Image Classification using a Brain Learning Algorithm

- **Presented by :**

  Pranika Kakkar - 1710110253
  Rishika Chaudhary - 1710110276

- **Under the Supervision of :**

  Dr. Sonal Singhal

SPIKING NEURAL NETWORK

- SNN's are ANN's that mimic natural neural networks.
- The neurons in SNN do not fire at every propagation cycle, they incorporate the sense of timing.
- After neuron firing, a signal is generated which travels to other neurons and the potential is increased or decreased.
- SNN is a feed forward network.

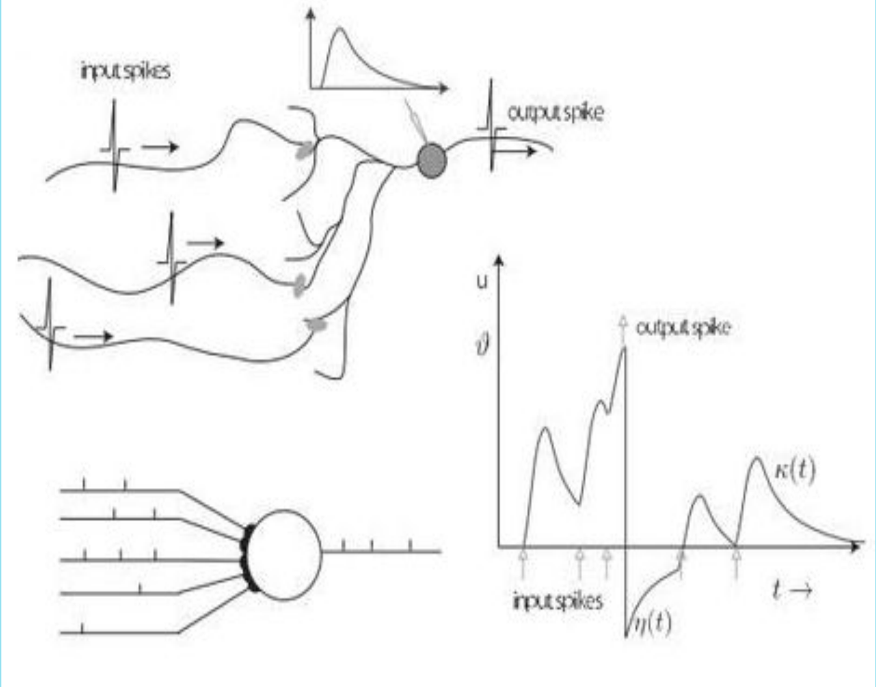


Fig. Spiking Neural Network mechanism

STDP ARCHITECTURE

$$w_{\text{new}} = \begin{cases} w_{\text{old}} + \sigma \Delta w (w_{\max} - w_{\text{old}}), & \text{if } \Delta w > 0 \\ w_{\text{old}} + \sigma \Delta w (w_{\text{old}} - w_{\min}), & \text{if } \Delta w \leq 0 \end{cases}$$

Hebbian learning [4] -

- When two neurons fire at almost the same time the connections between them are strengthened and thus they become more likely to fire again in the future.

- When two neurons fire in an uncoordinated manner the connections between them weaken and they are more likely to act independently in the future.

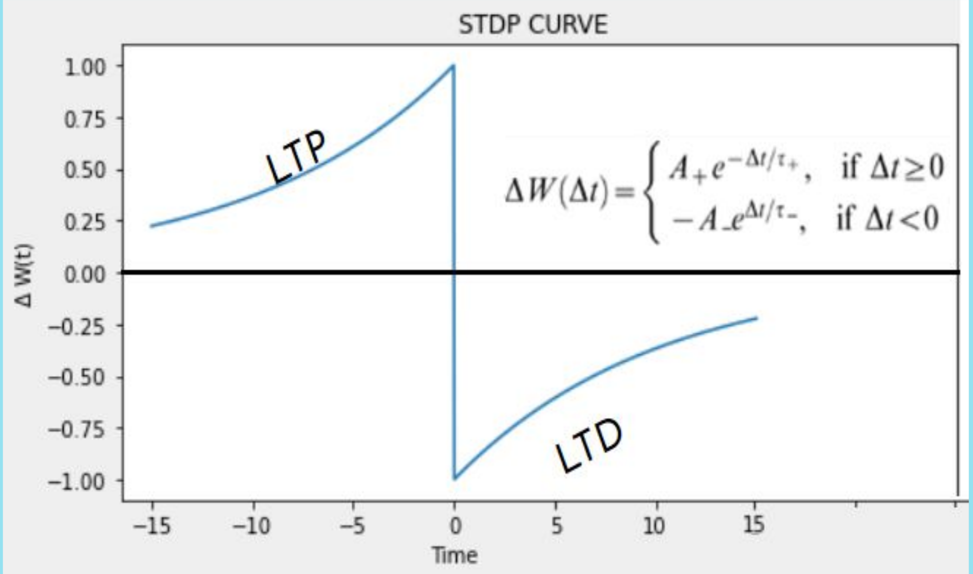


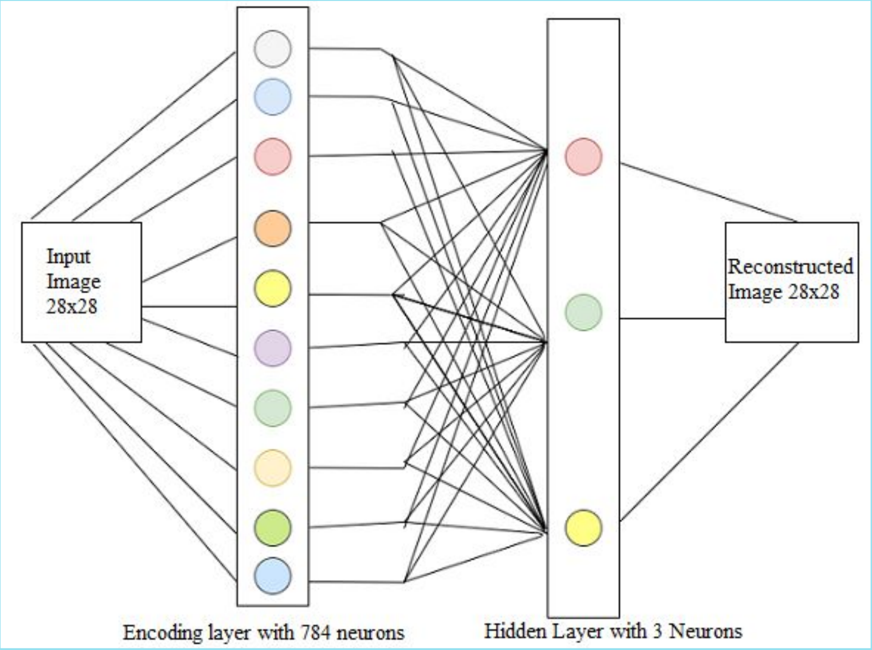
Fig. STDP Graph with exponential function

MOTIVATION & OBJECTIVE

- Due to their power efficiency, SNN are a potential candidate for human brain motivated neuromorphic evaluation.
- The human brain has robust unsupervised learning ability that is it can function without data samples that have already been labelled.
- SNN are able to process spatial and temporal patterns and are more computationally powerful than ANN [1].

- To implement the Spiking Neural Network architecture for image classification.
- To perform STDP as our brain learning algorithm, which will enable us to update the synapses efficiently.
- To be able to execute the reconstruction of the input image.

METHODOLOGY



- The python implementation of spiking neural network is done based on this architecture.
- The training and testing images are taken from the MNIST dataset, containing hand-written digits.
- Input is a 28x28 image and output is the reconstructed image of the same shape, implemented using STDP.

OVERVIEW OF THE NETWORK STRUCTURE

CODE WORKFLOW

- Image Encoding
- Synapse Update with Brain learning algorithm
- Neural Model
- Reconstruction of images

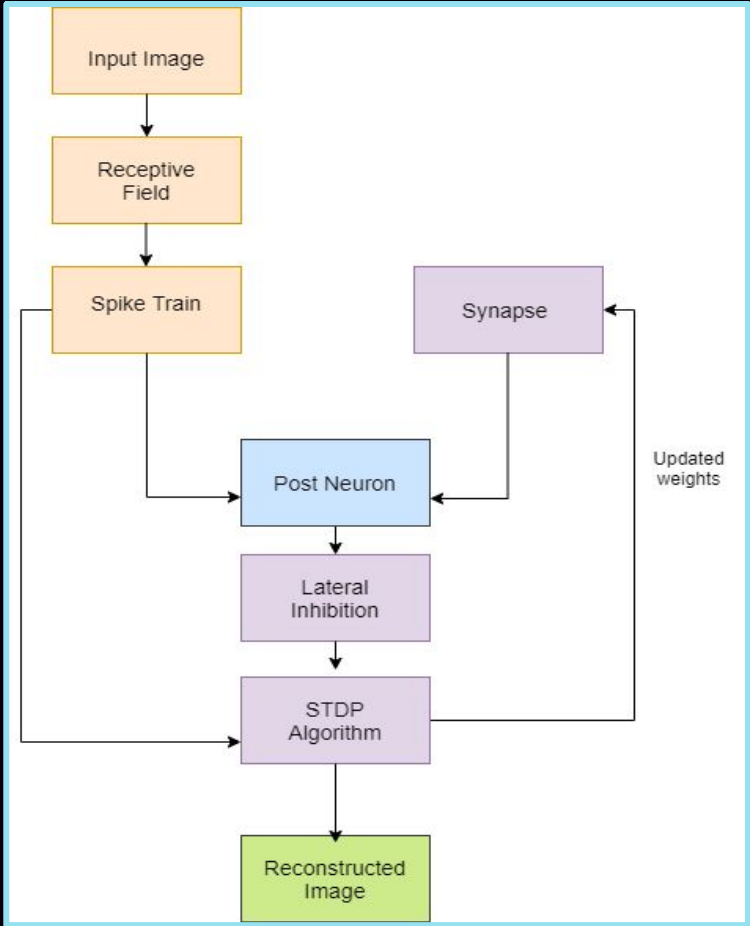


Figure - Code Workflow

DATA SET

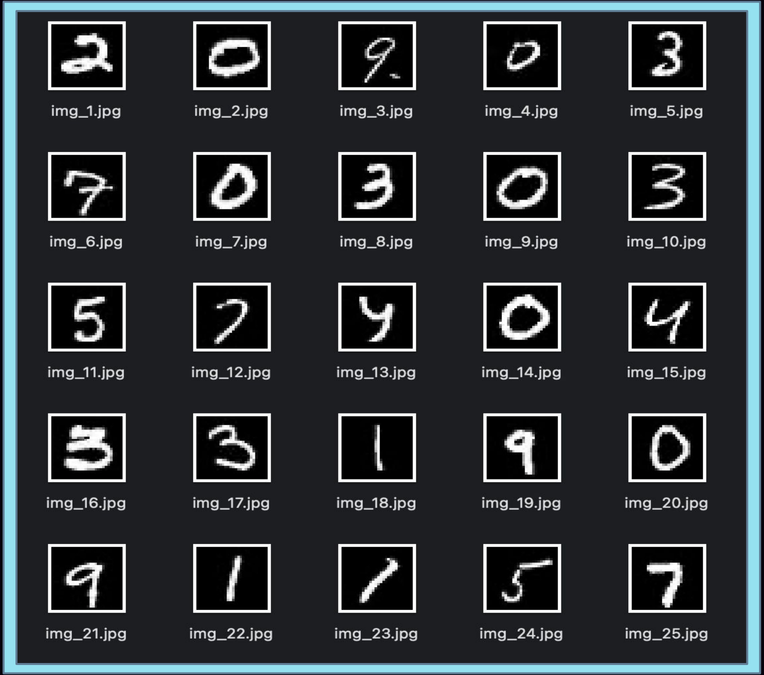


Figure - Sample Images from [3]

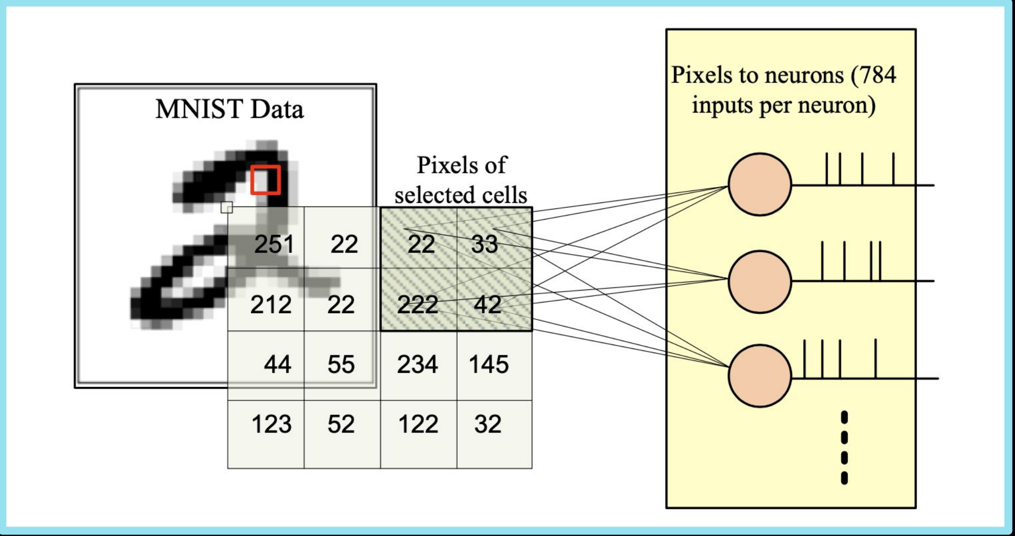
- ❖ The MNIST database of handwritten digits.[3]
- ❖ Useful for learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.
- ❖ Each image size is 28X28 Pixel
- ❖ 1000 images for training

IMAGE ENCODING

- The encoding an image involves the conversion of the pixel values to spike train.

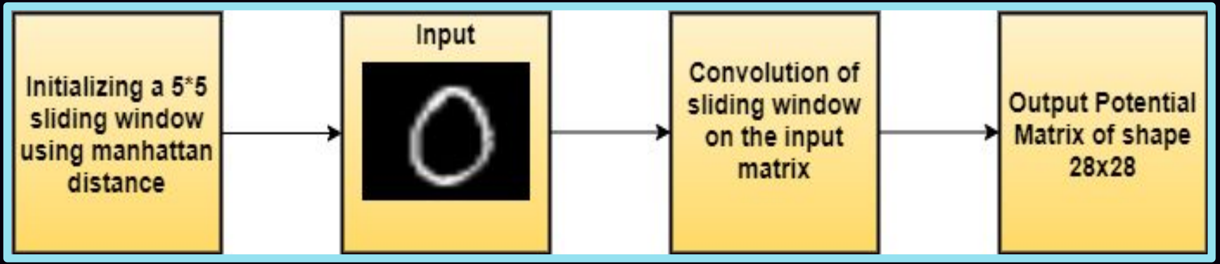
- Involves the mechanism of formation of receptive field which outputs the membrane potential matrix.

- With the help of the potential matrix, the spike train would be generated.



Neuromorphic spike encoding mechanism of input image [2]

RECEPTIVE FIELD & SPIKE TRAIN




```
[[-0.5   -0.125  0.25  -0.125 -0.5  ]
 [-0.125  0.25   0.625  0.25  -0.125]
 [ 0.25   0.625  1.     0.625  0.25 ]
 [-0.125  0.25   0.625  0.25  -0.125]
 [-0.5   -0.125  0.25  -0.125 -0.5  ]]
```

Figure - Sliding Window

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-0.023529	-0.272549	-0.558824	-0.563725	-0.473039	-0.587255	-0.589706	-0.540686	-0.437255	-0.226961	-0.005882	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-0.011765	-0.351961	-0.639706	-0.453922	-0.044608	-0.230882	-0.411275	0.040686	-0.066667	-0.642647	-0.558333	-0.030882	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-0.257843	-0.636765	-0.446569	0.902451	1.764706	1.867647	1.711275	1.964216	1.628922	0.367157	-0.54902	-0.480392	-0.078431	0	0	0	0	0	0
0	0	0	0	0	0	0	-0.105882	-0.569608	-0.555882	0.680392	2.04951	2.384314	2.160784	2.315196	2.2	2.273529	1.501471	0.102941	-0.616667	-0.227451	0	0	0	0	0	0
0	0	0	0	0	0	-0.005882	-0.290686	-0.558333	0.220098	1.655882	2.203431	1.430882	0.440196	0.014706	0.484804	1.167157	1.57549	0.868137	-0.210784	-0.554412	-0.15098	0	0	0	0	0
0	0	0	0	0	0	-0.111275	-0.561275	-0.082843	0.995098	1.982843	1.926961	1.081373	-0.712255	-1.290196	-0.373529	0.506373	1.387255	1.378431	0.404412	-0.571078	-0.290686	0	0	0	0	0
0	0	0	0	0	0	-0.376961	-0.496569	0.327451	1.404412	1.506863	1.270588	0.239216	-0.694118	-0.919118	-0.603922	0.005882	1.347549	1.978922	0.868137	-0.336275	-0.456863	0	0	0	0	0
0	0	0	0	0	-0.041176	-0.526961	-0.129902	0.920098	1.5	0.942157	0.230882	-0.601471	-0.57598	-0.182353	-0.541667	-0.476471	0.666667	2.32549	1.490196	-0.131863	-0.547549	-0.041176	0	0	0	0
0	0	0	0	0	-0.261275	-0.470588	0.276961	1.493137	1.506373	0.344118	-0.354412	-0.535784	-0.196078	-0.007843	-0.270098	-0.609314	-0.060294	1.975	2.081373	0.103922	-0.698529	-0.176961	0	0	0	0
0	0	0	0	0	-0.419118	-0.394118	0.588725	1.894608	1.014216	-0.266176	-0.585294	-0.123529	0	0	-0.045098	-0.510784	-0.586765	1.138725	2.347549	0.495588	-0.803922	-0.396078	0	0	0	0
0	0	0	0	0	-0.405882	-0.354902	1.190196	1.994118	0.30098	-0.737745	-0.389706	-0.007843	0	0	0	-0.403431	-1.034804	0.623039	2.457353	1.052941	-0.733333	-0.263725	0	0	0	0
0	0	0	0	0	-0.470588	-0.276961	1.771569	1.895098	-0.161765	-0.816667	-0.115686	0	0	0	0	-0.186275	-0.944608	0.228922	2.609804	1.692647	-0.45	-0.403431	0	0	0	0
0	0	0	0	0	-0.709314	-0.17549	2.079412	1.731863	-0.434314	-0.662745	0	0	0	0	0	0	-0.927941	-0.04902	2.39951	1.972059	-0.606863	-0.591176	0	0	0	0
0	0	0	0	0	-0.825	-0.088725	2.137255	1.67402	-0.521078	-0.547059	0	0	0	0	0	-0.019608	-1.011765	-0.064706	2.39902	1.840686	-0.541667	-0.47549	0	0	0	0
0	0	0	0	0	-0.825	-0.14951	2.122059	1.689216	-0.536275	-0.607843	0	0	0	0	0	-0.222549	-0.981863	0.317157	2.336275	1.441176	-0.506373	-0.239706	0	0	0	0
0	0	0	0	0	-0.711275	-0.279412	2.059314	1.82598	-0.439706	-0.769608	-0.013725	0	0	0	-0.023529	-0.410294	-0.57451	0.596078	2.018627	0.823529	-0.491667	-0.415196	0	0	0	0
0	0	0	0	0	-0.429902	-0.276961	1.647549	2.072059	0.041176	-0.872059	-0.333333	-0.017647	0	-0.047059	-0.427451	-0.331863	-0.010784	1.036765	1.456863	0.460294	-0.49902	-0.313235	0	0	0	0
0	0	0	0	0	-0.395098	-0.278431	1.108333	1.936765	0.683333	-0.52402	-0.662745	-0.363235	-0.27549	-0.52402	-0.336275	0.114216	0.298039	0.983333	1.146078	0.137255	-0.511765	-0.076471	0	0	0	0
0	0	0	0	0	-0.429902	-0.397549	0.854412	1.789216	1.056373	0.214216	-0.684804	-1.208333	-1.097059	-0.642157	-0.003922	0.437255	0.52402	1.173529	0.636765	-0.321078	-0.431373	0	0	0	0	0
0	0	0	0	0	-0.252941	-0.556863	0.387745	1.768627	1.764216	1.031863	0.636275	0.12402	0.226471	0.664706	1.156373	1.080882	1.187745	0.728922	-0.054412	-0.438725	-0.14902	0	0	0	0	0
0	0	0	0	0	0	-0.526961	-0.448039	0.727941	1.758824	1.918137	2.318627	2.817647	2.917157	2.65049	1.930882	1.360294	0.623529	-0.095588	-0.50049	-0.213235	0	0	0	0	0	0
0	0	0	0	0	0	-0.272549	-0.604902	-0.433333	0.381863	1.103431	1.603922	2.166176	2.312745	1.597059	1.02451	0.338235	-0.383333	-0.52598	-0.181863	-0.021569	0	0	0	0	0	0
0	0	0	0	0	0	0	-0.186275	-0.553922	-0.633333	-0.321569	-0.304412	-0.417647	-0.401471	-0.262745	-0.485294	-0.626961	-0.540196	-0.168627	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-0.092157	-0.366176	-0.558824	-0.516176	-0.459804	-0.504902	-0.556373	-0.508333	-0.311765	-0.078431	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure - Output Potential Matrix

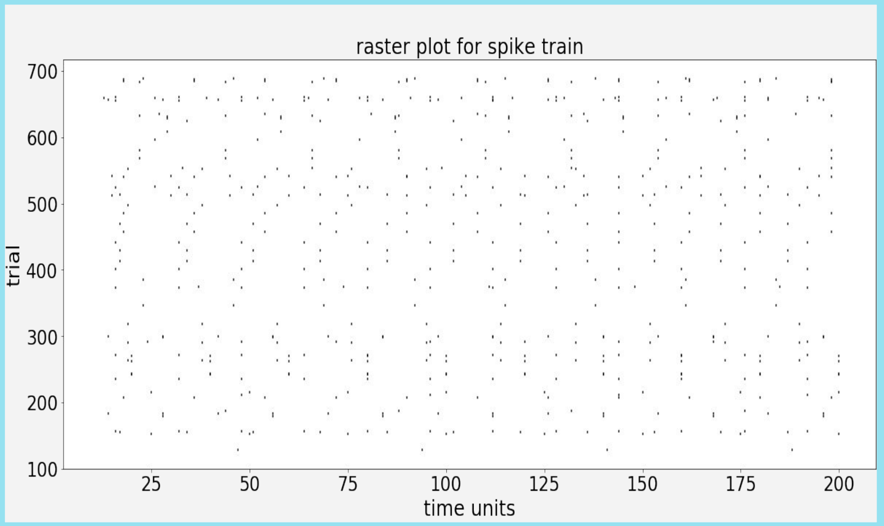


Figure - Spike Train Raster Plot

NEURON MODEL

- Neurons - Basic building block of SNN.
- As long as Pt> Pmin, there is a constant leakage of potential.
- For an n-input SNN, Pt - Membrane Potential , Wi - Value of Synapse weight, Rp - Rest Potential [1]

$$P_t = \begin{cases} P_{t-1} + \sum_{i=1}^{n} W_i S_{it} - D, & \text{if } P_{\min} < P_{t-1} < P_{\text{threshold}} \\ P_{\text{refract}}, & \text{if } P_{t-1} \geq P_{\text{threshold}} \\ R_p, & \text{if } P_{t-1} \leq P_{\min} \end{cases}$$

Equation- To explain Neuron model process[1]

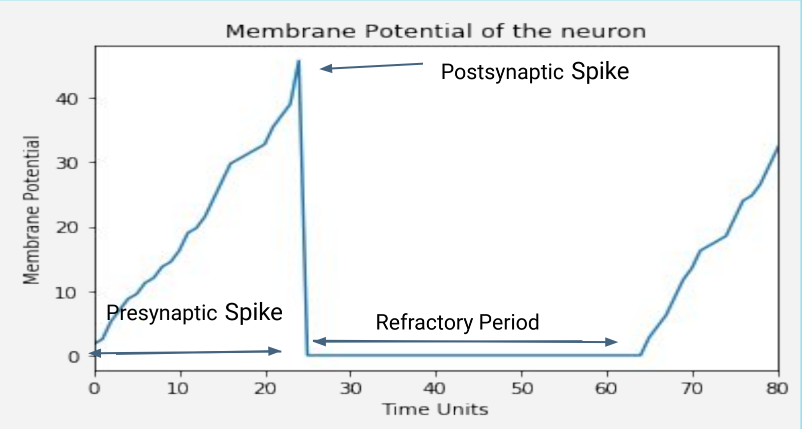


Figure - Membrane Potential Output

SYNAPSE AND LEARNING ALGORITHM

- Each neuron in the first layer is connected to all the neurons in the hidden layer by a weighted path, synapse.
- Synapses are updated later using STDP algorithm.
- To optimise our network and make it more robust **Lateral Inhibition** is introduced.
- the weights of the first spiking neurons, , or the winner neurons, are increased and the other non spiking neurons experience a reduction in their weight value.

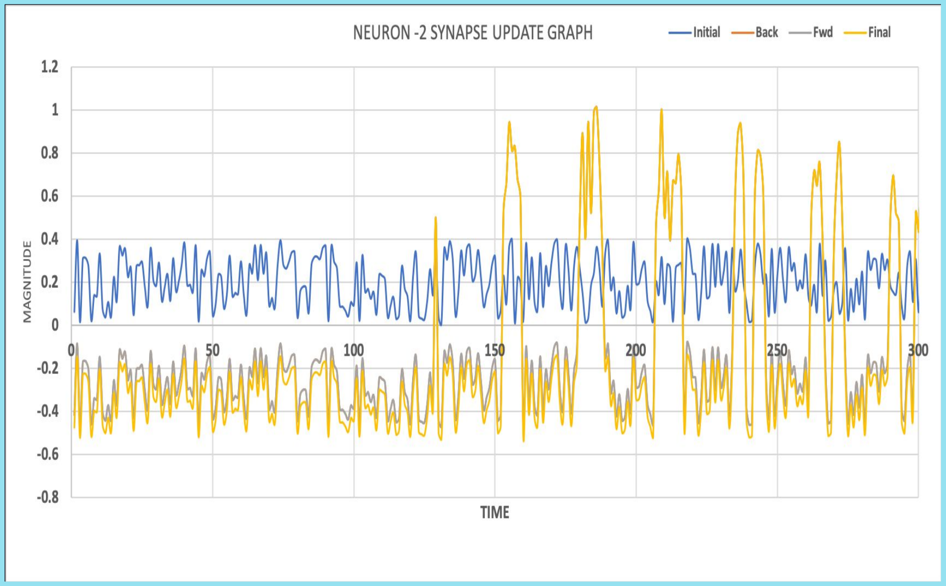


Fig. Winner Neuron with updated synapse

IMAGE RECONSTRUCTION

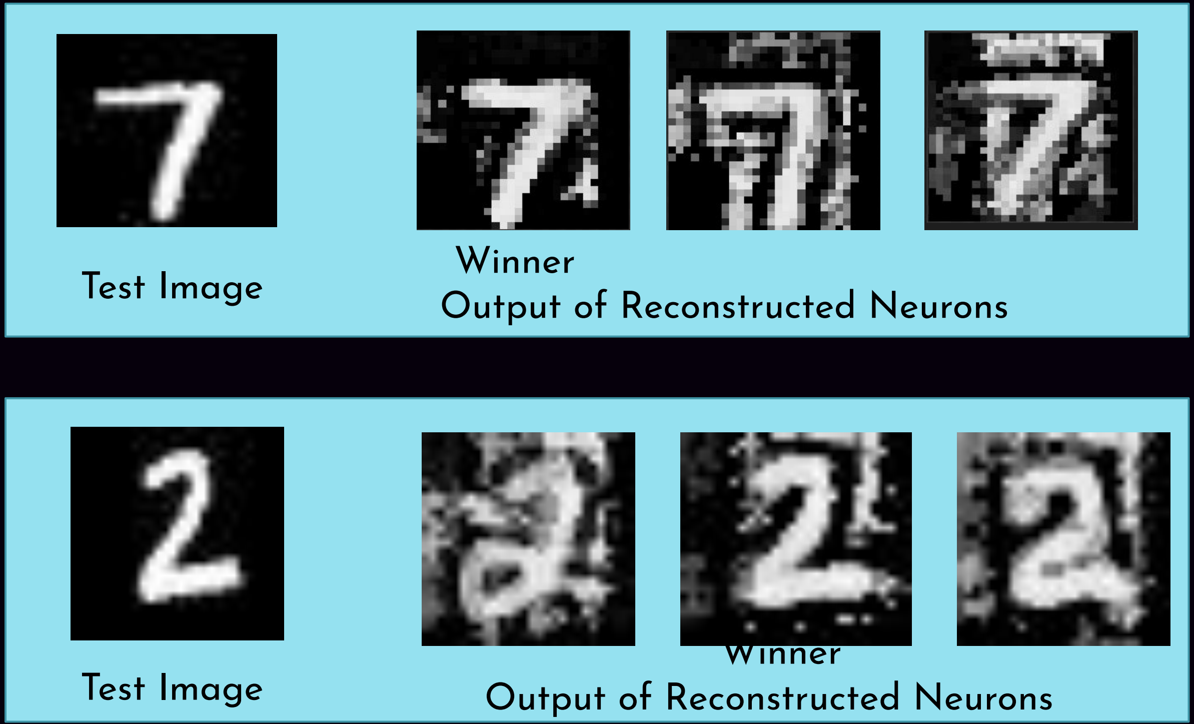


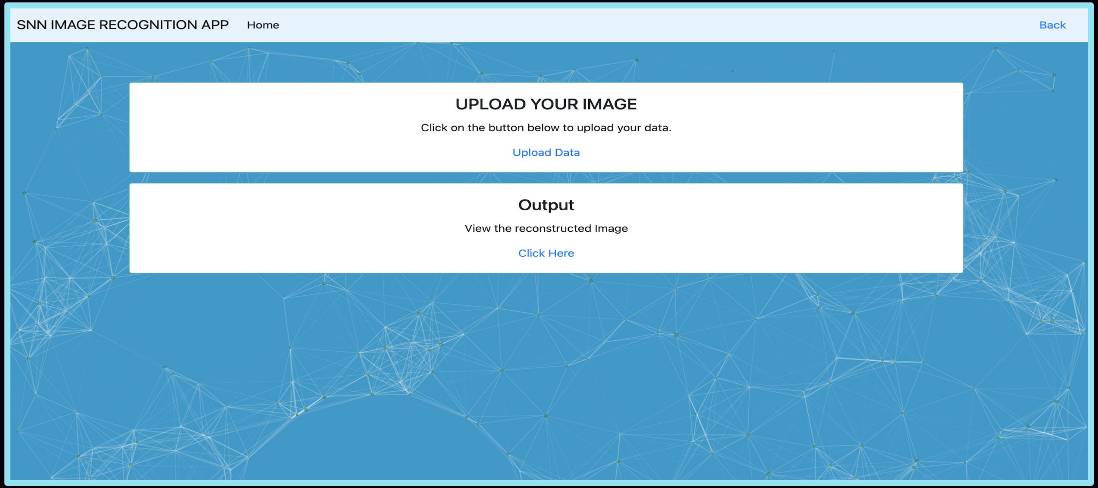
Figure - Reconstructed Images using Updated Synapse Weights

REFERENCES

	Authors	Year	Title
1	Taras Iakymchuk , Alfredo Rosado-Muñoza and Juan F Guerrero-Martínez	2015	Simplified spiking neural network architecture and STDP learning algorithm applied to image classification [1]
2	VS. Aadithiya, Jani Babu Shaik, Sonal Singhal, S. M. Picardo, and N. Goel.	2020	Design and mathematical modelling of inter spike interval of temporal neuromorphic encoder for image recognition
3	Dataset		https://archive.ics.uci.edu/ml/support/Pen-Based+Recognition+of+Handwritten+Digits
4	Yi-Ling Hwong	2017	Unsupervised Learning with Spike-Timing Dependent Plasticity
5	Diehl, Peter and Cook, Matthew	2015	Unsupervised learning of digit recognition using spike-timing-dependent plasticity

CONCLUSIONS AND FUTURE WORK

- Creation and implementation of essential modules - image encoding, synapse and learning algorithm and Neuron Model.
- Training and testing of the model using the MNIST image dataset and reconstruction of the image using updated weights.
- Application of Lateral Inhibition- Winner Takes All approach for optimization.
- Optimization of the model to reduce complexities and facilitate the use of other RGB datasets.
- The model can find its use in facial recognition, useful for an attendance management system.



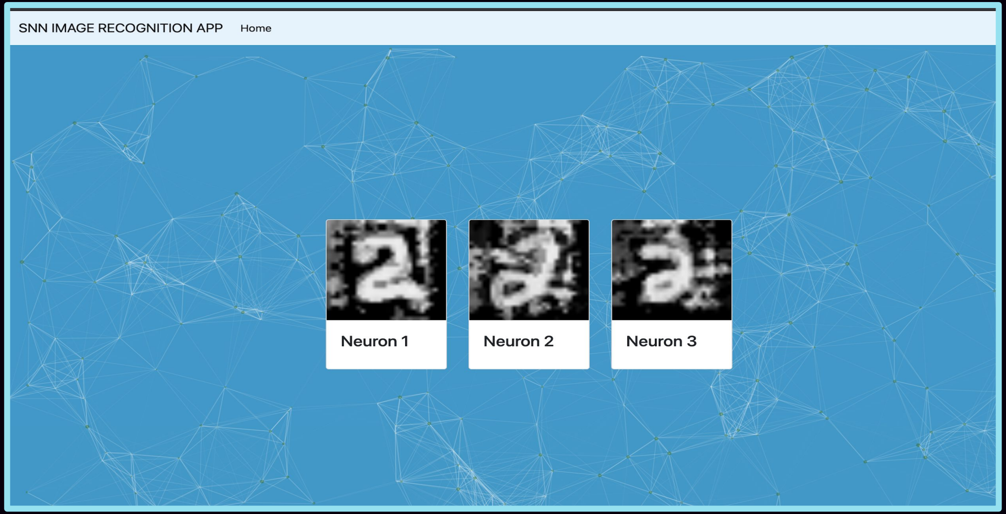


Figure - GUI of the App and the reconstructed output.

THANK YOU