Appendices

Appendix A

Computational Setup

In this appendix we include the Computational Setup of our Cortical Spectro-Temporal Model (CSTM), which describes its object-oriented design as well as the parallelization strategy used for its implementation in High Performance Computing (HPC) resources. This appendix also includes preliminary Strong and Weak Scaling tests of our code on HPC resources.

A.1 Inheritance and Compositional Structure

We implemented our algorithms in standard C++14 using the Object-oriented programing (OOP) paradigm in a set of classes interrelated by inheritance and composition. We implemented proximal afferent and distal inter-columnar connectivity as well as minimum-maximum margins in afferent synaptic weights in the Encoder Layer (EL) class. We configured such class as a composition of objects of class Complex Self-Organizing Map (CSOM). The main member in the Encoder Layer (EL) class is a Standard Template Library (STL) vector of Complex Self-Organizing Map (CSOM) objects. A complete diagram of the hierarchical inheritance and compositional structure of the implementation can be seen in Fig. A.1.

A.2 Encoder Layer (EL) Parallelization

We parallelized the EL class by means of a hybrid Message Passing Interface (MPI)+Open Multi-Processing (OpenMP) paradigm. We distributed CSOMs among MPI ranks as a deck of cards is distributed among different players. Each MPI rank ends up with one or more CSOMs and the CSOMs in each rank are distributed among different OpenMP threads (Fig. A.2 A and B respectively). Information among MPI ranks must be transferred in each time step. We gather all the information corresponding to the CSOMs in each rank and then use MPI Bcast function to transmit such information using a special comunication protocol by means of which we specify the boundaries in the information corresponding to each CSOM(Fig. A.2 C). By means of this strategy each MPI rank has to call MPI Bcast just once in order to transmit its data. The EL uses MPI I/O parallel file system to save its status in Matlab/Octave format (Fig. A.2 D). Each MPI rank gathers all the data corresponding to its CSOMs in the EL and communicates the part of the file it will use to the other MPI ranks, in order to store the data without interfering with the other ranks in the MPI environment. Then, each MPI rank saves all its data with a unique call to MPI Write.

The implementation has Checkpoint and Restart capacity in its training stage where there is total flexibility in terms of the number of ranks with which the execution is restarted. A cortical layer could have been saved with n ranks and such layer could be restarted with m ranks without affecting the final results.

It is important to note that the number of CCs shown in Fig. A.2 is merely illustrative and does not reflect the real numbers in terms of computational resources used for this implementation.

We performed all computational experiments on Cooley, a visualization and analysis cluster at Argonne National Laboratory. We ran the simulations using 25 nodes (9 CCs per node and one node per MPI rank) and 9 OpenMP threads per node/rank (one thread per CC).

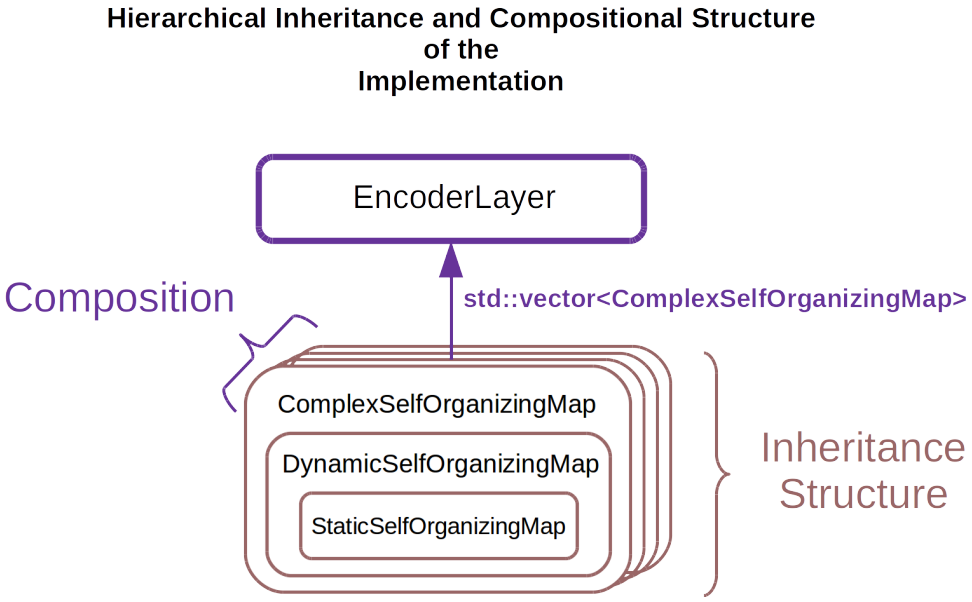

Figure A.1: Hierarchical Inheritance and Compositional Structure of the Model Implementation. Complex Self-Organizing Map (CSOM) inherits from Dynamic Self-Organizing Map (DSOM) which inherits from Static Self-Organizing Map (SSOM). The Encoder Layer (EL) is formed by the composition of a set of CSOMs gathered in a std::vector Standard Template Library (STL) container.

A.3 Initial Strong and Weak Scaling Tests on Cooley

Beyond the notion that our computational approach is intended to be applied in leadership supercomputers in the future, in the present work an essential step is to test our code in order to see how it uses the resources provided by Cooley Nodes. Parallel scalability is a measurement that indicates how efficient is our code when using increasing numbers of parallel processing elements–Nodes or Processes and Central Processing Units (CPUs) or Threads on Cooley. Fig. A.3 shows the scaling capacity of our code in terms of run time vs. number of processing elements used for the task. In our tests we always constrain our code to run one MPI rank per Cooley node. Each MPI rank spreads an specific number of threads through the different CPUs in its corresponding node (Fig. A.2 A and B). There are two ways to measure the parallel performance of a given application. The measure to be applied will depend on whether the application is CPU-bound or memory-bound. Such measurements are referred to as *strong* and *weak* scaling, respectively.

Straight lines in Fig. A.3 (left) show an initially good strong scalability of our code while the small slope exhibited by Fig. A.3 (right) allows us to foresee a good weak scaling performance of our code in high end leadership supercomputers.

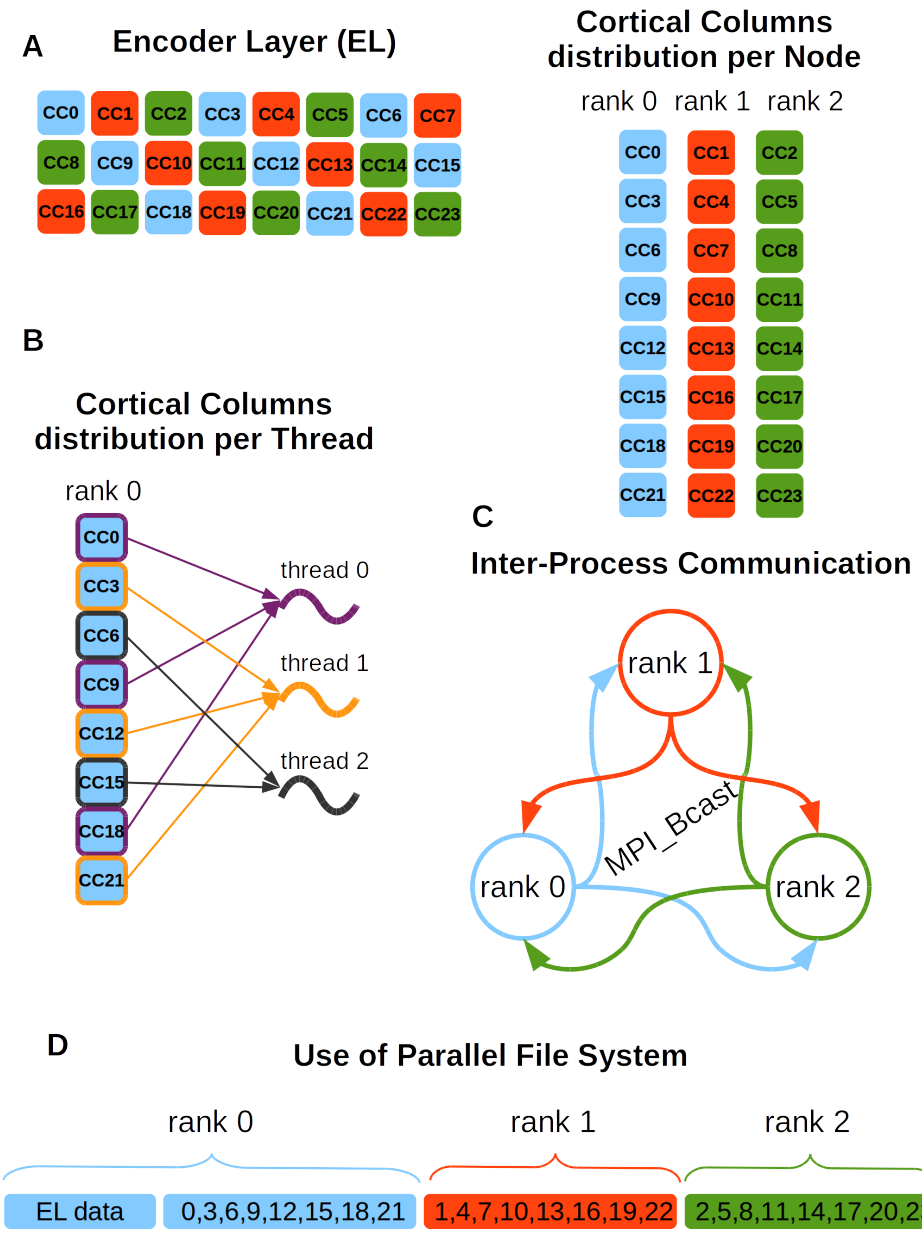

Figure A.2: Encoder Layer (EL) MPI+OpenMP parallelization. (A) Distribution of CSOM objects in an EL with 3 by 8 (24) CCs among three MPI ranks with three OpenMP threads per rank. Certain ranks could take care of a different number of CSOMs depending on the number of MPI ranks as well as the number of CCs in the EL. (B) Each MPI rank distributes its CSOMs among different threads in the same fashion. (C) MPI IPC among different ranks. IPC is carried out at each time step since each MPI rank requires the complete EL output at each time step. Each MPI rank broadcasts the information corresponding to its CCs to the other ranks in the MPI environment. (D) EL information distribution in a file to save its status. Each MPI rank puts the formated data corresponding to its CSOMs in a STL stringstream class template. Rank 0 also takes care of the EL structure, connectivity and parameters. Once each rank has its stringstream with the formated data, it communicates its file view to the other ranks. Then each rank writes its stream of bytes in parallel without interfering with other ranks in the MPI environment. An EL with a different number of ranks can load the same file without affecting the final results. Each rank in the new EL loads the complete file in a STL stringstream class template and then takes the informations that concern it from such structure.

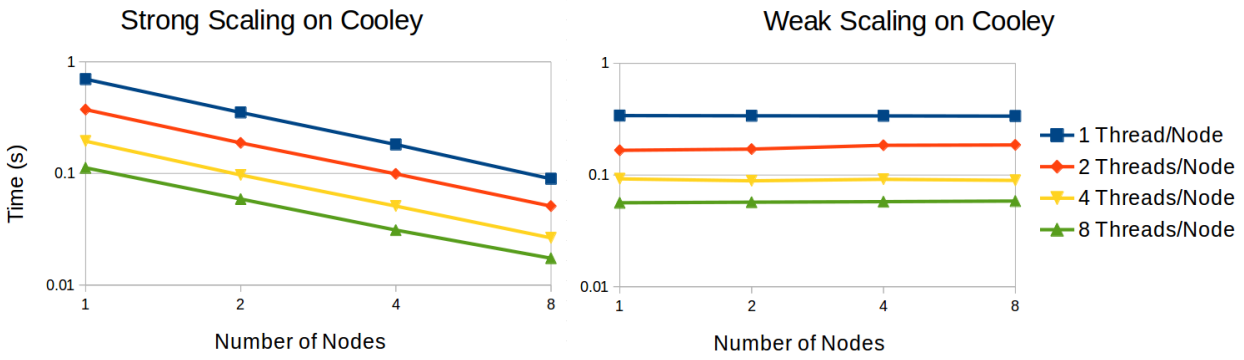

Figure A.3: Strong and Weak scaling tests on Cooley nodes. Left: Strong scaling. Run time vs. the number of nodes for different number of threads per node. The problem size stays fixed but the number of processing elements are increased. Right: Weak scaling. Run time vs. the number of nodes for different number of threads per node. In this case the problem workload assigned to each processing element stays constant and additional elements are used to solve a larger total problem (i.e. a problem that would not fit in the available RAM on a single node).

Appendix B

Encoder Layer parameters swept

This appendix includes a battery of complementary experiments showing the classification accuracy levels of different instances of the EL in the CSTM.

B.1 Word Classification Performance Tests

The EL has shown exceptional classification performance. In order to attain the appropriate configuration on the parameters describing the structure of the model, we produced a swept of models with different sizes in the EL. We progressively increased the number of cortical columns (CCs) as well as the distal receptive fields in the EL and tested the corresponding phonetic classification performances.

In Fig. B.1 we show the initialization parameters for the different ELs.

Model	Cortical Columns	Number of Units	Receptive Fields	Used Nodes
ELayer0	1*1	15*15	1*1(0)	1
ELayer1	3*3	15*15	3*3(1)	1
ELayer2	9*9	15*15	9*9(4)	9
ELayer3	15*15	15*15	15*15(7)	25
ELayer4	21*21	15*15	15*15(7)	49

Figure B.1: EL parameters swept. The number of CCs goes from 1 to 441 in 4 steps. The receptive fields go from 1 to 225 CCs in 4 steps too. The number of neural units in each CC is kept constant and the number of compute nodes to run the models is varied appropriately in order to get a fair distribution of CCs in each compute node.

The EL0 had one CC with 15 for 15 (225) Neural Units, the EL1 had an array of 3 for 3 (9) CCs with 225 Neural Units and so on. As depicted by the figure, a receptive field of 1 for 1 corresponds to a parameter of 0 and represents a receptive field of one CC. This means that each CC has lateral distal connections only with itself. A receptive field of 3 for 3 represents the fact that each CC can be distally connected with 9 CCs around it, including itself. For each EL we used a number of nodes as to try to run 9 CCs per Node distributed by means of OpenMP multi-threading operations.

Fig. B.2 shows the average phonetic word classification performance obtained by each EL. As can be seen in the figure, even the smallest instance of the EL (the EL0) outperformed the Multiresolution Spectro-Temporal Sound Analysis (MRSTSA) algorithm in the word classification task. This supports our hypothesis that the improvement in the performance comes from the algorithmic sequential characteristics of the ELs.

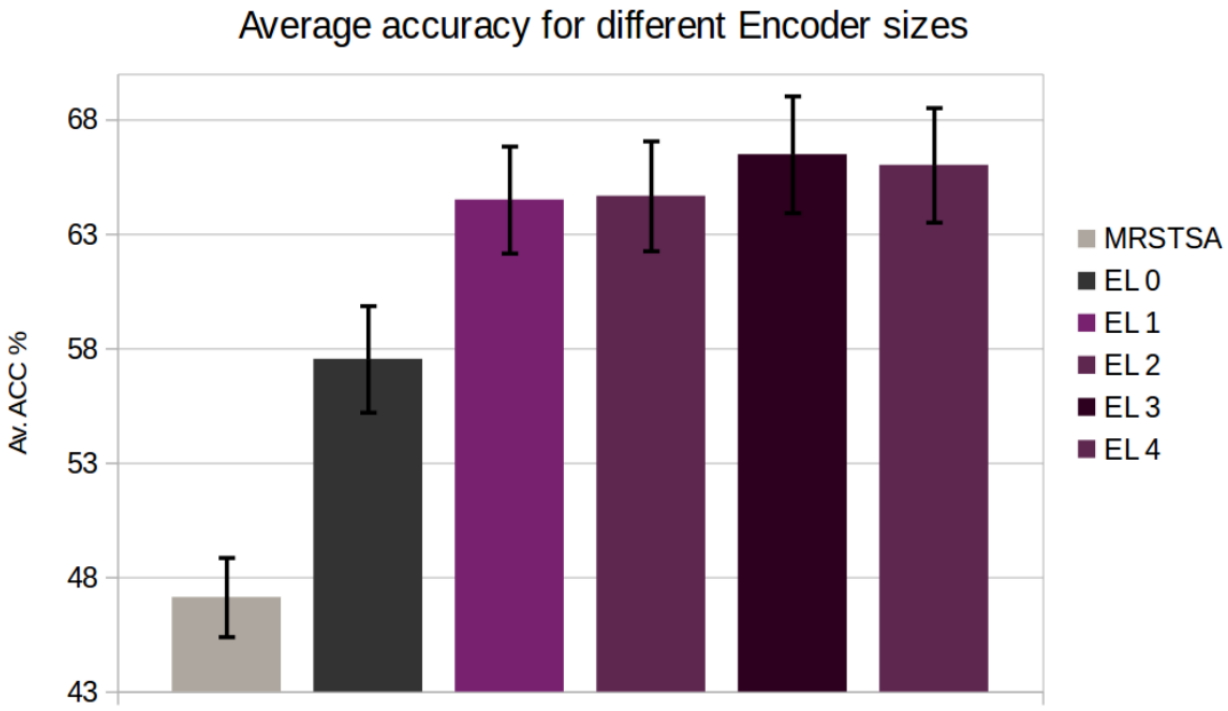

Figure B.2: Classification Accuracy for the MRSTSA and for different instances of the EL.

From Fig. B.2, a clear tendency of the performance can also be seen where the classification accuracy grows with the size of the 4 first ELs. Bigger models with larger numbers of CCs present more phonetic hypotheses which could allow better probabilistic inference in the sequential processing task. It is also known that smaller models suffer from a distal dendrite sparsity which generates repetitive Massive Firing Events (MFEs) and a lack of Sparse Distributed Representations (SDRs) whose origins are prediction faults produced in the processing of the stream of data. In this way, a modification in some parameters of the model, could affect others in unpredictable ways and a comprehensive study of such phenomena far exceeds the purpose of the current research. Regardless of that, we consider that the present battery of experiments shows a sound initial parameter configuration as to pose an appropriate EL arrangement.

B.2 Experimental Data to Run the Tests

For the entire collection of experiments we generated corpora of 500 words with mono, di and trisyllabic English words with vocabularies of five words using Festival Text to Speech Synthesis.

The vocabularies used including the following:

- Monosyllabic vocabulary: *map, dog, mouse, with* and *truck*.
- Disyllabic vocabulary: *answer, doctor, teacher, summer* and *tennis*.
- Trisyllabic vocabulary: *computer, telephone, rectangle, tomato* and *magazine*.

The Voices used include the following:

We use the following English speaking voices provided by Festival: `cmu_us_fem_cg`, `cmu_us_gka_cg`, `cmu_us_ksp_cg`, `cmu_us_rxr_cg`, `cmu_us_jmk_cg`, `cmu_us_rms_cg`, `cmu_us_slt_cg`, `cmu_us_jmk_arctic_clunits`, `cmu_us_rms_arctic_clunits`, `cmu_us_slt_arctic_clunits`.