ML 19/20-5.8 Serialization of Spatial Pooler

Anik Biswas(1324853) anik.biswas@stud.fra-uas.de
Yarlagadda Raghavendra(1324413) raghavendra.yarlagadda@stud.fra-uas.de
Lakshmi Mounika Kolisetty(1324400) lakshmi.kolisetty@stud.fra-uas.de
Manash Chakraborty(1325085) manash.chakraborty@stud.fra-uas.de

Abstract— Abstract : Hierarchical temporal memory (HTM) is a machine intelligence model which is inspired by the Neocortex and provides a framework to potentially perform learning and prediction of spatiotemporal data. Spatial Pooler (SP) is an important component of HTM, that models how neurons learn from connections and make an effective representation of input by converting binary input patterns into Sparse Distributed Representations. Current implementation of Spatial Pooler model does not support Serialization. This paper attempts to provide a best fit Serialization Model for the Spatial Pooler Software Module in .NET environment.

Keywords -- Hierarchical temporal memory, Neocortex, spatiotemporal, Spatial Pooler, Sparse Distributed Representations, Serialization.

I. Introduction

Hierarchical temporal memory (HTM), inspired by the architecture and functionality of Neocortex, is designed to learn sequences and make predictions. Vast amount of information is fed to our brain through our sensory systems which remodels raw external data from light, sound pressure, skin deformations etc. In order to organize these data and constitute our perceptibility cortical neurons forms synaptic connections to a subset of the presynaptic neurons. This learning mechanism by cortical neurons to specific input patterns and collective input specific representation of a population of neurons inspires the algorithmic property of Hierarchical temporal memory. The similarity of HTM structure with that of Neocortex starts with the cortical microcolumn of Neocortex. In HTM similar structure comprises of columns, each consisting of multiple cells. One or more regions establish a level [1]. Levels form a hierarchical structure as in Figure 1.

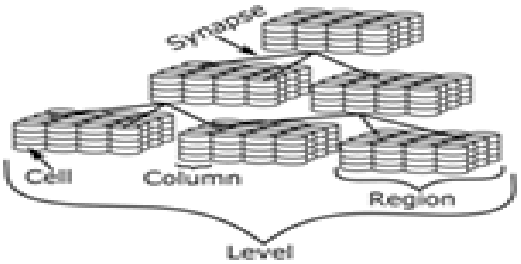

Figure. 1 Depiction of HTM Structure

Connections are formed via synapses. Proximal and Distal synapses subsequently form feedforward and neighboring connections. Two primary algorithms of the current version of HTM are Spatial Pooler(SP) and Temporal Memory(TM).The SP takes a binary input and produces a new Sparse Distributed Representation (SDR) of the input, here an SDR is a binary vector typically having a sparse number of active bits, i.e., a bit with a value of "1". Hence SP works as a mapping function that transforms input domain to a feature domain. The mapped output from Spatial Pooler in the form of SDR Data is then subsequently fed to Temporal Memory (HTM sequence Memory) as depicted in Figure 2.

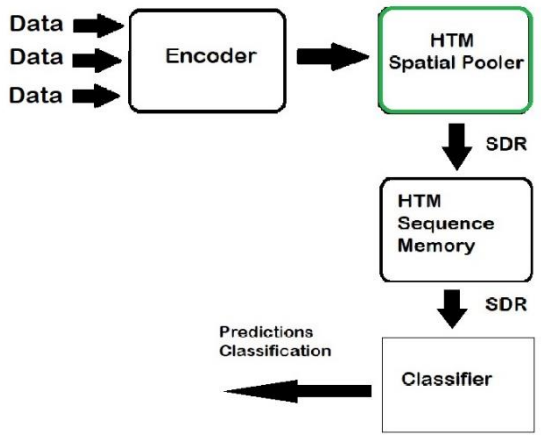

Figure. 2 Depiction of Spatial Pooler in End-End HTM System

The computational aspect of Spatial Pooler relies on Hebbian learning, homeostatic excitability control, topology of connections in sensory cortices and activity-dependent structural plasticity. The computational properties of SP include **1)**.Maintaining input space topology by mapping similar input to similar output **2**)Forming fixed sparsity representation **3**)Being robust to noise and being fault tolerant [1].

Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. In this paper we attempt to analyse the compatibility of usual serialization method with Spatial Pooler algorithm and

implement best fit model for it.The HTM Spatial Pooler software model incorporates complex data handling constituted by different type of properties. Hence it is imperative to consider powerful Serialization and formatting method for this purpose. In the first part of our paper we discuss briefly about established theoretical framework of Spatial Pooler structure and properties from the perspective of Software implementation. In the subsequent sections of this paper we try to integrate feasible serialization technique in Spatial Pooler module in .NET environment.

II. Spatial Pooler Structure and Properties

In an end-to-end HTM system the Spatial Pooler continuously transforms input patterns into SDRs. Subsequently HTM temporal memory learns sequences of these SDRs and makes predictions for future inputs. A single layer in an HTM network is organized as a collection of mini-columns each with a set of cells. The HTM neuron model integrates the dendritic characteristics of neocortex pyramidal cells with specific roles of proximal and distal dendritic segments of HTM neurons. Detected patterns on proximal dendrites lead to potential for action and establish the neuron's classic receptive region. Patterns recognized by distal synapses of a neuron serve as predictions by depolarizing the cell without specifically triggering a potential for action [2].

In HTM theory, different cells within a mini-column represent this feedforward input in different temporal contexts. SP models the synaptic development of proximal dendritic segments. Since cells in a mini column share the same classical field of feedforward reception, the SP models how to learn from this common receptive field. The SP output is the activation of the mini-columns in response to feedforward inputs.

In the neighbouring mini-columns, the SP models local inhibition. This inhibition conducts a k-winners-take-all computation. Just a small fraction of the mini-columns that have the most active inputs are active at any time. Feedforward connections to active cells will be altered at every stage in compliance with Hebbian learning laws. A homeostatic excitatory control mechanism operates on a slower time scale.Each SP mini-column forms synaptic connections to the input neurons population. We assume the input neurons are topologically organized in an input space. Using $\mathbf{x_j}$, it can be defined that where the input $\mathbf{j^{th}}$ neuron and binary $\mathbf{z_j}$ variable are to show its activation. The input space dimensionality depends on the applications. For example, for images input space is two dimensional or for scalar values it would be one dimensional. There are a variety of encoders available for various types of data. The output neurons are also organized in a different space topologically; we refer to the position of the $\mathbf{i^{th}}$ SP mini-column as $\mathbf{y_i}$ [1].

The synapses for the $\mathbf{i^{th}}$ SP mini-column are located in a hypercube of the input space centered at x^c_i with an edge length of γ. Each SP mini-column has potential connections to a fraction of the inputs in this region. These are called "potential" connections, since a synapse is only connected if its synaptic permanence is above the level of the connection. The set of possible input connections for the mini-column $\mathbf{i^{th}}$ is initialized as.Each synapse is modelled with a scalar permanence value and consider a connected synapse if its permanence value is above a threshold for the relation. The set of synapses associated with a binary matrix W is denoted as,

$$W_{ij} = 1 \quad \text{if} \quad D_{ij} \geq \theta_c$$

$$= 0 \quad \text{otherwise}$$

where $\mathbf{D_{ij}}$ gives the synaptic permanence from the j^{th} input to the ith SP mini-column. The synaptic permanence's are scalar values for potential synapses between 0 and 1, initialized to be independent and distributed identically according to a uniform distribution between 0 and 1.

$$D_{ij} = U(0,1) \quad \text{if } j \in \mathbb{\Pi}_i$$

$$= 0 \quad \text{otherwise}$$

Neighbouring SP mini-columns inhibit each other via a local inhibition mechanism. A SP mini-column becomes active if the feedforward input is above a stimulus threshold θ_{stim} and is among the top s percent of its neighbourhood. [1]

III. Serialization Methodology

In Serialization mechanism the name of the class, the public and private fields of an object, the assembly containing the class, are converted to a stream of bytes, which is then written to a data stream. Through this process the object is stored in a storage medium. When deserialized it restores the exact state of the object that was created as depicted in Figure 3.

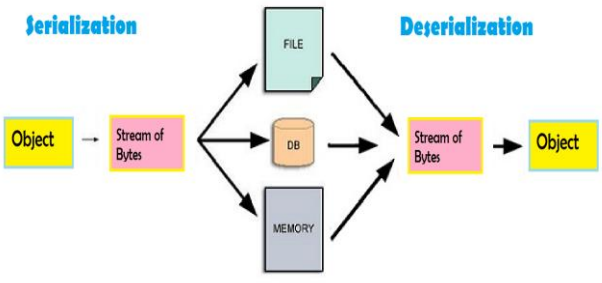

Figure 3 Serialization-Deserialization

Spatial Pooler is a class in the HTM software model that encapsulates various data properties and data methods. So, the idea is to create a serialization method that will successfully save all the properties of a particular Spatial

Pooler object to a variable or file and when called upon it will clone the same attributes to the object.

JSON Serialization : In our approach we have implemented the Serialization model primarily through enhanced JSON serialization technique. JSON (Java Script Object Notation) is a lightweight and easily understandable format for storing and transporting data. Additionally, Newtonsoft.Json package incorporates robust Serialization and Deserialization algorithm equipped to handle complex data structure through in JSON format. We have structured our Serialization Project work in three part as following :

a) Integrating Serialization and Deserialization method in Spatial Pooler Class
b) Identifying complex class properties and integrating Member Serialization attribute for relevant classes
c)Performing Unit Test to validate Serialization functionality

Serialization Method in Spatial Pooler Class :
We have implemented two methods named Serializer() and Deserializer() inside the class **SpatialPooler.cs** which is under NeCortexApi directory in HTM.
The Serializer function primarily works on the following algorithm :

1.It serializes the current instance of Object to a string through function :

```
JsonConvert.SerializeObject(this, settings)
```

2. It uses a JsonSerializerSettings named settings in order to customize the handling of serialization. Due to the constitution of complex properties and class dependencies we had to calibrate our Serializer settings in the following way so that optimum serialization and deserialization could be achieved.

JsonSerializerSettings

DefaultValueHandling = DefaultValueHandling.Include,

ObjectCreationHandling = ObjectCreationHandling.Auto,

ReferenceLoopHandling = ReferenceLoopHandling.Serialize,

ConstructorHandling = ConstructorHandling.AllowNonPublicDefaultConstructor,

TypeNameHandling = TypeNameHandling.Auto

DefaultValueHandling.Include allows default values of the Data Type like Null, Minimum Value etc to be included in the output.
ObjectCreationHandling.Auto feature enables Serializer to reuse existing objects and creates new objects when needed.
ReferenceLoopHandling.Serialize serializes loop references.
ConstructorHandling.AllowNonPublicDefaultConstructor lets Json.Net to use private default constructors when initializing objects during deserialization.
TypeNameHandling.Auto include the type name when the type of the object being serialized is not the same as its declared type.

3. The serialized values which are stored in a variable is in turn written to a .json file through File.WriteAllText() method.

4. The Deserializer() method uses the same Json Serializer Settings as discussed above. It reads the content of the .Json file through File.ReadAllText() method applies the settings and Deserializes public and private properties to a Spatial Pooler object by the method :

```
JsonConvert.DeserializeObject<SpatialPooler>
```

Integrating Member Serialization Attribute on class properties :

Spatial Pooler contains a complex class property Connection. Connection contains various local as well as class properties. Hence it is imperative for the Json serializer to know which members have the Serializable Attribute. During troubleshooting we observed that there occurred discrepancies in serializing the values of a few properties on some of the remote classes. For example, in properties like input Matrix or memory which derives from Sparse Matrix and In Memory Distribution Array. So, we had to add the attribute [JsonObject(MemberSerialization = MemberSerialization.Fields)] to some of the class member like **InMemoryDistributedDictionary** or **Sparse Object Matrix** and attribute [JsonProperty] to some of the constructor properties like "`Dimensions`" in HtmModuleTopology etc.

Unit Test : We have implemented two Uni Test cases.

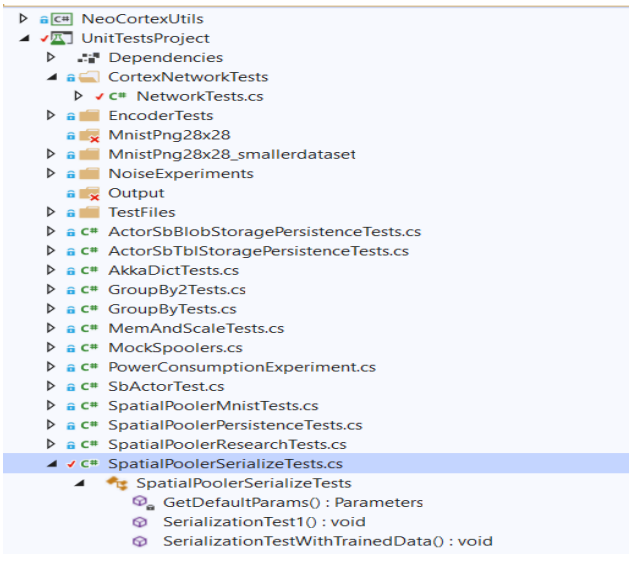

Figure 4 Test File Location

The Test File name is SpatialPoolerSerializationTests.cs which contains two Uni Tests SerializationTest1() and SerializationTestWithTrainedData() as depicted in Figure 4. For both the tests we declare some default parameters for Connection and applied the same by calling `GetDefaultParams()method.`

SerializationTest1 : This is a basic test that demonstrate Serializer functionality. In this test Parameterized Connection is applied to a new Spatial Pooler object sp1. The sp1 instance is then serialized by calling Serializer() function which is integrated in Spatial Pooler Class. The object properties are serialized in a Json file and as well as stored in a string **ser** . Subsequently the content of the file is deserialized again to a variable sp2. Sp2 can be verified to be a Spatial Pooler object by inspecting the properties in runtime. The accuracy of the serialization can be further validated by Serializing sp2 to a string des and comparing **ser** and **des** through Assert function.

SerializationTestWithTrainedData : This test runs SpatialPooler 32x32 with input of 16x16, Number of Active Column Per Inhibition area as (.02*32*32) and the same Connection Parameter.

Test runs 5 iterations and keeps stable SDR encoded sequence.After 5 steps, current instance of learned SpatialPooler (SP1) is serialized to JSON and then deserialized to second instance SP2.Second instance SP2 continues learning of the same input. Expectation is that SP2 continues in stable state with same set of active columns as SP1.

IV. Result

Both the test ran successfully as depicted in Figure 5.

Figure 5 Test Run Status

In Test1 the Serializer is successfully able to serialize the public and private properties of Spatial Pooler Object to a variable or file. Below is a snippet of some of the properties serialized to the file spTesting.json in Figure 6.


```
"connections": {
  "potentialRadius": 10,
  "potentialPct": 0.75,
  "m_GlobalInhibition": true,
  "m_LocalAreaDensity": -1.0,
  "m_NumActiveColumnsPerInhArea": 40.96,
  "m_StimulusThreshold": 0.0,
  "synPermInactiveDec": 0.01,
  "synPermActiveInc": 0.1,
  "synPermConnected": 0.1,
  "synPermBelowStimulusInc": 0.01,
  "minPctOverlapDutyCycles": 0.001,
  "minPctActiveDutyCycles": 0.001,
  "predictedSegmentDecrement": 0.1,
  "dutyCyclePeriod": 10,
  "maxBoost": 1.0,
```


Figure 6 Serialized Properties

Similarly, the Deserialization could be performed and all the properties in sp2 could be verified that with the initial Spatial Pooler object. The comparison was confirmed to be accurate by the Assert function. Fig 7 shows some of the properties deserialized to sp2 and captured in runtime.

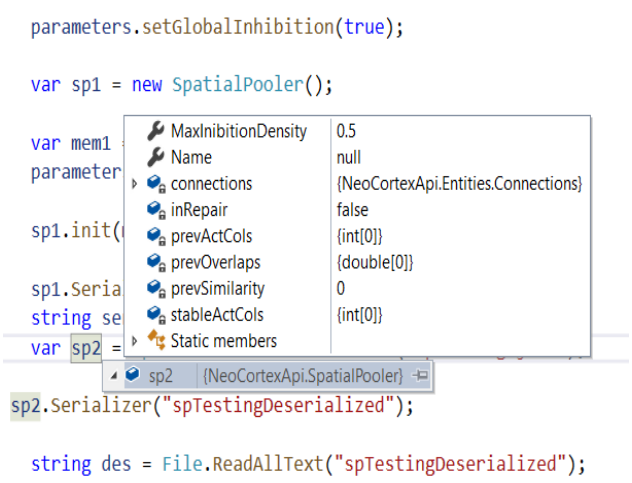

Figure 7 Deserialized Spatial Pooler Object Properties

In Test2 also it could be verified that serialization and deserialization function could be performed as per expectation. After first 5 iteration of training the Spatial Pooler it was serialized and then deserialized to a new object sp2. Sp2 was again fed into next 5 iteration and assert function verified that on both the instances same string was generated by stringyfy vector which indicates that SP2 continues in stable state with same set of active columns as SP1.

V. Conclusion

The Serialization approach proposed in this paper is able to successfully Serialize and deserialize Spatial Pooler Object and can be integrated to the HTM Module. Json Serializer Settings had to be modified in order to obtain accurate result. Also, Member Serialization attribute had to be added to some of the complex class properties. The same settings can be further calibrated in future as per requirement if there is any enhancement on the current Spatial Pooler Model for Serialization purpose.

References

[1] Y. Cui, A. Subutai and J. Hawkins, "The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding," *Front. Comput. Neurosci,* 2017.

[2] J. Mnatzaganian, E. Fokoué and D. Kudithipudi, "A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler," *Front. Robot. AI,* 2017.