

Improve the Scalar Encoder of NeocortexApi

Bishal Gautam
bishal.gautam@stud.fra-uas.de

Abstract—Encoder encodes any input to the form of SDRs, which are then processed by the neocortexapi. Neocortexapi is an implementation of Hierarchical Temporal Memory (HTM), which tries to mimic the neocortex region of brain in terms of architecture and processing. In this paper, one of the encoders, Scalar Encoder's will be described and improved. Scalar Encoder is responsible for encoding numeric or floating-point values in a series of 0's and contiguous block of 1's. Current version of Scalar Encoder is tested and improved by choosing appropriate value for parameters and by improving the interface and current logic. Proper Unit Testing had been implemented that shows the improvement of newer version of encoder over the previous version of ScalarEncoder.

Keywords—sparse distributed representations, scalar encoder, neocortex, NeocortexApi, .NET6, APIs, HTM

I. INTRODUCTION

Massive amount of sensory information is processed continuously in the neocortex region. Neocortex never stops processing the information and still work flawlessly. All the information is processed as a series of active bits passed down from the sensory organs to the neurons through the synaptic connections. This information is represented in neuron in the form of active pulse for a time instant. The brain, especially neocortex, can hold huge amount of information from life to death. This is the possible because of sparse representation of neurons for a particular input. In sparse representation, large array of bits is inactive(0's) and a very few are active(1's). Each bit carries some meaning. Two SDRs having more than a few overlaps would represent that two SDRs are semantically similar. Encoder in neocortexapi corresponds to sensory organs. Encoder encodes any input in the form of spare distributed representations.

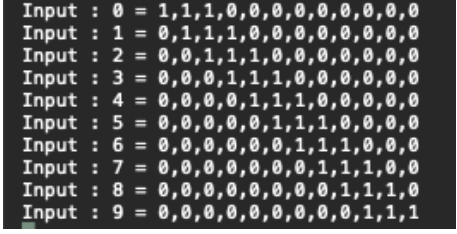
Scalar Encoder is a type of encoding technique in HTM system that encodes scalar values such as numeric or floating-point value in the series of 0's and 1's. In this type of encoder, output has 0's with adjacent block of 1's. The goal of this paper is to ameliorate the encoding of scalar values. Enhancement is done on preexisting scalar encoder by setting appropriate parameters and by adding various functionalities. Proper unit testing had been implemented that compares and show the improvement of newer version of ScalarEncoder over the older version.

II. MATERIALS AND METHODS

A. Encoder Overview

First step of using neocortexapi is to transform the data to a form (SDRs) so that it can be digested for further processing

(i.e. by Spatial Pooler). Encoder is responsible for converting the data to a SDRs representation. There are various types of encoders such as Geospatial encoder, Image encoder, Scalar Encoder, DateTime Encoder, etc. Geospatial encoder takes geometric coordinates values and convert them to SDRs. Two objects can be said to be close to each other geographically if their SDRs overlap over some threshold. Similarly in Scalar Encoder, two values SDRs would be similar if they are equal or close to each other.



Input : 0	=	1,1,1,0,0,0,0,0,0,0,0,0
Input : 1	=	0,1,1,1,0,0,0,0,0,0,0,0
Input : 2	=	0,0,1,1,1,0,0,0,0,0,0,0
Input : 3	=	0,0,0,1,1,1,0,0,0,0,0,0
Input : 4	=	0,0,0,0,1,1,1,0,0,0,0,0
Input : 5	=	0,0,0,0,0,1,1,1,0,0,0,0
Input : 6	=	0,0,0,0,0,0,1,1,1,0,0,0
Input : 7	=	0,0,0,0,0,0,0,1,1,1,0,0
Input : 8	=	0,0,0,0,0,0,0,0,1,1,1,0
Input : 9	=	0,0,0,0,0,0,0,0,0,1,1,1

Fig. 1. Scalar Encoding of scalar values with total number of bits 12 and 3 number of active bits

Two encodings are said to be similar if they have more overlapping active bits. As shown in figure 1, we can see that the encoding of 1 and 2 are nearly similar to each other. Likewise, the encoding of 6 and 7 are nearly similar to each other. But the encoding of 1 and 7 is totally different than the encoding of 1 and 2. This is due to the fact that 1 and 2 has less difference than 1 and 7 and hence 1 and 2 are more similar both semantically and with SDRs overlap.

There are some important aspects that need to be considered when encoding data. They are described as [1]:

- For data which are semantically similar, encoding should result in SDRs with overlapping active bits
- Encoding should be deterministic, meaning that same input should always produce the same SDR as output.
- The output should have the same dimensionality (total number of bits) for all inputs.
- The output of encoding should have similar sparsity for all input. Output should have enough one-bits to handle noise and subsampling.

B. Parameters and Terms for Scalar Encoder

Before encoding scalar values, scalar encoder is initialized with parameter dictionary. The parameters which the encoder depends on can be explained as:

- N: Total number of bits to represent the input data
- W: Number of active bits (1's) to represent the input data. W should be odd to avoid centering problem.
- MinVal: Minimum Value that the SDR can represent without clipping
- MaxVal: Maximum Value that the SDR can represent without clipping. For periodic input, input data to encode should be strictly less than MaxVal
- Periodic: Boolean Value, if true, representation is repeated after certain interval
- ClipInput: If true, value less than the minimum has SDR similar to minimum value and values more than the maximum has SDR similar to maximum value for periodic data
- Resolution: For a particular value of resolution, input separated by more than or equal to resolution would have different representations. For e.g., if resolution = 0.5, then 4 and 5 will have different encoding but 4.1 and 4.4 could have similar encoding.
- Radius: For a particular value of radius, input separated by the radius would have completely different representations with non-overlapping active bits.
- Name: It is an optional string that will become a part of description

N, Resolution and Radius are three mutually exclusive parameters. Exactly one of N, Radius or Resolution should be given for the input space, rest of the two should be "0" or not set on the input space. Rest of the two parameters will be calculated internally based on the parameters supplied to the scalar encoder.

Total Buckets represent the total number of input spaces that can be represent as distinct SDRs. For example, for a particular setting if the bucket size is 14 then the encoder can encode 14 distinct SDRs. The equation to calculate the Bucket size for an encoder is:

$$\text{Total Buckets} = (N - W + 1) \quad (1)$$

III. TEST CASES WITH RESULTS

Current version of Scalar encoder has some defects, in this part we will be discussion the pitfalls of currently implemented scalar encoder.

After finding the pitfalls, we will be experimenting to find the correct solution with a way to improve current version of scalar encoder.

A. Test Case generating the bucket size problem

In the first experiment, we are tried to regenerate the error of bucket size problem with the predefined parameters. For the experiment, we have:

- a. N = 10
- b. W = 3
- c. MinVal = 0
- d. MaxVal = 9
- e. Periodic = False
- f. ClipInput = True

The output with the above defined parameter is shown as:

Input : 0	=	1,1,1,0,0,0,0,0,0,0
Input : 1	=	0,1,1,1,0,0,0,0,0,0
Input : 2	=	0,0,1,1,1,0,0,0,0,0
Input : 3	=	0,0,1,1,1,0,0,0,0,0
Input : 4	=	0,0,0,1,1,1,0,0,0,0
Input : 5	=	0,0,0,0,1,1,1,0,0,0
Input : 6	=	0,0,0,0,0,1,1,1,0,0
Input : 7	=	0,0,0,0,0,0,1,1,1,0
Input : 8	=	0,0,0,0,0,0,0,1,1,1
Input : 9	=	0,0,0,0,0,0,0,0,1,1

Fig.2. Output of Scalar Encoder with similar SDRs

The total number of different encodings (buckets) that our configuration can represent is $(N - W + 1) = (10 - 3 + 1) = 8$ for a Resolution of 1. We need to represent data from 0 to 9, which in total is 10. So, we need 10 different representations. Because in our case, the number of encodings that an encoder can encode distinctly is 8 (available bucket 8), which is less than the required distinct encoding (i.e., 10 or available bucket needs to be 10 at least), this causes an overlap in SDR of input space. In the above case, overlap occurs for the input of 2 and 3.

B. Test Case generating the bucket size problem – Second

In the test case A, the overlapping was clearly due to the low number of bucket size. So, for our second experiment, we increased the value of N to 11, we have:

- a. N = 11
- b. W = 3
- c. MinVal = 0
- d. MaxVal = 9
- e. Periodic = False
- f. ClipInput = True

The output with the above defined parameter is shown as:

Input : 0	=	1,1,1,0,0,0,0,0,0,0,0
Input : 1	=	0,1,1,1,0,0,0,0,0,0,0
Input : 2	=	0,0,1,1,1,0,0,0,0,0,0
Input : 3	=	0,0,0,1,1,1,0,0,0,0,0
Input : 4	=	0,0,0,0,1,1,1,0,0,0,0
Input : 5	=	0,0,0,0,0,1,1,1,0,0,0
Input : 6	=	0,0,0,0,0,0,1,1,1,0,0
Input : 7	=	0,0,0,0,0,0,0,1,1,1,0
Input : 8	=	0,0,0,0,0,0,0,0,1,1,1
Input : 9	=	0,0,0,0,0,0,0,0,0,1,1

Fig. 3. Output of Scalar Encoder for the given parameter with similar SDRs

The total number of distinct that our configuration can represent is $(N - W + 1) = (11 - 3 + 1) = 9$ (i.e., the available bucket size is 9). Since we need to represent data from 0 to 9 as a distinct encoding, we need 10 different encodings (i.e., the required bucket size is 10). Since the number of available bucket is less than the required bucket. This cause in an overlap in SDR of input space. In our case, 4 and 5 input overlapped.

C. Test Case for the solution of the given predefined parameter

We experimented by increasing the value of N again so as to increase the number of distinct encodings that a encoder can encode (bucket size). The parameters value is:

- N = 12
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

The output with the above defined parameter is shown as:

```
Input : 0 = 1,1,1,0,0,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0,0,0
Input : 3 = 0,0,0,1,1,1,0,0,0,0,0,0
Input : 4 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 5 = 0,0,0,0,0,1,1,1,0,0,0,0
Input : 6 = 0,0,0,0,0,0,1,1,1,0,0,0
Input : 7 = 0,0,0,0,0,0,0,1,1,1,0,0
Input : 8 = 0,0,0,0,0,0,0,0,1,1,1,0
Input : 9 = 0,0,0,0,0,0,0,0,0,1,1,1
```

Fig. 4. Output of the Scalar Encoder after increasing N for appropriate bucket size

The total number of distinct encodings that our configuration can represent is $(N - W + 1) = (12 - 3 + 1) = 10$, (i.e., the available bucket size is 10). Since we need to represent data from 0 to 9, we need 10 distinct SDRs representations. The number of available distinct representations that our encoder can encode is equal to the number of required distinct representations. This causes a discrete SDR representation of different input.

D. Test Case for the Solution of Non-Distinct Encodings

When the value of N is used as input to set encoder settings (Radius and Resolution values are zero or not set), we found out that some values of N were too low that resulted in similar SDRs for distinct inputs. In order to avoid the problem, argument exception check had been implemented. This makes sure that either inputs would have distinct encoding or argument exception is thrown in case the present setting of encoder could result a non-distinct encoding for the given input space.

$$requiredN = (int) Math.Ceiling(W + MaxVal - MinVal) \quad (2.a.)$$

$$requiredN = (int) Math.Ceiling(MaxVal - MinVal) \quad (2.b.)$$

If the encoder settings are set for periodic input, i.e., Periodic is True in encoder setting then equation 2.b. is used

as comparison for required value of N. If the encoding should be done for non-periodic inputs, i.e., Periodic is False in encoder setting then equation 2.a. is used as comparison with the present value of N. If present value of N is less than requiredN then argument exception is thrown. This is shown in figure 5. Proper parameters testing is shown on ScalarEncoder Unit test file.

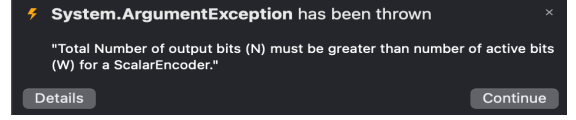


Fig. 5. Improved Scalar Encoder throws exception when the value of N result in non-distinct encoding.

When the value of Resolution is used as input to set encoder settings (i.e., Radius and N is zero), we found out that some values of Resolution were too high that resulted in similar SDRs for different inputs. In order to avoid the problem of similar encoding for different input, proper argument exception check had been implemented. This makes sure that when the value of Resolution is more than required (i.e., 1), argument exception would be thrown. If the value of Resolution is proper, distinct encoding is outputted for the given input space. The output of encoder that tries to generate non-distinct encoding provided the value of Resolution is shown in figure 6. Proper parameters testing is shown on ScalarEncoder Unit test file.

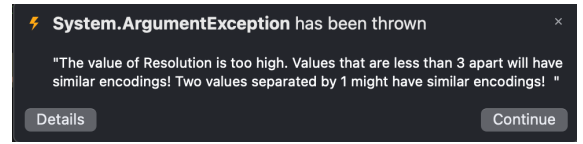


Fig. 6. Improved Scalar Encoder throws exception when the value of Resolution is too high (prevents non-distinct encoding)

When the value of Radius is used as input to set encoder settings (i.e., Resolution and N is zero), we found out that some values of Radius were too high that resulted in similar SDRs for different inputs. Argument exception check had been implemented for Radius so that either distinct encoding is output for the given input space or argument exception is thrown if the inputted settings could result in non-distinct encoding for the given input space.

$$Resolution = Radius / W \quad (3)$$

Equation 3 is used to calculate the value of Resolution for a given Radius. Then check is implemented with the calculated value of Resolution. If Resolution is less than or equal to 1, distinct encoding is done by the Encoder, else argument exception is thrown. This is shown in figure 7. Proper parameters testing is shown on ScalarEncoder Unit test file.

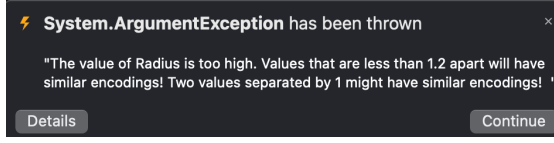


Fig. 7. Improved Scalar Encoder throws exception when the value of Radius result in non-distinct encoding.

E. Test Case generating the OverflowException

Instead of relying on N for encoding, this time we choose different values of Resolution for Encoding.

- Resolution = 1 to 10
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

While using the above parameters for encoding, we observed System.OverflowException. This is shown in the figure 8.a.

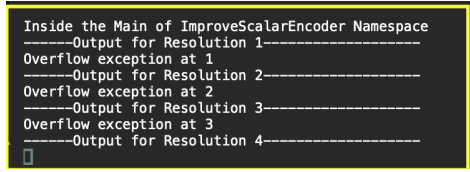


Fig. 8.a. Output showing Exception for a given set of parameters

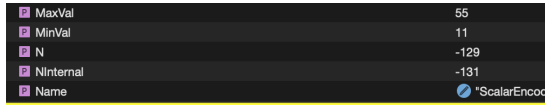


Fig. 8.b. Values of N when the Exception occurred

After observing the values of local variable using breakpoints, we found out that the value of N to be negative as shown in figure 8.b. This should be the reason for the overflow exception.

F. Test Case generating IndexOutOfRangeException

When we choose Radius for encoding, clearly as viewed on the figure 9, at certain value of Radius, IndexOutOfRangeException occurred. Proper parameters testing is shown on ScalarEncoder Unit test file.

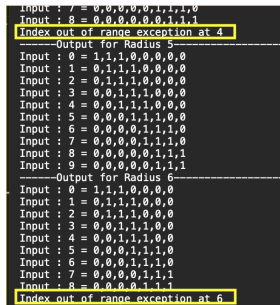


Fig. 9. IndexOutOfRangeException for some input space

G. Test Case explaining the OverflowException

As a comment on test case E, we worked with various values of Resolution (without setting N or Radius). After experimenting and through debugging we found that System.OverflowException occurred because the default Radius configuration was negative (-1), which was set inside the EncoderBase.cs file. This resulted in negative value of N. Since the output array was initialized using the equation 4 as:

$$\text{Output} = \text{int}[N] \quad (4)$$

This resulted in an OverflowException.

Setting the default value of Radius to 0, which was previously -1 inside the Initialize method of EncoderBase.cs file, N was no longer negative and OverflowException was solved. After solving this issue, IndexOutOfRangeException was observed in some input space. Experiments is shown on the unit test file for improved scalar encoder.

H. Test Case explaining IndexOutOfRangeException

In the test case H, we solved the problem of OverflowException. After that we need to solve the problem of IndexOutOfRangeException. By experimenting with various inputs, we find out that the exception was due to the result of minbin or maxbin value, which denotes the starting position and ending position of the active bits.

We first find the centerbin and on adding padding to the left and right of the centerbin we get the values of minbin and maxbin. minbin represent the index position to start the encoding and maxbin represent the index position of the end of encoding for a value of N. The values of parameters used for encoding for this test case is:

- Resolution = 1 to 10
- W = 3
- MinVal = 13
- MaxVal = 29
- Periodic = False
- ClipInput = True

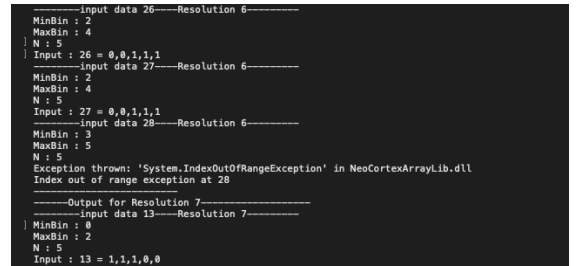


Fig. 10. Experiment showing IndexOutOfRangeException

Clearly in figure 10, in case of exception, minbin is 3 and maxbin equals 5. The value of N is 5 for this experiment. Minbin and maxbin represent the index position and index position starts at 0. This means that our encoder tries to set the data from N[3] to N[5]. The value of N is 5, because of this we can only go up to N[4] as indexing starts at 0 in c-sharp and most other programming language. This resulted in IndexOutOfRangeException.

I. Test Case Solving IndexOutOfRangeException

Experimenting with different values and by using breakpoints, we found out that there might be issue in the value of center bin or N. Center bin value after experimenting seems to be fine, logic seemed okay. So, we then looked for the value of N. Because of low value of N, the error seems to persist in some input space. To solve this, we changed the value of N in line 157 of ScalarEncoder.cs file.

$$N = (int)(nFloat) \quad (5)$$

$$N = (int)Math.Round(nFloat) \quad (6)$$

The value of N was changed from equation 5 to equation 6. This makes sure that the decimal values are rounded to nearest whole number. This increased the value of N for some input space.

For the input parameters of test case H, the problem seems to be solved.

```

-----input data 26-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 6
Input : 26 = 0,0,1,1,1,0
-----input data 27-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 6
Input : 27 = 0,0,1,1,1,0
-----input data 28-----Resolution 6-----
MinBin : 3
MaxBin : 5
N : 6
Input : 28 = 0,0,0,1,1,1
-----input data 29-----Resolution 6-----
MinBin : 3
MaxBin : 5
N : 6
Input : 29 = 0,0,0,1,1,1

```

Fig.11.a. Experiment solving IndexOutOfRangeException for the given parameter

But still the problem seems to be unsolved for some different input space. The parameters which resulted IndexOutOfRangeException were:

- Resolution = 1 to 10
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

Figure 11.b. shows the output of the above parameters.

```

-----input data 6-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 4
Input : 6 = 0,1,1,1
-----input data 7-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 4
Input : 7 = 0,1,1,1
-----input data 8-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 4
Input : 8 = 0,1,1,1
-----input data 9-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 4
Exception thrown: 'System.IndexOutOfRangeException' in NeoCortexArrayLib.dll
Index out of range exception at 9

```

Fig. 11.b. IndexOutOfRangeException still persists

After working with different settings, we found out that changing the value of N in line 157 of ScalarEncoder.cs to equation 7 solved the issue of IndexOutOfRangeException.

$$N = (int)Math.Ceiling(nFloat) \quad (7)$$

```

-----input data 6-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 5
Input : 6 = 0,1,1,1,0
-----input data 7-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 5
Input : 7 = 0,1,1,1,0
-----input data 8-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 5
Input : 8 = 0,1,1,1,0
-----input data 9-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 5
Input : 9 = 0,0,1,1,1

```

Fig. 11.c. IndexOutOfRangeException solved for any input space

Using Math.Ceiling increased the value of N in case of any decimal value of nFloat. This resulted in increase in the number of positions for output bits. Due to this, calculated maxbin was always less than or equal to N.

J. Test Case showing the usecase of Interactive Console

When the provided configuration leads in non-distinct encoding, we implemented some functions inside Program.cs of our project in our project that makes possible to take input from the user. If user selects the option yes, then the value of N or Radius or Resolution is updated that leads to distinct encoding. If the user selects no option, then System.ArugumentException is thrown by the ImprovedScalarEncoder. Example is shown in figure 12.

```

The current value of Radius will encode two values in similar encoding if two value are separated by less than 1.2.
Are you sure you want to proceed with the entered value of Radius or do you want to update Radius for a Resolution of 1 ?
Enter [yes] if you would like to update Radius. [no] if you don't.
Update the Value ? : 

```

Fig.12. Example case of interactive console when the value of Radius leads to non-distinct encoding

K. Test Case making Scalar Encoder callable using command line argument

In improved version of the ScalarEncoder, it can take input from both the dictionary settings and command line arguments. To work with the command line arguments, arguments passed on the Main function of our program can be directly passed to the ScalarEncoder. Because of function overloading, method that processes the command line arguments will be called instead of the method that takes dictionary as input for encoding.

- The command line arguments can be set on visual studio by going to the properties of our current solution and adding arguments.
- Command line arguments can also be passed directly from the console and the program can be run as shown below:

Listing 1. Command line code to encode the given input

```
dotnet run --project path-to-project-file --n 14 --w 3 --minval 1 --maxval 8 --periodic false --clipinput true
```

In our case the path to the project file would be:

```
neocortexapi/source/ScalarEncoderImproved/ScalarEncoderImproved.csproj
```

User can create their own project and implement Scalar Encoder inside. If the user in the same folder as project file, i.e., inside their project folder where .csproj file is present, then user do not need to specify the project option while encoding using command line arguments.

```
bisalgt@bisalgt neocortexapi % dotnet run --project source/Samples/NeoCortexApiSample/NeoCortexApiSample.csproj --n 14 --w 3 --minval 1 --maxval 8 --periodic false --clipinput true
Inside the Main of ImproveScalarEncoder Namespace
Input : 1 = 1,1,1,0,0,0,0,0,0,0,0,0,0,0
Input : 1.6363636363636362 = 0,1,1,1,0,0,0,0,0,0,0,0,0,0
Input : 2.2727272727272725 = 0,0,1,1,1,0,0,0,0,0,0,0,0,0
Input : 2.9090909090909087 = 0,0,0,1,1,1,0,0,0,0,0,0,0,0
Input : 3.5454545454545455 = 0,0,0,0,1,1,1,0,0,0,0,0,0,0
Input : 4.181818181818182 = 0,0,0,0,0,1,1,1,0,0,0,0,0,0
Input : 4.381818181818183 = 0,0,0,0,0,0,1,1,1,0,0,0,0,0
Input : 5.454545454545455 = 0,0,0,0,0,0,0,1,1,1,0,0,0,0
Input : 6.090909090909092 = 0,0,0,0,0,0,0,0,1,1,1,0,0,0
Input : 6.727272727272728 = 0,0,0,0,0,0,0,0,0,1,1,1,0,0
Input : 7.363636363636365 = 0,0,0,0,0,0,0,0,0,0,1,1,1,0
bisalgt@bisalgt neocortexapi %
```

Fig. 13. Output of using command line for encoding

Proper exception handling is implemented if the supplied argument is not in a correct order or format.

L. Test Case Implementing the Unit Tests

UnitTests had been implemented that shows the difference between the unimproved version of Scalar Encoder and improved version of Scalar Encoder. The two versions are checked with similar parameter. Clearly, improved version handles all the exceptions making the Scalar Encoder deterministic. Improved version makes sure that only distinct encoding is done, configurations that leads to non-distinct encoding are thrown as exception. The unit test can be found on ScalarEncoderImprovedTests.cs file.

IV. LINKS TO THE UNIT-TEST FILE AND PROGRAM FILES

1. ScalarEncoderImprovedTests.cs : <https://github.com/bisalgt/neocortexapi/blob/master/source/UnitTestsProject/EncoderTests/ScalarEncoderImprovedTests.cs>
2. ScalarEncoderImproved.cs : <https://github.com/bisalgt/neocortexapi/blob/master/source/NeoCortexApi/Encoders/ScalarEncoderImproved.cs>
3. Program.cs : <https://github.com/bisalgt/neocortexapi/blob/master/source/ScalarEncoderImproved/Program.cs>

V. CONCLUSION

Working with the unimproved Scalar Encoder, we found various exceptions, errors and misconfigurations. We implemented proper logic and corrected the exception handling in the improved version of Scalar Encoder.

If the given encoder settings resulted in non-distinct encoding, then exception is thrown to the user with a helpful message. This helps user to debug the program to generate a distinct encoding. If the user wishes to get a distinct encoding, they can also update the value of parameter from the intuitive terminal by selecting either yes or no. We implemented proper exception handling so that N, Radius and Resolution are always mutually exclusive. Implementation of how N is calculated is changed. This makes sure that there are required number of total bits for scalar encoder for distinct encoding of given input space.

Overall, the improved version only encodes the input data that can result a distinct encoding. If distinct encoding cannot be done by the encoder, then exception is thrown, making sure only distinct encoding passes.

VI. REFERENCES

- [1] Purdy, S., "Encoding Data for HTM Systems," <https://arxiv.org/abs/1602.05925>, 2016.
- [2] Hawkings, J. & Ahmad, S., "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex", <https://arxiv.org/abs/1511.00083v2>, 2015