
- Optimization function (for neural network parameters updating)
- …

To train the LSTM model, the initial neural network will run through a number of epochs, each epoch will “look” at all the samples of validation data and training data, each sample will go through the processes mentioned above. As a consequence, it takes a lot of time to train a good final LSTM model.
The training phase of a LSTM model will benefit from parallelism.

3. The challenge:
The hard part is that the LSTM model training does not contain independent loops.

For example:
- Epochs cannot be parallelized (at least we think so now), because after each epoch, the parameters must be updated by the optimization functions (Gradient Descent or Adam).
- The process of "looking" at validation data and training data is not totally independent and it is based on how we implement the optimization functions.

By doing this project, we hope to find a solution of parallelism for this type of training that uses Gradient Descent or Adam as optimization functions.

4. Resources:

Resources that we may use as starter code:
- https://mattgorb.github.io/lstm_numpy
- https://blog.varunajayasiri.com/numpy_lstm.html
- https://github.com/nicklashansen/rnn_lstm_from_scratch/blob/master/RNN_LSTM_from_scratch.ipynb

For full references, we've already listed in the "List of references" above.

Computer: We do not plan to use a personal computer/laptop. We will take advantage of some online services, such as Google Colab and Kaggle.

5. Goals and Deliverables

5.1. Goals:

5.1.1. Plan to achieve (100%):

The final parallelized version will run x5 faster than the sequential version.

5.1.2. Hope to achieve 1 (125%):

The final parallelized version will run x10 faster than the sequential version.

5.1.3. Hope to achieve 2 (150%):

The final parallelized version will run faster than the sequential version, and training time will not scale up as quickly as the size of the dataset.

Example:
- Dataset A (1000 samples) training takes 60s.
- Dataset B (2000 samples) training takes *much less* than 120s (should be 80s or 90s for example).

5.1.4. In case the work goes slowly (75%):

The final parallelized version will run faster than the sequential version.

5.2. Deliverables

We hope that we can deliver a good source code that can be easily followed and understood.

We plan to present a sequence prediction demo at the seminar: We will talk about the input data we have prepared, what predictions we hope to get, training the LSTM model directly in the seminar, what predictions we actually get, and conclusion about the results.

Schedule:

Time	Assignee	To do
Phase 1 20/3 - 26/3	19120454 - Bùi Quang Bảo	Understand the LSTM model and analyze what and how to parallel.
Phase 2 27/3 - 9/4	19120454 - Bùi Quang Bảo	Finish the sequential implementation of the LSTM model.
Phase 3 10/4 - 30/4	19120454 - Bùi Quang Bảo	Parallel LSTM model training. Update the sequential implementation if necessary. Present the project at weekly reports or mid-term seminars, if any.
Phase 4 1/5 - 7/5	19120454 - Bùi Quang Bảo	Write the report about the project and the results.
Phase 5 Unknown	19120454 - Bùi Quang Bảo	Demo and present the project at the final seminal.
Phase 6 Unknown	19120454 - Bùi Quang Bảo	Update the project after the final seminar, if required.