Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

Implementation of Scalar Encoder in HTM

Komala Balakrishna
komala.balakrishna@stud.fra-uas.de

Prajwal Nagaraja
prajwal.nagaraja@stud.fra-uas.de

Sahana Prasad
sahana.prasad@stud.fra-uas.de

***Abstract*—Scalar Encoder is one of the encoding techniques and is a part of Hierarchical Temporal Memory (HTM). HTM is a machine intelligence technology which is trying to imitate the process and architecture of neocortex. The main purpose for scalar encoder is to encode numeric or floating-point value into an array of bits, where the output has 0's with an adjacent block of 1's. The location of the block of 1's varies continuously depending on the input value.**

Keywords- HTM, neocortex, numeric, array, block

I. INTRODUCTION

Scalar encoder is a part of Hierarchical Temporal Memory (HTM) systems. For solving problems like classification, prediction and anomaly detection for a wide range of datatypes, HTM provides a very flexible and biologically precise framework. The input for HTM systems should be in the form of Sparse Distributed Representations (SDRs). SDR consists of a large array of bits in which most of the bits are zeros and few bits are ones.

The encoders are used to covert the data input into SDR in HTM systems. The encoders also decide which bits should be ones and which bits should be zeros in the output for any given input value in such a way that it captures the important semantic characteristics of data. Hence if the inputs are alike, then output is more likely to have highly overlapping SDRs.

II. METHODOLOGY

Encoder should create SDRs, no matter what it represents, that have a fixed number of bits 'N' and fixed number of active (1's) bits 'W'. What do you know which values are perfect for N and W?
We cannot be a big fraction of N, to preserve the properties that come from sparsity. But if W is too small then we lose the properties resulting from a distributed representation.
There are a number of specific aspects to consider when encoding the data:

1. Semantically related data can trigger SDRs with active bits overlapping.
2. The same input should always be generating the same SDR as output.
3. Of all inputs the output will be of the same dimensionality (total number of bits).
4. The output should be of equal sparsity for all inputs and should have ample one-bit to accommodate noise and subsampling

You need to understand the aspects of your data to construct an efficient encoder that can lead to similarity. In the hearing frequency example above, the encoder was programmed to have similar pitch sounds but did not take into consideration how noisy the sounds were, which would require a different approach. The first step to design an encoder is to decide any element of the data you wish to capture.

The main features for sound may be pitch and amplitude; for dates, it may be whether it's a weekend or not. The encoder will create overlapping representations for inputs that are identical in one or more of the chosen data characteristics.

Encoders should be deterministic so that each time the same output is generated from same input. With encoded representations there is a shift in values, hence without this property there will be a redundancy in the sequence learned in HTM. Stop building random or adaptive-element encoders.

An encoder's output also must generate the same number of bits for each of its inputs. Using a bit-by-bit assumption, SDRs are compared and worked so that a bit with a certain "value" is always in the same place. If the encoders provided different SDR bit lengths, comparisons and other operations would not be possible.

Furthermore, enough one-bits must be sufficient to accommodate noise and sub-sampling. A general rule of thumb is to have one bit of at least 20-25. Encoders which has less than 20 one bits in representation will not function well in HTM systems, because of noise and non- determinism they will be prone to errors.

When we construct an encoder implementation, we first divide the range of values into buckets and then map the buckets into a collection of active cells

1) Choose the range of values i.e. MinVal and MaxVal.
2) Range = MaxVal – MinVal.
3) The number of bits that are set to encode a single value the ‘width’ of output signal ‘W’ should be chosen.
4) Total number of bits in the output ‘N’ used for representation should be selected.
5) The resolution should be calculated based on periodic or non-periodic parameter.
6) Creating the encoded representation by beginning with N unset bits and then setting the W consecutive bits to active from index i.

III. TEST CASES WITH RESULTS

1. Test Case to encode Days of Week.

We know that there are seven days in a week namely Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday, in order to encode each day, we need seven different representation.

The encoding method would be here periodic since the days would repeat. Remember that there are four parameters to this encoding scheme: minimum value, maximum value, number of bits (N) and number of active bits (W).

1) This MinVal is 0 (Sunday) and the MaxVal 6 (Saturday).
2) The range is calculated with the formula MaxVal – MinVal = 7.
3) The number of bits that are set to encode a single value the ‘width’ of output signal ‘W’ used for representation is 3.
4) Total number of bits in the output ‘N’ used for representation is 8.
5) We are choosing the value of N=9 and W=3 to get the desired output which shifts between Monday to Sunday like shown below:

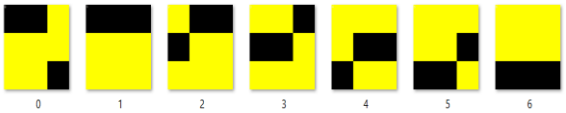
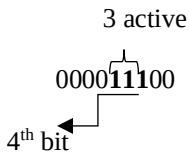


Fig.1. Expected Output of Days of Week

6) If we choose any other values for N and W for example N=8 and W=3 then it does not match the expected output.

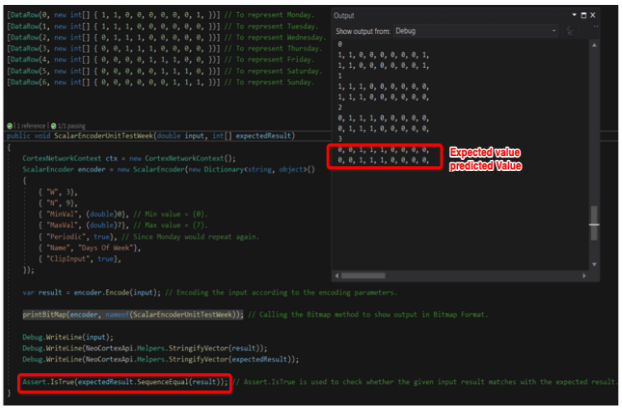
Fig.2. Output of Days of Week by altering the value of N and W

7) This example is periodic because the days of the week keep repeating for every 8th value, so resolution has to be calculated based on formula Range/N = 0.77.
8) The representation will be 9 bits with 3 consecutive active bits starting at the 4th bit as shown below:



A scalar encoder would produce a reasonable representation, except that for Saturday and Sunday there would be little to no variation in the encoding, because they are at opposite ends of the spectrum.

Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.
After setting all the parameter values run the program, the output images will be captured and saved in a folder it will show how the shifting is happening for every days of week.



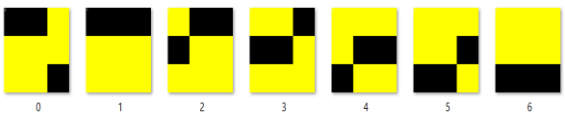


Fig.3. Output of Days of Week

2. Test case to encode ASCII codes.

ASCII (American Standard Code for Information Interchange) is an accepted format to use in computer as computer can only understand numbers.

There are 128 ASCII codes in between 0 to 127 to represent various characters which include upper-case and lower-case letters, numbers, punctuation symbols and others in the form of numeric code.

As every ASCII code is unique the parameters N, W should be chosen in such a way that, for every ASCII code there should be a shift which is from MinVal to MaxVal. Steps to achieve encoding for ASCII codes is mentioned below:

1) Since the ASCII codes are in between 0 to 127 MinVal = 0 and MaxVal = 127
2) Computing the Range = MaxVal – MinVal which is equal to 127.
3) Choosing the value of 'W' and 'N' in such way that there should be a shift for ASCII value in the output.
4) Hence the number of bits that are set to encode a single value the 'width' of output signal 'W' used for representation is 21.
5) Total number of bits in the output 'N' used for representation is 148.
6) We are choosing the value of N=148 and W=21 to get the desired output which shifts between 0 to 127 like shown below:

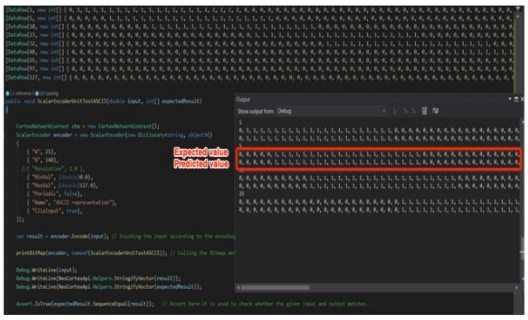


Fig.4. Expected Output of ASCII code.

7) If we choose any other values for N and W for example N=146 and W=23 then it does not match the expected output which is shown below:

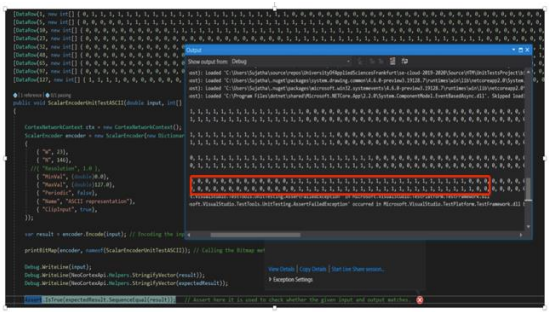
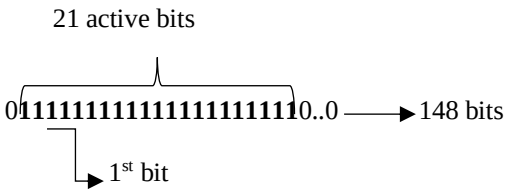


Fig.5. Output of ASCII code when value of N and W is altered

8) As ASCII code is non-periodic, resolution is calculated which is 1.
9) In the output to know from which bit the ones start, below formula will be used to calculate which is:

$$\left(\frac{[(\text{Input}-\text{MinVal}) + (\text{Resolution}/2)]}{\text{Resolution}}\right) + \text{Padding}$$

10) For example, ASCII code 1 is taken to calculate the same where input is 1, MinVal is 0 Resolution is 1 and padding is -0.5. After calculating using the above formula the bit position is 1
11) Hence, ASCII code 1 will be represented with 148 bits where 21 are active bits starting at 1st bit position which can be seen below.



Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.
After setting all the parameter values run the program, the output images will be captured and saved in a folder. Bitmap method is executed in the code to produce these data in 2D Bitmap format.

There is a 1-bit shift in every ASCII code which is in between 0 to 127, hence most of the bit overlap in adjacent ASCII codes as we can see in the below figure Fig.6(Output of ASCII codes) which is the snapshot of output of encoded ASCII codes.

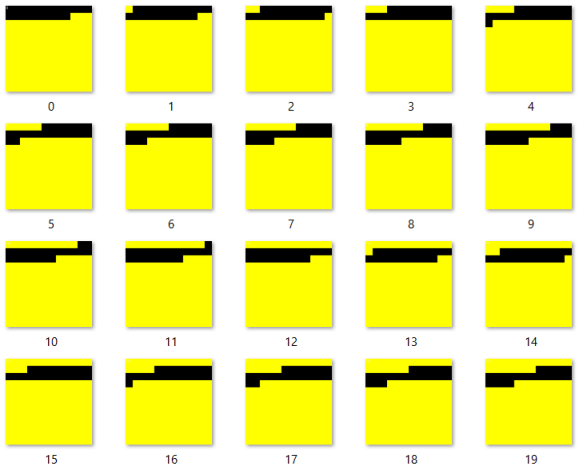


Fig.6. Output of ASCII codes

3. Test Case to encode basic RGB colors

RGB stands for Red, Green and Blue. RGB is a primary color that can be combined in various forms to produce group of colors. In this example we are going to encode 13 basic colors (including 3 primary colors) that is produced as subset of primary colors.

Each of these basic colors have its own RGB decimal code from 0-255, based on which they can be differentiated [6]. For example, if a basic color has RGB decimal code of (238,130,238) respectively, then this implies that Red is maximum, Green is medium, and Blue is maximum [6]. This results in a shade of Violet.

To design and encode these basic colors we are considering MaxVal = 16, MinVal = 1 because there are 16 colors in total.

Below are the steps to encode values with this approach:

1) To design and encode these basic colors we are considering MaxVal = 16, MinVal = 1 because there are 16 colors in total.
2) We are calculating the value of Range = MaxVal – MinVal. So, Range = 15.
3) The number of active bits is W = 7 for each representation.
4) The total number of bits N is considered as N = 22
5) We are choosing the value of N=22 and W=7 to get the desired output which is shown below:

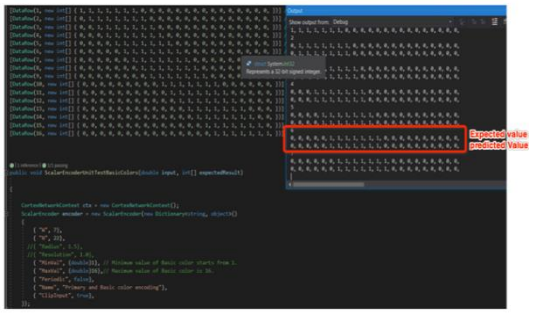


Fig.7. Expected Output of RGB.

6) If we choose any other values for N and W for example N=20 and W=9 then it does not match the expected output which is shown below:

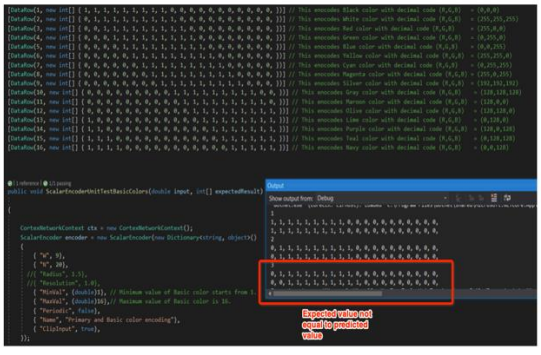


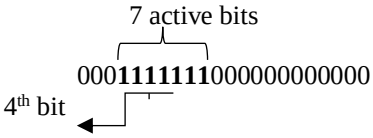
Fig.8. Output of RGB when value of N and W is altered

7) As RGB representation is non-periodic the resolution is calculated based on formula Range/N-W. Hence resolution is 1.
8) To know from which bit the active bits start, below mentioned formula can be used:

$$\left[\frac{[(\text{Input-MinVal}) + (\text{Resolution}/2)]}{\text{Resolution}}\right] + \text{Padding}$$

9) For example, In RGB Value 4 is taken to calculate the same where input is 4, MinVal is 1 Resolution is 1 and padding is 3. After calculating using the above formula and then subtracting with the halfwidth we get the bit position which is 4.

The proposed values will produce encoded data of 22 bits with 7 consecutive active bits starting from 4th bit as shown below-



Once the output for encoded data is obtained for all the 16 colors. Bitmap method is executed in the code to produce these data in 2D Bitmap format.

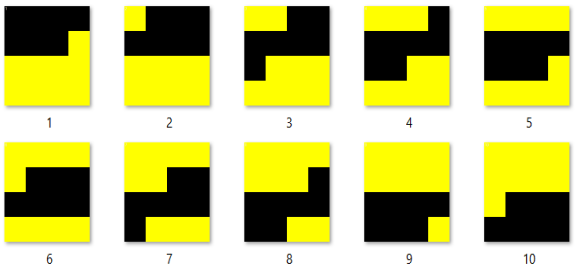


Fig.9. Output of basic RGB colours

4. Test Case to encode Drumbeats according to MIDI level 1 devices.

MIDI stands for Musical Instrument Digital Interface. General MIDI is one of the recognized platforms for defining list of sounds for different instruments and channels [5]. According to general MIDI level 1 devices, channel 10 is exclusively dedicated for drum sounds with drum keys varying from 35-81[5].
The encoding of these drum sounds must be done uniquely from key 35 – 81.
Below are the steps to encode values with this approach:

1) We are considering MaxVal = 81 and MinVal = 35.
2) The range here would be MaxVal - MinVal = 46.
3) The number of active bits is W= 7 for each representation.
4) The total number of bits N is represented as 53.
5) We are choosing the value of N=53 and W=9 to get the desired output which is shown below:

Fig.10. Expected Output of Drumbeats

6) If we choose any other values for N and W for example N=50 and W=9 then it does not match the expected output which is shown below:

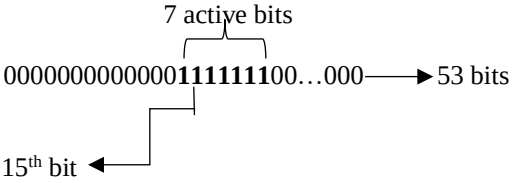
Fig.11. Output of Drumbeats when value of N and W is altered

7) As this example is for non-periodic, then resolution is calculated to be 1
8) The exact starting position of active bits can be calculated by the formula:

$$\left(\frac{[(\text{Input-MinVal}) + (\text{Resolution}/2)]}{\text{Resolution}}\right) + \text{Padding}$$

9) For example, In Drumbeats Value 49 is taken to calculate the same where input is 49, MinVal is 35, Resolution is 3 and padding is 3. After calculating using the above formula and then subtracting with the halfwidth we get the bit position which is 15.

The proposed value will give result of 53 bits with 7 consecutive active bits starting from 15th bit as shown below-



Once the output for encoded data is obtained for all the drum keys from 35-81, Bitmap method is executed in the code to produce these data in 2D Bitmap format. The following 2D bitmap is shown widely for drum keys from range 43-58.

Fig.12. Output of Drumbeats

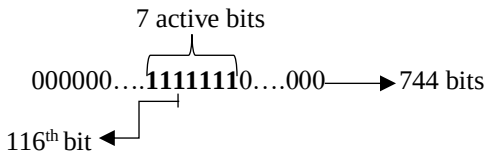
5. Test case to encode power consumption for the data available.

This test case will enable us to encode power consumption (In Kilowatts) for the data available for 1 particular month. The data is further divided into date and time (24 hours clock). In total 744 readings are available for the entire month hence the encoding parameters should be selected in such a way that all 744 readings should be represented uniquely [7]. Steps to achieve the same is mentioned below:

1) The MinVal for this is 4.7 and the MaxVal for this 81.9
2) Computing the Range = MaxVal – MinVal which is equal to 77.2.
3) The number of bits that are set to encode a single value the 'width' of output signal 'W' used for representation is 7.
4) Total number of bits in the output 'N' used for representation is 744.
5) As power consumption is non- periodic, resolution is calculated which is 0.104.
6) In the output to know from which bit the ones start, below formula will be used to calculate which is:

$$\left(\frac{[(\text{Input-MinVal}) + (\text{Resolution}/2)]}{\text{Resolution}}\right) + \text{Padding}$$

7) For example, Power consumption for 16.8 is taken to calculate the same where input is 16.8, MinVal is 4.7 Resolution is 0.104 and padding is 3. After calculating using the above formula and then subtracting with the halfwidth we get the bit position which is 116.
8) Hence, Power consumption for 16.8 will be represented with 744 bits where 7 are active bits starting at 116th bit position which can be seen below.



Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.
After setting all the parameter values run the program, the output images will be captured and saved in a folder.
Bitmap method is executed in the code to produce these data in 2D Bitmap format.
In the power consumption test case, the inputs separated is more than the radius. Hence the output is non-overlapping as we can see below in Fig.13 (Output of Power Consumption) which is the snapshot of output of Power Consumption.

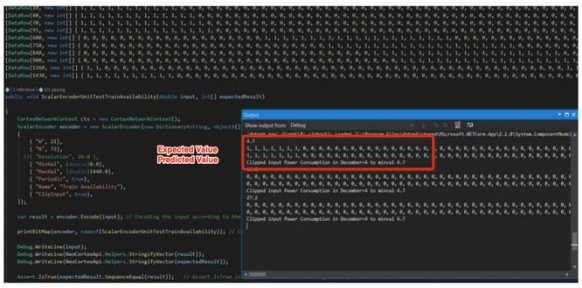


Fig.13. Output of Power consumption in code

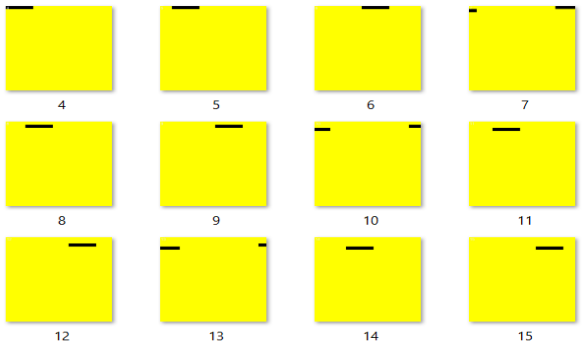


Fig.14. Output of Power consumption

6. Test case to encode player scoring goal

Encoding goals scored by different players by the help of differentiating them by jersey number. Considering players have jersey numbers 0-10. We choose to split the range into 11 Buckets since we have to differentiate each jersey number.

1) This MinVal is 0 and the MaxVal 10
2) Computing the Range = MaxVal – MinVal which is equal to 10.
3) Hence the number of bits that are set to encode a single value the ‘width’ of output signal ‘W’ used for representation is 11.
4) Total number of bits in the output ‘N’ used for representation is 21.
5) We are choosing the value of N=21 and W=11 to get the desired output which is shown below:

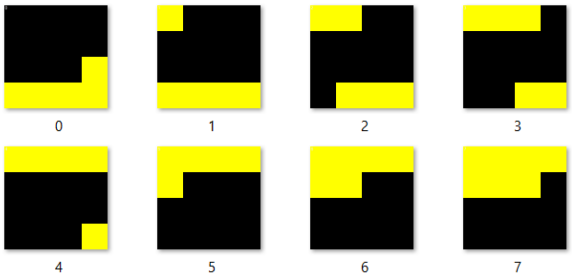


Fig.15. Expected Output of Player scoring goal

6) If we choose any other values for N and W for example N=18 and W=7 then it does not match the expected output which is shown below:

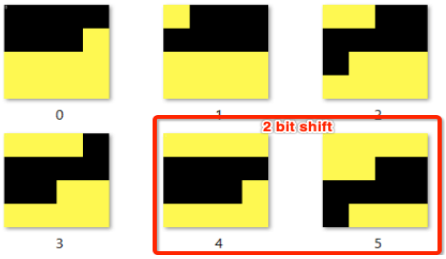
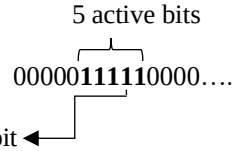


Fig.16. Output of Player scoring goal when value of N and W is altered

7) As power consumption is non- periodic, resolution is calculated which is 0.104.
8) In the output to know from which bit the ones start, below formula will be used to calculate which is:

$$\left\lfloor \frac{[(\text{Input}-\text{MinVal}) + (\text{Resolution}/2)]}{\text{Resolution}} \right\rfloor + \text{Padding}$$

9) For example, Goal scored by jersey number 5 is taken to calculate the same where input is 5, MinVal is 0 Resolution is 1 and padding is 2. After calculating using the above formula and then subtracting with the halfwidth we get the bit position which is 6.
10) Hence, Goal scored by jersey number 5 will be represented with 21 bits where 11 are active bits starting at 6th bit position which can be seen below.



Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.

After setting all the parameter values run the program, the output images will be captured and saved in a folder.
Bitmap method is executed in the code to produce these data in 2D Bitmap format.

In this scoring goal test case, there is a 1-bit shift in every goal which is in between 0 to 10, hence most of the bit overlap in adjacent values as we can see in the below figure Fig.17(Output of Player Scoring Goals) which is the snapshot of output of encoded Scoring goals test case.

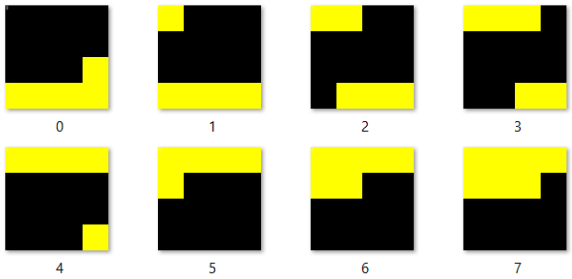


Fig.17. Output of Player Scoring Goal

7. Test Case to encode Hearing Frequency

We can imitate how the cochlea encodes frequency in the simplest possible way. The cochlea has hair cells which respond to different but overlapping ranges of frequencies.

The Data what we have is that, human ear is exposed to different range of noise intensity levels. The range 0-39.9db is No risk; Range 40.0-119.9db is Danger; Range 120.0db and above is harmful.

1) Hearing frequency is in between 0 to 160db, hence the MinVal = 0 and MaxVal = 160
2) Computing the Range = MaxVal – MinVal which is equal to 160.
3) Hence the number of bits that are set to encode a single value the ‘width’ of output signal ‘W’ used for representation is 7.
4) Total number of bits in the output ‘N’ used for representation is 9.
5) We are choosing the value of N=9 and W=7 to get the desired output which is shown below:

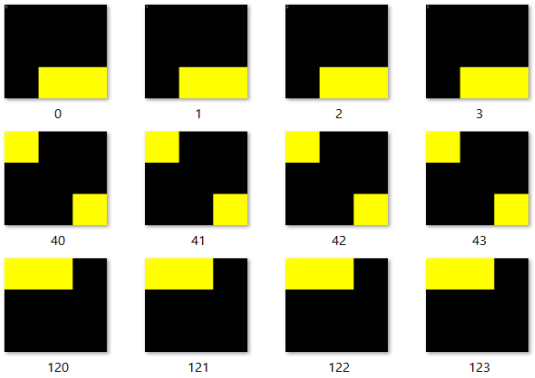


Fig.18. Expected Output of Hearing Frequency

6) If we choose any other values for N and W for example N=8 and W=5 then it does not match the expected output which is shown below:

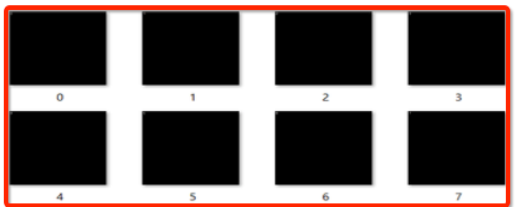


Fig.19. Output of Hearing Frequency when value of N and W is altered

7) We have 3 ranges i.e. (0-39.9db) has one encoding value, 40.0-119.9db has another encoding value, 120.0db and above has different encoding value.

Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.
After setting all the parameter values run the program, the output images will be captured and saved in a folder.
Bitmap method is executed in the code to produce these data in 2D Bitmap format.

In this hearing frequency test case, there is a shift within those categories mentioned above and we can see the same in the below figure Fig.20(Output of Hearing Frequency) which is the snapshot of output of encoded Hearing Frequency test case.

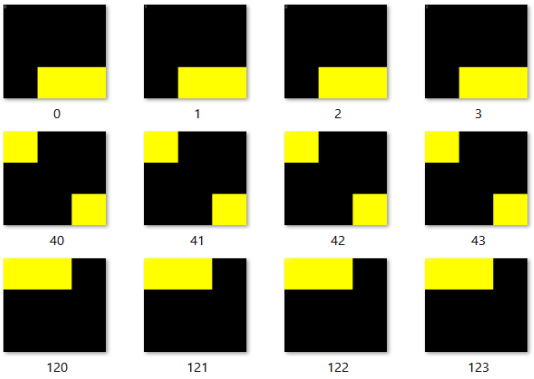


Fig.20. Output of Hearing Frequency

8. Test case to encode Train Availability in a station.

This test case will enable us to encode the availability of train for an entire day. Assuming that the trains will arrive every 20 mins. Firstly the 24 hours clock will be converted into minutes which will be equal to 1440 minutes a day.

1) Since it is a day clock in minutes it starts from 0 and end at 1440, hence the MinVal = 0 and MaxVal = 1440
2) Computing the Range = MaxVal – MinVal which is equal to 1440.
3) Choosing the value of 'W' and 'N' in such way that there should be a shift for every 20 minutes in the output.
4) Total number of bits in the output 'N' used for representation is 72.
5) The number of bits that are set to encode a single value the 'width' of output signal 'W' used for representation is 21 is.
6) We are choosing the value of N=72 and W=21 to get the desired output.
7) If we choose any other values for N and W for example N=8 and W=5 then it does not match the expected output.
8) The time interval between adjacent trains can be changed by altering the values of N and W for the known MinVal and MaxVal.

Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.
After setting all the parameter values run the program, the output images will be captured and saved in a folder.
Bitmap method is executed in the code to produce these data in 2D Bitmap format.

In this availability of train test case, there is a shift after every 20 minutes which is in between 0 to 1440, As it is periodic most of the bit overlap in adjacent values as we can see in the below figure Fig.21(Output of Train Availability) which is the snapshot of output of encoded availability of train test case.

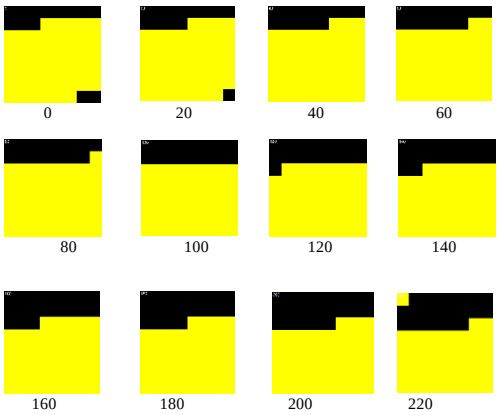


Fig.21. Output of Train Availability

9. Test Case to encode Age of Passengers in U-Bahn

Encoding the different category of people traveling in U-Bahn according to their age. Let us say we have children's, teenagers, adults, and senior citizens traveling in U-Bahn. We must differentiate travelers based on this category choosing the bracket of age as difference of 9.9 years.

Assuming the people entering have ages in the range of 1 year to 100 years. We would like to encode differently example 0-6 years as one category and other category such as 6-15.9, 16-25.9, 26-35.9, 36-45.9, 85+ years. So, we are encoding different category age of people in different way.

1) Age of passenger is in between 1 to 100 years, Hence the MinVal = 1 and MaxVal = 100
2) Computing the Range = MaxVal – MinVal which is equal to 99.
3) Hence the number of bits that are set to encode a single value the 'width' of output signal 'W' used for representation is 21.
4) Total number of bits in the output 'N' used for representation is 31.
5) We are choosing the value of N=31 and W=21 to get the desired output which is shown below:

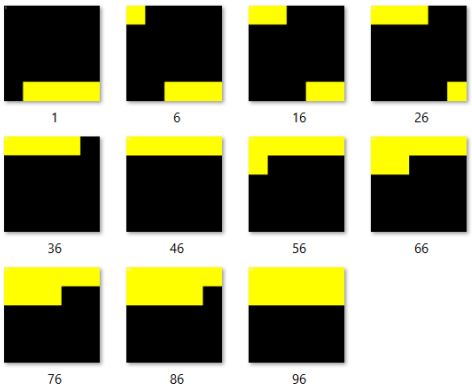


Fig.22. Expected Output of Age of Passengers in U-Bahn

6) If we choose any other values for N and W for example N=35 and W=19 then it does not match the expected output which is shown below:

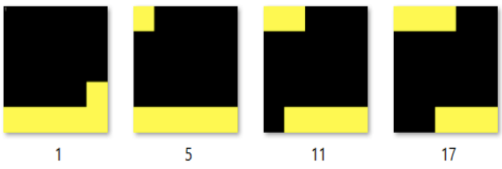


Fig.23. Output of Age of Passengers in U-Bahn when value of N and W is altered

Once all the inputs are encoded, we call the Bitmap method to show the output in 2D Bitmap format.
After setting all the parameter values run the program, the output images will be captured and saved in a folder.

Bitmap method is executed in the code to produce these data in 2D Bitmap format.

In this Age of passenger in U-Bahn test case, there is a shift within those categories mentioned above and we can see the same in the below figure Fig.9(Output of Age of passenger in U-Bahn) which is the snapshot of output of encoded Age of passenger in U-Bahn test case .

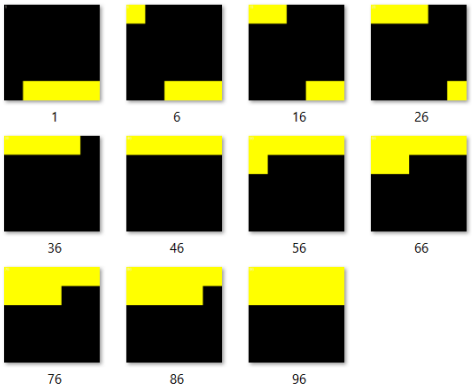


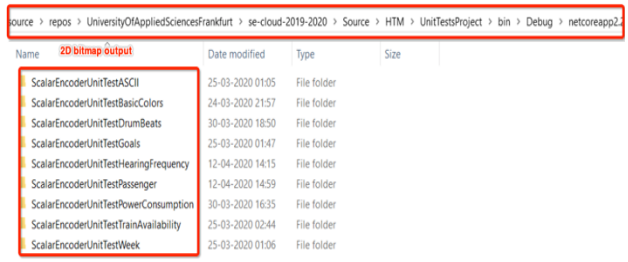
Fig.24. Output of Age of Passengers in U-Bahn

All the test cases generate 2D bitmap output. These bitmap outputs are generated by following code:

```
string filename;
Directory.CreateDirectory(folderName);
for (double i = (long)encoder.MinVal; i < (long)encoder.MaxVal; i += 1)
{
    var result1 = encoder.Encode(i);
    int[,] twoDimenArray = ArrayUtils.MakeIDArray<int>(result1, (int)Math.Sqrt(result1.Length), (int)Math.Sqrt(result1.Length));
    var twoDimArray = ArrayUtils.Transpose(twoDimenArray);
    filename = i + ".png";
    NeoCortexUtils.DrawBitmap(twoDimArray, 1024, 1024, Path.Combine(folderName, filename), Color.Yellow, Color.Black, text: i.ToString());
}
```

IV. LINK TO ACCESS OUR TEST CASES

The Below link can be used to access this test cases and it will be used to generate the figure and see output for all values in your local folder.
https://github.com/UniversityOfAppliedSciencesFrankfurt/se-cloud-2019-2020/blob/Group3/Source/HTM/UnitTestsProject/EncoderTests/ScalarEncoderTests.cs
In the below
In our case all the output that is generated on local machine which can be seen as follows:



V. CONCLUSION

The needs of most of the application are covered by the number of encoders available. For a new data type, if a new encoder needs to be built, there are few rules which can be followed to build the encoder, which are as follows:

1) The data which are semantically alike should result in SDRs where active bits are overlapping.
2) If the inputs are same, then outputs should always have same SDR
3) The dimensionality of the output should be same for all the inputs.
4) Similar sparsity should be there in output for all the inputs and to handle noise and subsampling it should have enough one-bits.

VI. REFERENCES

1) Purdy, Scott. (2016). Encoding Data for HTM systems https://www.researchgate.net/publication/301844094_Encoding_Data_for_HTM_Systems

2) Hawkins, J. & Ahmad, S. (2015). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. arXiv, 1511.00083. Neurons and Cognition; Artificial Intelligence. Retrieved from http://arxiv.org/abs/1511.00083

3) Ahmad, S., & Hawkins, J. (2016). How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. arXiv, 1601.00720. Neurons and Cognition; Artificial Intelligence. Retrieved from http://arxiv.org/abs/1601.00720

4) Webber, F. D. S. (2015). Semantic Folding Theory And its Application in Semantic Fingerprinting, 57. Artificial Intelligence; Computation and Language; Neurons and Cognition. Retrieved from http://arxiv.org/abs/1511.08855

5) Abu Zneit, Rushdi & Alqadi, Ziad & Mohammad, Abu & Zalatas, (2017). A Methodology to Create a Fingerprint for RGB Color Image. IJCSMC, Vol. 6, Issue. 1, January 2017, pg.197 – 204 Procedural Analysis of Procedural Analysis of Procedural Analysis of Procedural Analysis of Procedural Analysis of Procedural Analysis of Procedural Analysis of RGB Color Image O. 61. 205-212. https://www.researchgate.net/publication/313199892_A_Methodology_to_Create_a_Fingerprint_for_RGB_Color_Image

6) Vogl, Richard & Widmer, Gerhard & Knees, Peter. (2018). Towards multi-instrument drum transcription. https://www.researchgate.net/publication/325841542_Towards_multi-instrument_drum_transcription
7) Power Consumption Excel Sheet https://github.com/UniversityOfAppliedSciencesFrankfurt/se-cloud-2019-2020/tree/Group3/Source/HTM/UnitTestsProject/EncoderTests/Documentation