EnsembleSTDP: Distributed in-situ Spike Timing Dependent Plasticity Learning in Spiking Neural Networks

Author: Yuga Hanyu* and Khanh N. Dang †

School of Computer Science and Engineering
University of Aizu, Fukushima, Japan
E-mail: {*s1290224, † khanh}@u-aizu.ac.jp

Date: December 18, 2024

Outline

- Introduction
- Proposed Method
- Evaluation & Result
- Conclusion

Outline

- **Introduction**
- Proposed Method
- Evaluation & Result
- Conclusion

Spiking Neural Network [2]

- Spiking Neural Networks (SNNs) use spikes to communicate information
- Compared to traditional ANNs, SNNs are energy efficient

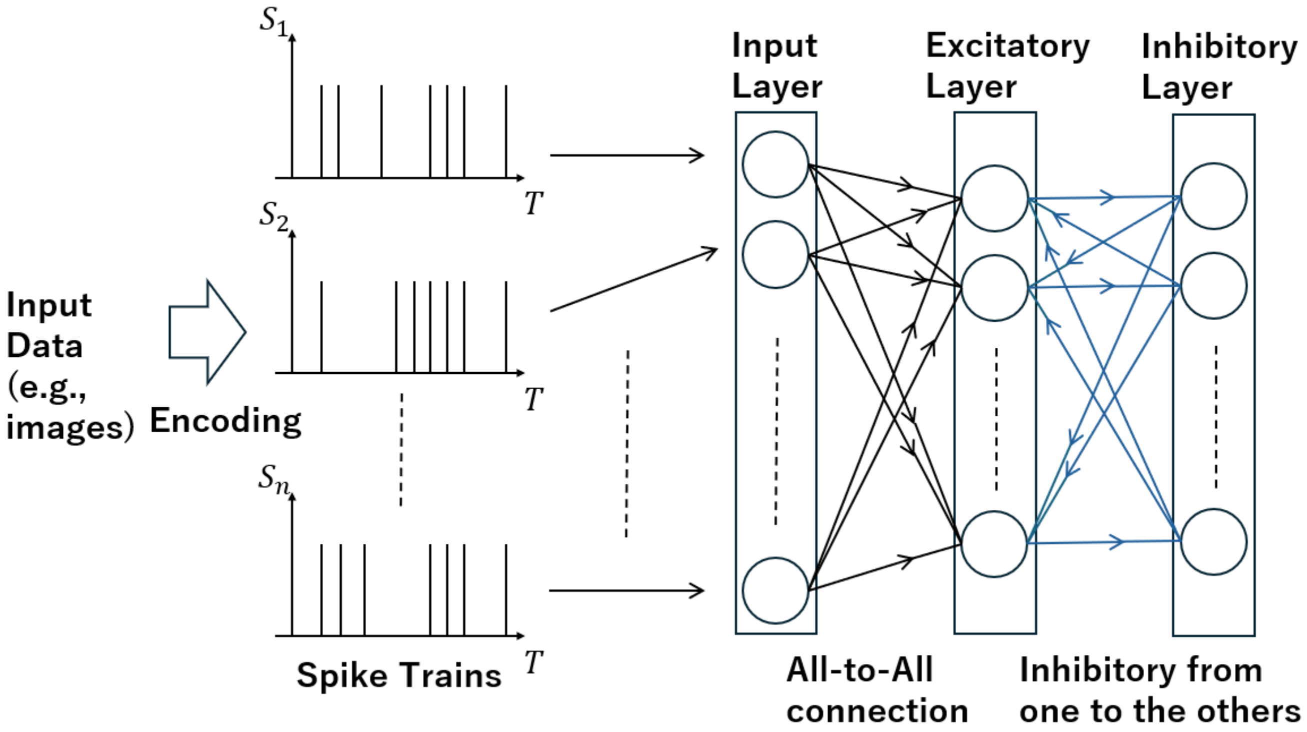


Fig. 1 Structure of an SNN

Spike Timing Dependent Plasticity [3]

STDP is a learning method based on spike timings

- lightweight computation
- great energy efficiency
- suitable for on-chip learning

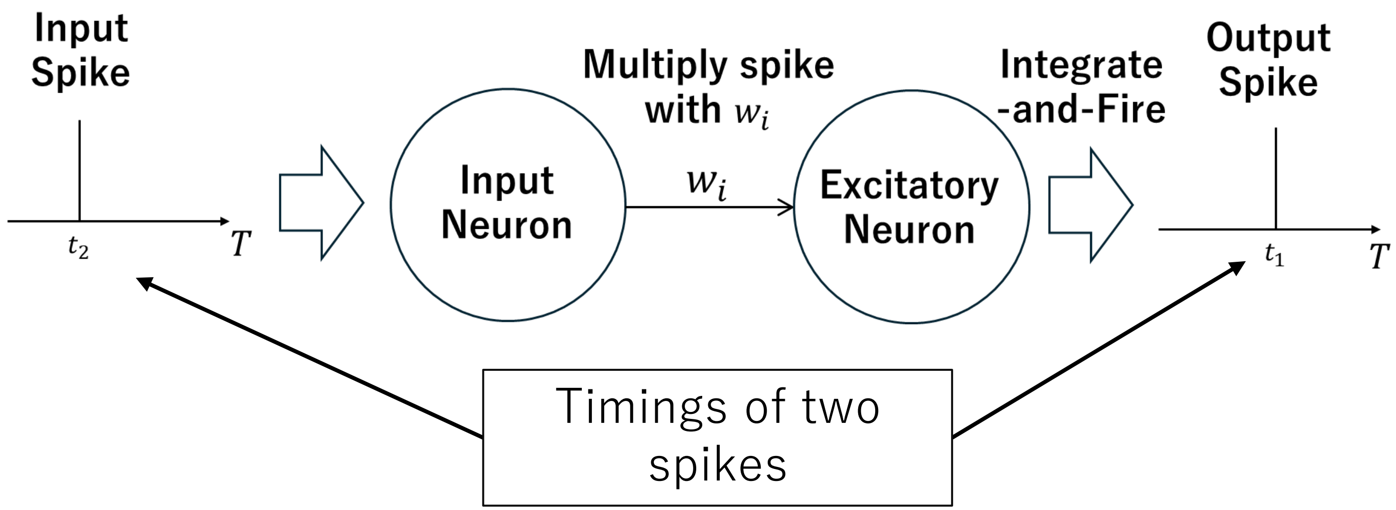


Fig. 2 Input and output spikes

Ensemble Learning

Split a dataset, train sub-models, and merge into one model

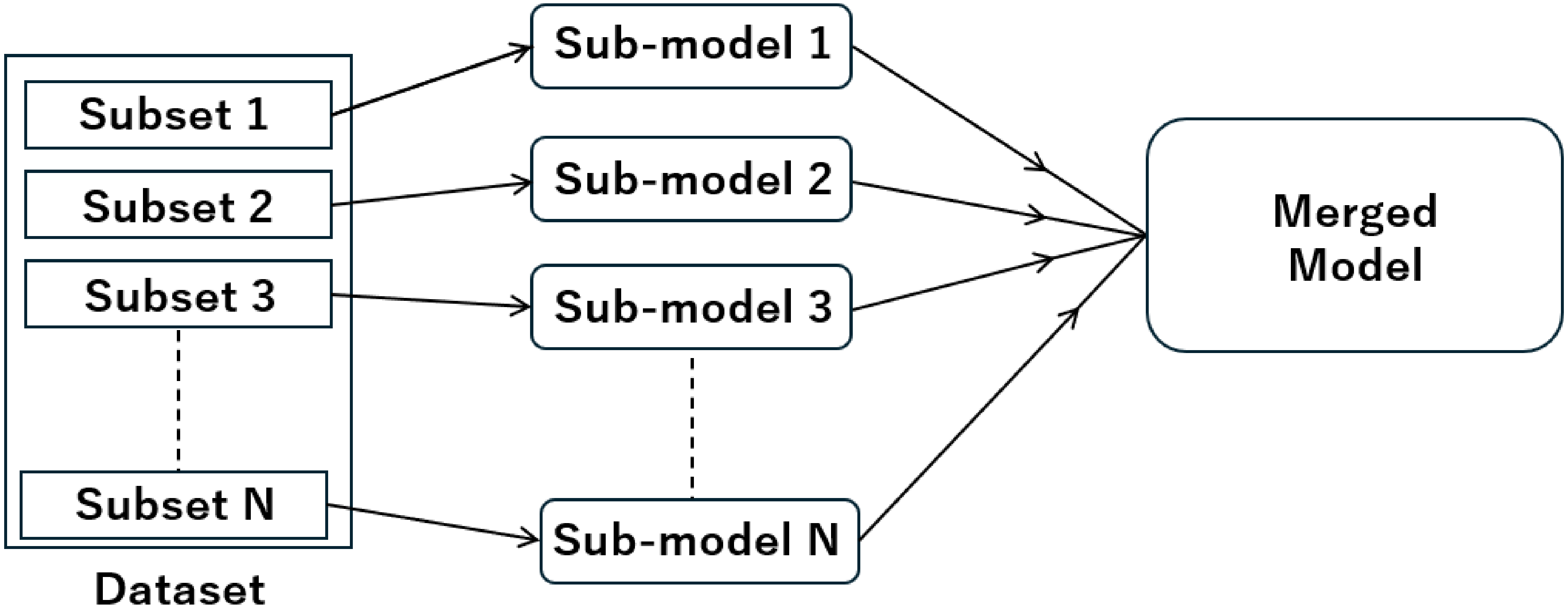


Fig. 3 Overview of Ensemble Learning

Motivations

Several challenges in lightweight on-chip learning with STDP

- **Distributed Datasets**: Training SNN sub-models on different local datasets leads to the varying performance
- **Model Redundancy:** Merging SNN sub-models with similar neuron characteristics leads to overall redundancy
- **No pure STDP learning:** Existing methods are not fully STDP-based, limiting efficient on-chip learning

Our contributions

- **EnsembleSTDP:** Efficient Ensemble Learning with STDP that allows characteristic differences of the learners

- **Efficient Model Compression:** Removing redundancy from the concatenated model

Outline

- Introduction
- **Proposed Method**
  - **❶ STDP Learning**
  - **❷ Concatenation**
  - **❸ Similarity generation**
  - **❹ Neuron removal**
- Evaluation & Result
- Conclusion

Proposed Distributed In-situ Learning Model

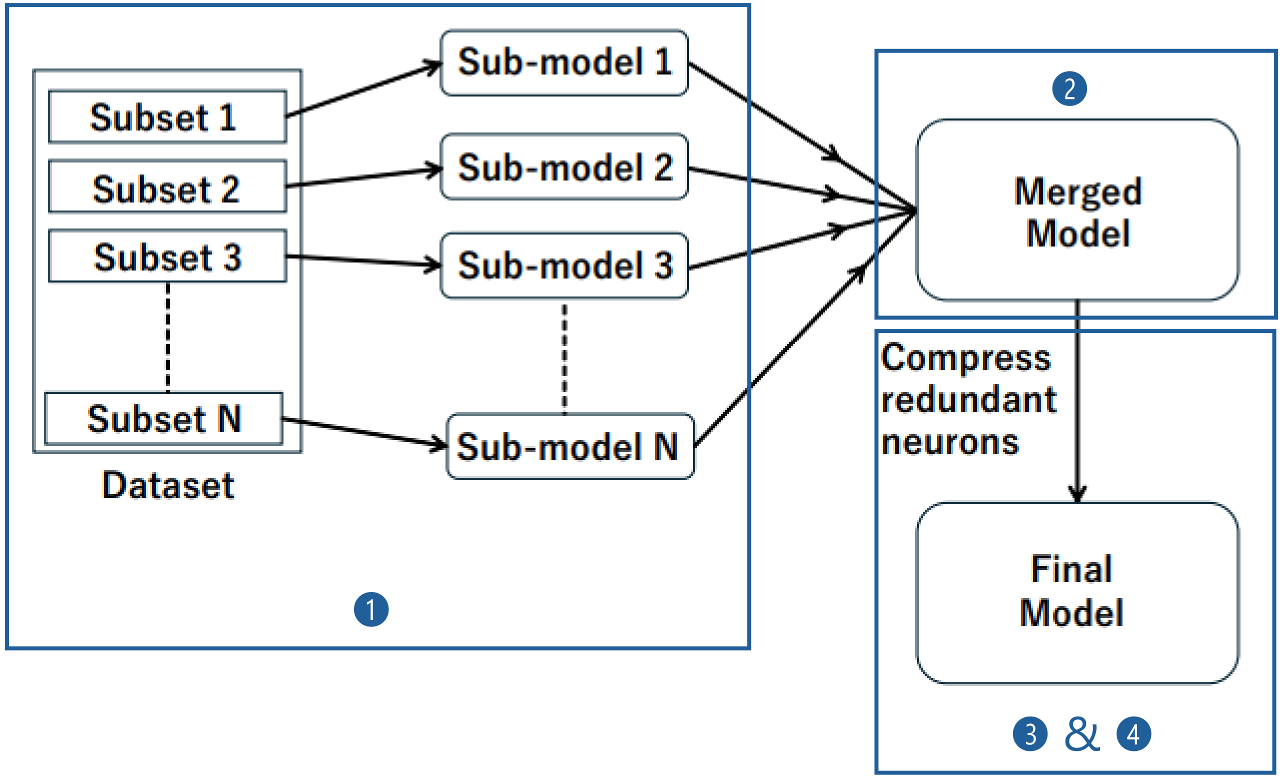


Fig. 4 Steps of Distributed Learning Model

Key steps:

1. In-situ STDP learning
2. Concatenation of the trained SNN models
3. Generate pairs of similar neurons
4. Remove an arbitrary number of neurons from the pairs

❶ In-situ STDP learning

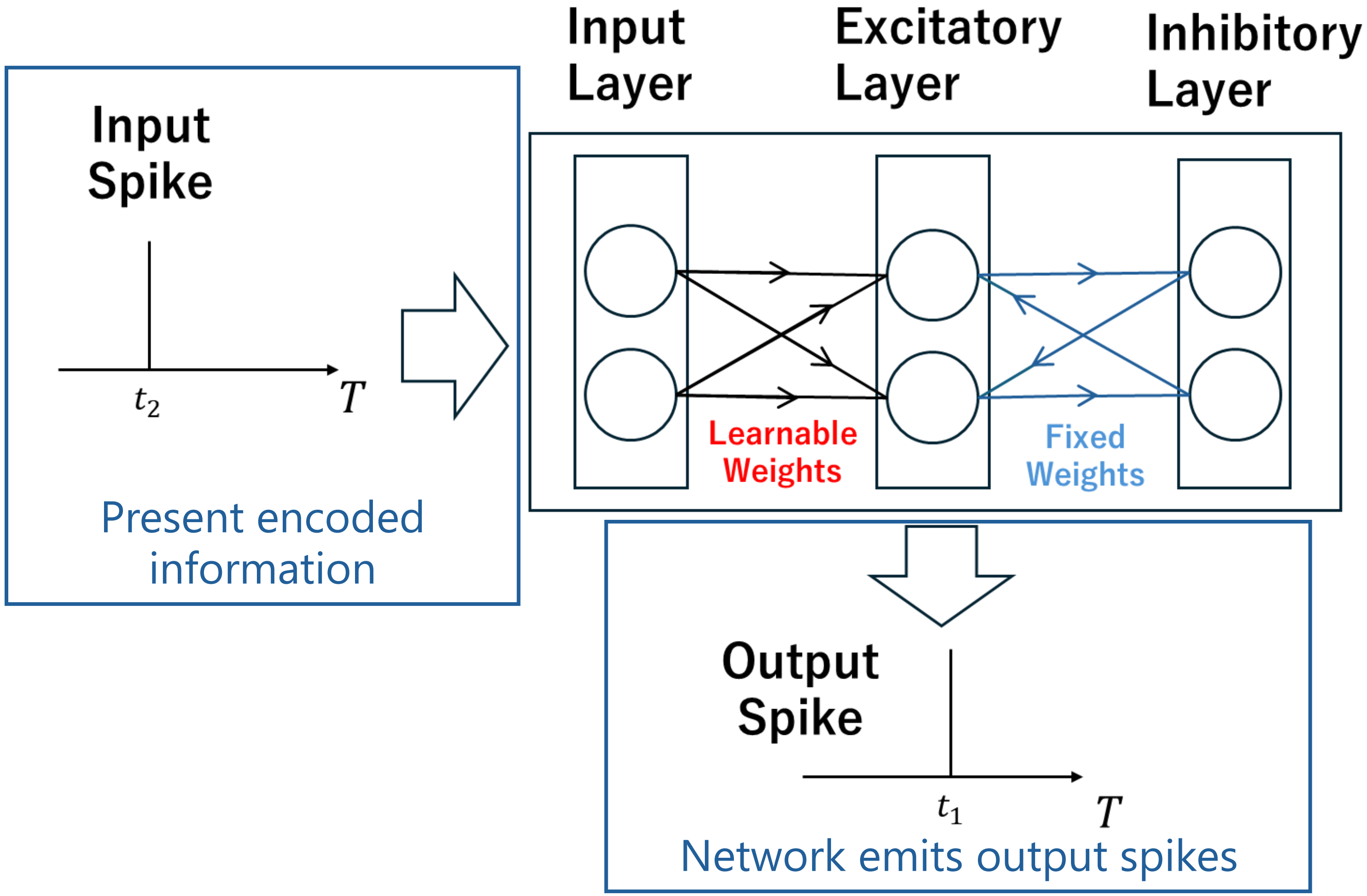


Fig. 5 Flow of the spikes

❶ In-situ STDP learning (cnt.)

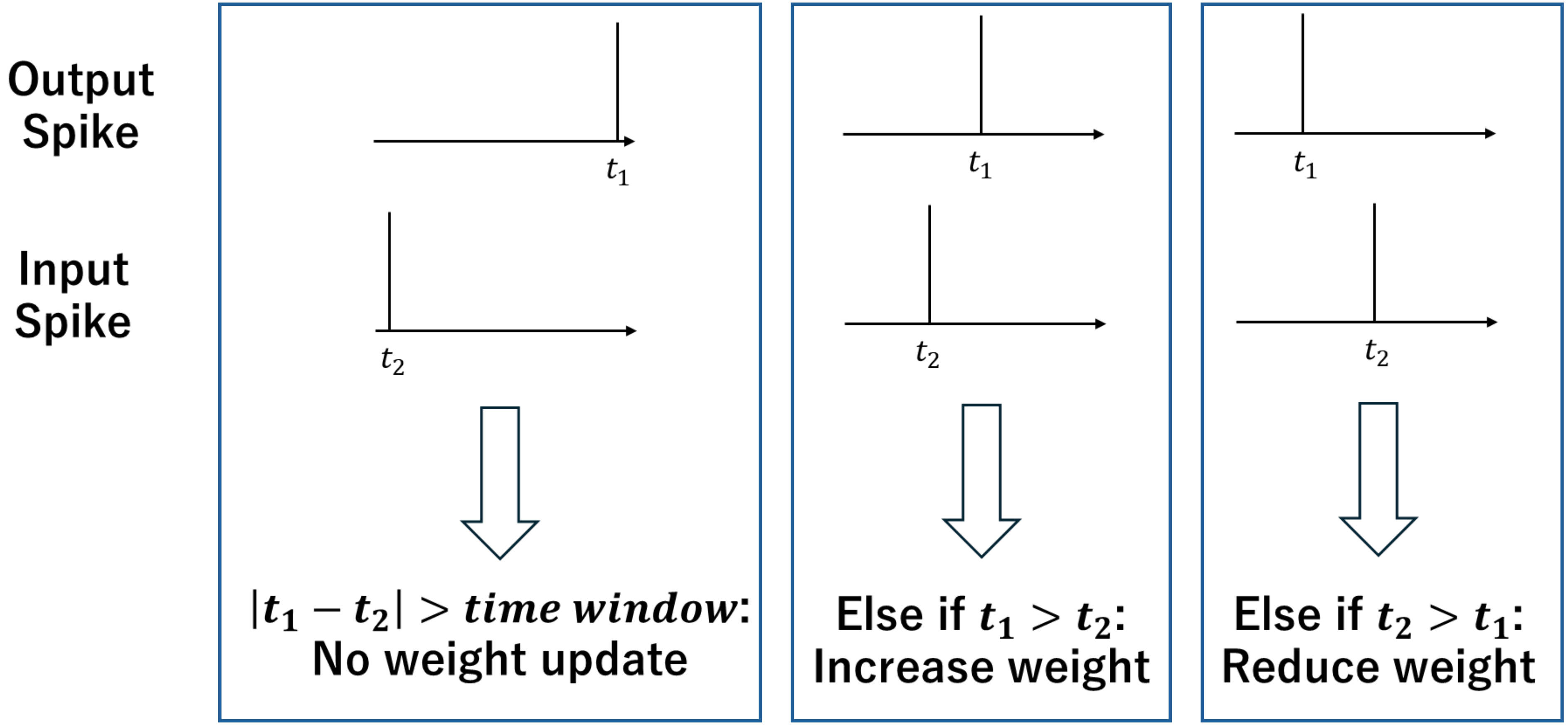


Fig. 6 Three cases of weight updating

❷ Concatenation of trained SNNs

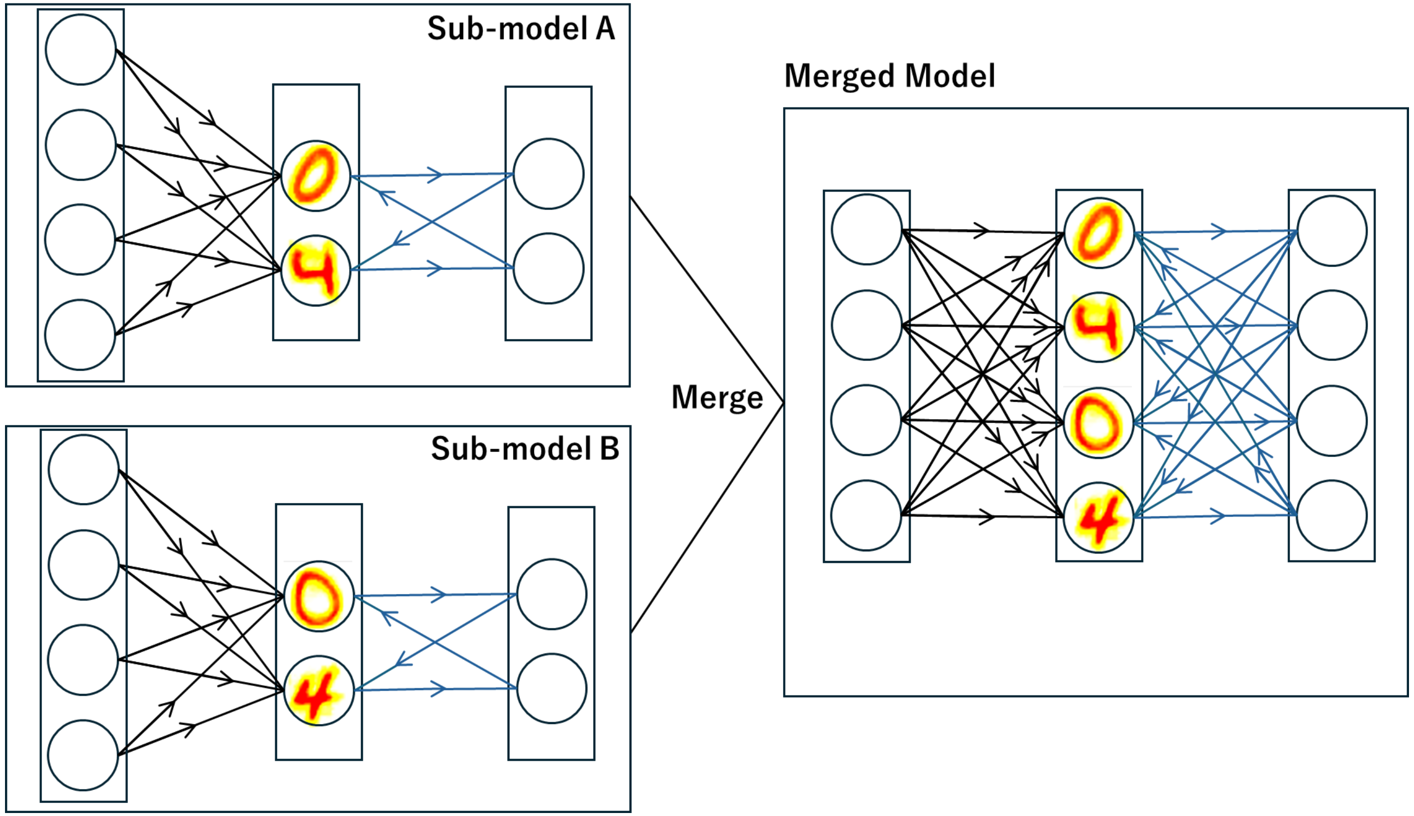


Fig. 7 Concatenation of sub-models

- Concatenation: Gathering the learned weights
- Better accuracy from concatenated model

Model	sub-model #1	sub-model #2	Merged Model
No. of neurons	64	64	128
Accuracy	75.38%	75.54%	77.60%

Table 1 Result of concatenating 2 sub-models

❸ Generate pairs of similar neurons

Algorithm 1 Generate pairs with similarity values.

Input: n
Output: pairs
pairs $\leftarrow \{\}$
for $i \leftarrow 0$ to $n-1$ **do**
    for $j \leftarrow i+1$ to n **do**
        similarity $\leftarrow$ `calculate_similarity`(i, j)
        pairs.append$((i, j, \text{similarity}))$
    end for
end for

Neuron pairs with similarity values

Calculate & Assign similarity value

n: number of neurons in the concatenated model

③ Generate pairs of similar neurons (cnt.)

Similarity is measured by comparing weight values of two neurons

- **Mean Squared Error**
- **Manhattan Distance**
- **Cosine Similarity**
- **Correlation Coefficient**

Weight 1

0.1	0.2
0.3	0.4

Weight 2

0.2	0.3
0.1	0.4

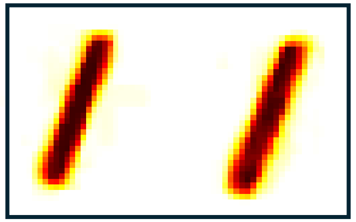
Similarity value:

$$|0.1 - 0.2| + |0.2 - 0.3| + |0.3 - 0.1| + |0.4 - 0.4| = 0.4$$

Example calculation (Manhattan Dist.)

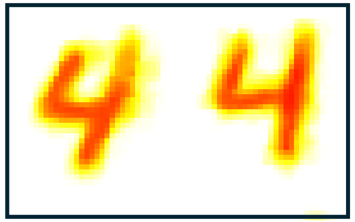
❹ Remove neurons from the pairs

❹-❶ Sort the pairs in order of high similarity



Similarity: 0.9

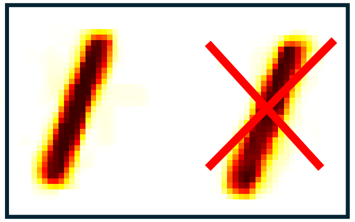
Similarity: 0.8



Similarity: 0.7

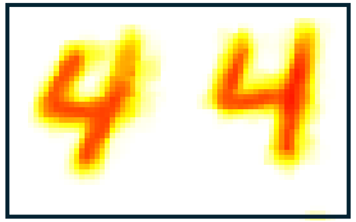
④ Remove neurons from the pairs (cnt.)

④-② Remove a neuron from higher similarity pairs (repeat for a number of times)



Similarity: 0.9

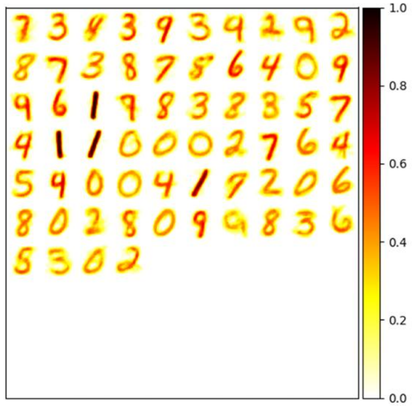
Similarity: 0.8



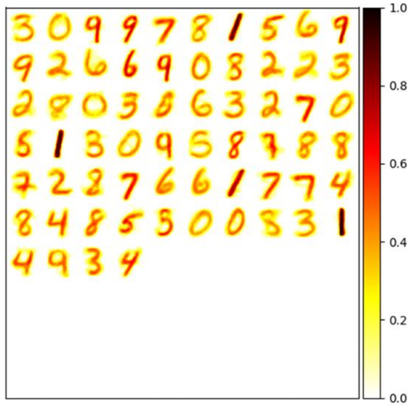
Similarity: 0.7

End Result

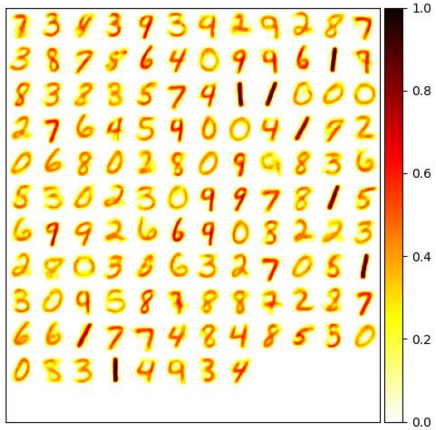
- Concatenating & Compressing two models



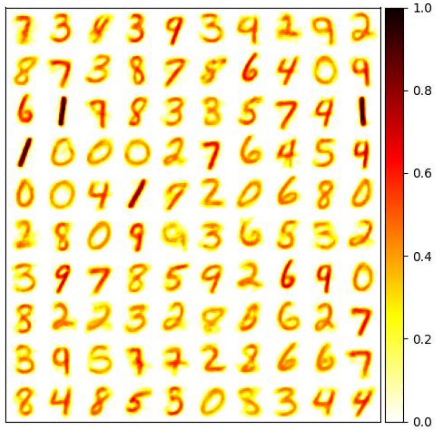
64 neurons
75.38%



64 neurons
75.54%



128 neurons
77.60%



100 neurons
77.08%

Outline

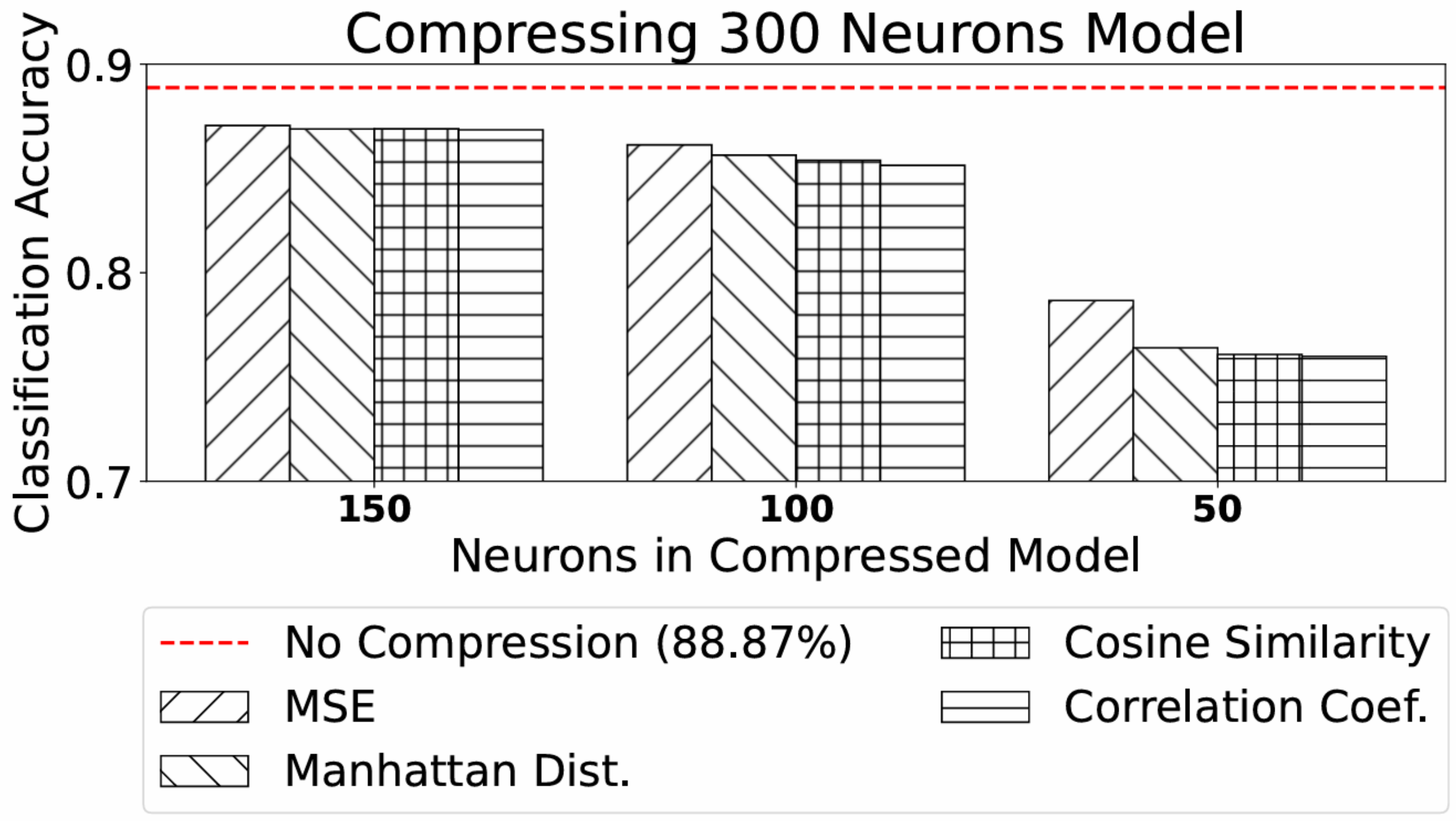
- Introduction
- Proposed Method
- **Evaluation & Result**
- Conclusion

Evaluation & Result

- Setup:
  - Machine: core i7-13700K, Nvidia RTX4070
- Model:
  - Learning Model: pre-and-post-synaptic STDP
  - Programming and Library: Python, BindsNet [2]
  - 5 sub-models, 100 neurons each, merged into a 300 model
  - 2 sub-models, 250 neurons each, merged into a 300 model
- Criterion:
  - Classification Accuracy
  - Processing Speed
  - Trade-off between compression/accuracy/speed

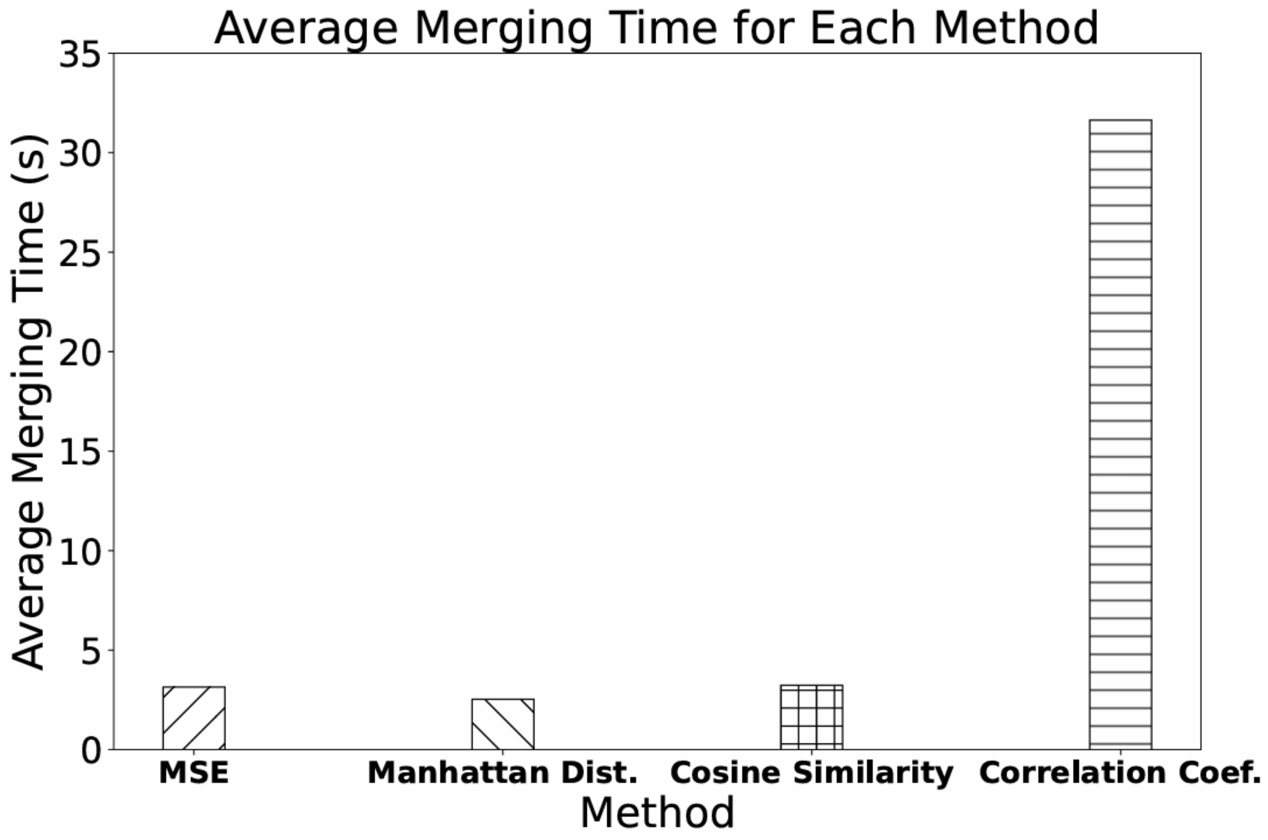
Compression vs Accuracy

Train a 300-neuron model on 30K images and compress



- No major differences until 50 neurons
- MSE performed best
- Performance retention

Execution time



- Correlation Coef. takes significantly longer time
- Correlation Coef. needs multiple data iterations
- > 99% of merging time is generating & sorting pairs

Single 300-neuron model vs Our (5x100-200)

Train 5 sub-models (100 neurons/model), merge and compress to 300 neurons.

Model	Baseline [3]	Our
#neurons	300	300 (5x100-200)
Training Time (minutes)	53.13	10.58
Classification Accuracy	88.87%	85.42%

5.02x speed up in training with trade-off of 3.45% accuracy.

Single 300-neuron model vs Our (2x250-200)

Train 2 sub-models (250 neurons/model), merge and compress to 300 neurons.

Model	**Baseline [3]**	**Our**
#neurons	300	300 (2x250-200)
Training Time (minutes)	53.13	26.8
Classification Accuracy	88.87%	89.28%

1.98x speed up in training and 0.41% accuracy gain.

Outline

- Introduction
- Proposed Method
- Evaluation & Result
- **Conclusion**

Conclusion

- Fully STDP based Ensemble Learning is implemented.
- The system allows different individual model performances & removes redundancy based on neuron similarity.
- Proposed method achieved 1.98x training process acceleration while gaining 0.41% of classification accuracy.
- Future work: Communication between sub-models during training & Genetic Algorithm to further optimize the final model.

Reference

[1] Z.-H. Zhou, Ensemble methods: foundations and algorithms. CRC press, 2012.

[2] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," Frontiers in computational neuroscience, vol. 9, p. 99, 2015.

[3] Q. Fu and H. Dong, "An ensemble unsupervised spiking neural network for objective recognition," Neurocomputing, vol. 419, pp. 47–58, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220313072

Thank you for your attention!

Proposed Merging Method

Similarity Measurements

Mean Squared Error -> Average squared difference of weight metrices

Manhattan Dist. -> Sum of absolute differences of weight metrices

Cosine Similarity -> Cosine angle between weight metrices

Correlation Coef. -> How much two weight matrices move with each other

Similarity Measurements (1)

Mean Squared Error

$\frac{1}{n}\sum_{i=1}^{n}(A_i - B_i)^2$

n: number of weights

A_i: i_th weight in weight matrix A

B_i: i_th weight in weight matrix B

Sensitive to larger differences

Similarity Measurements (2)

Manhattan Distance

$$\sum_{i=1}^{n} |A_i - B_i|$$

Simple, less sensitive to larger differences

Similarity Measurements (3)

Cosine Similarity (Range: -1 to 1)

$$\frac{A \cdot B}{|A| \cdot |B|}$$

$|A|$: magnitude of weight matrix A

$A \cdot B$: dot product of A and B

$|A| \cdot |B|$: multiplication between the magnitudes

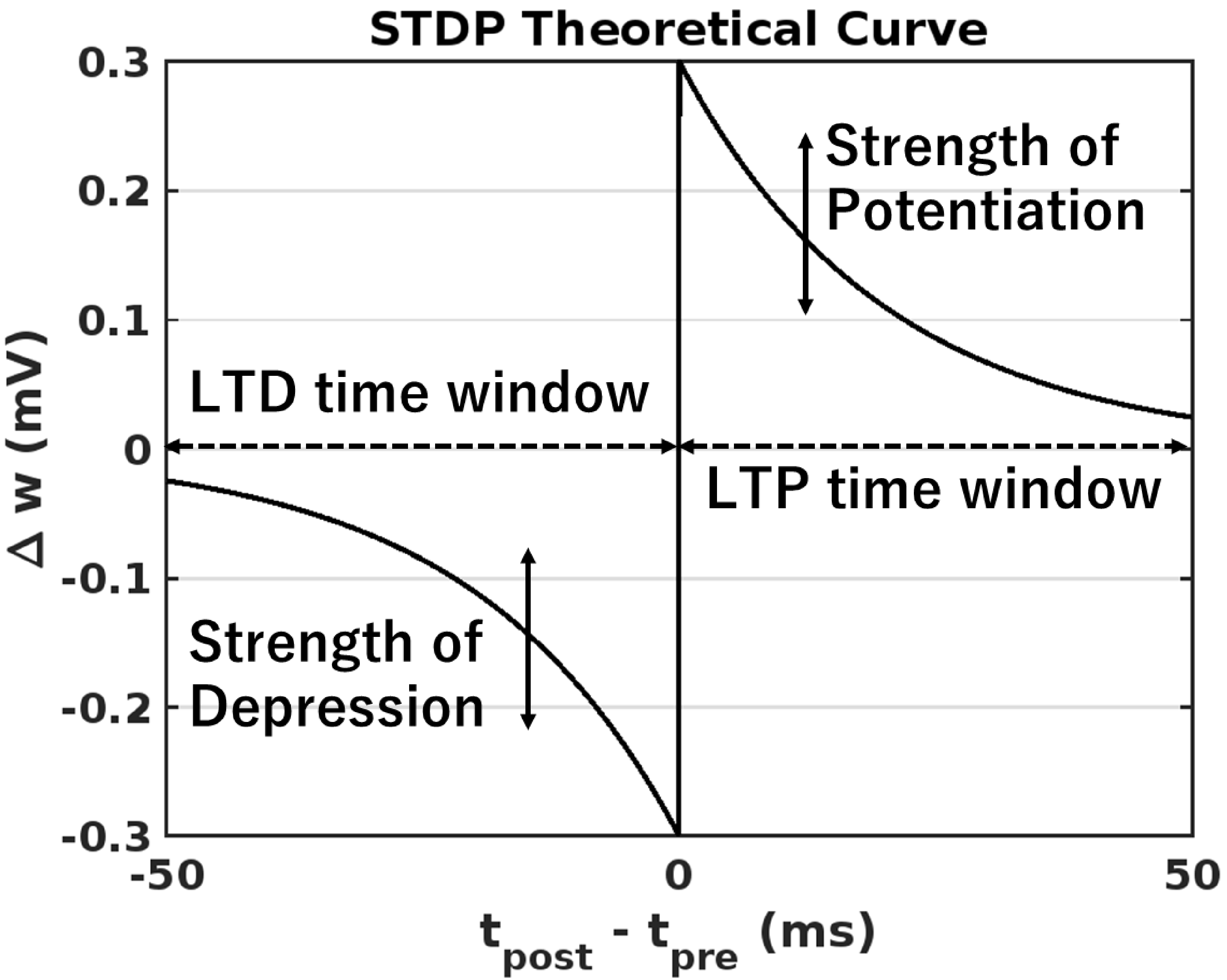
Measures similarity of patterns
-1 (Complete opposite), 0 (No similarity), 1 (Same)

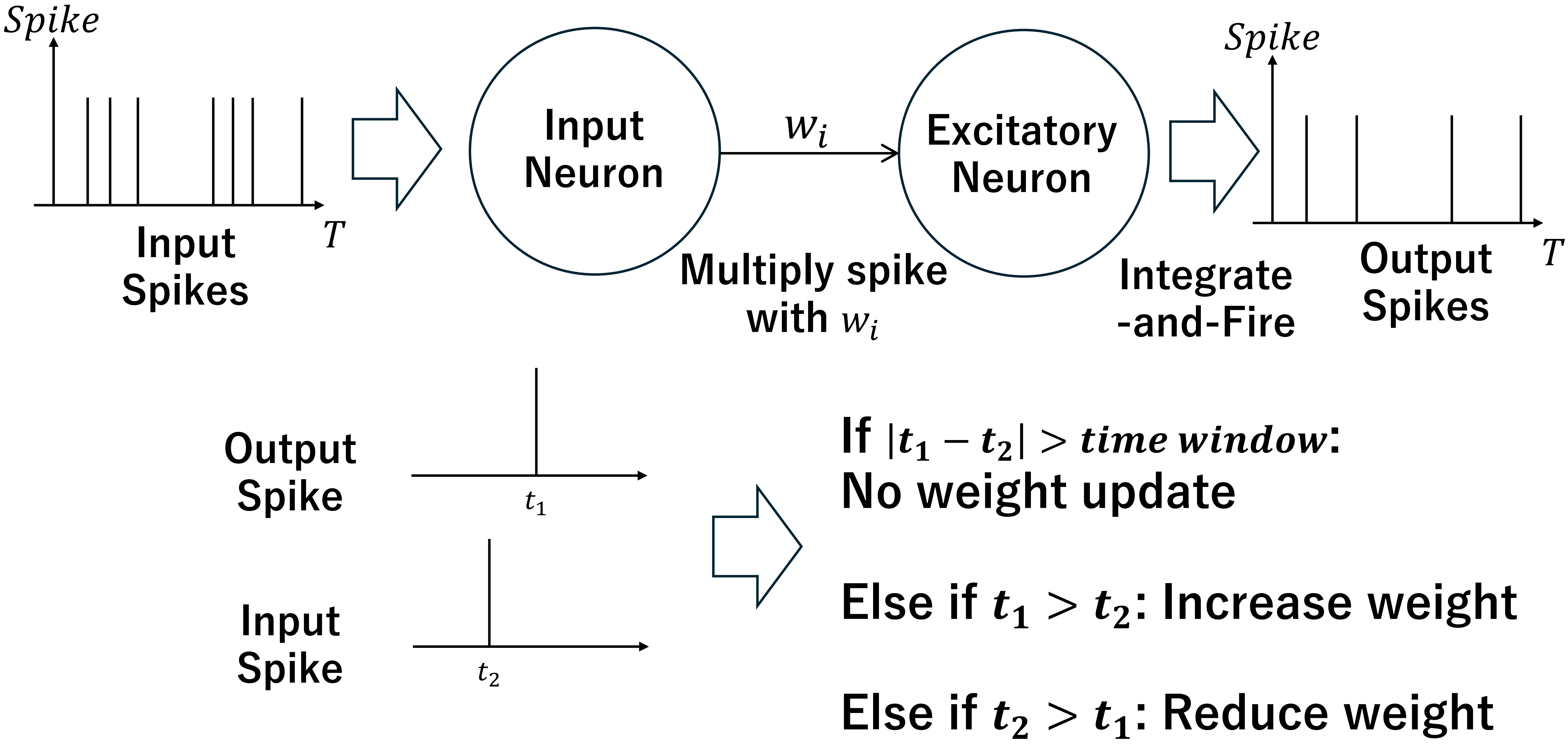
Similarity Measurements (4)

Correlation Coefficient (Range: -1 to 1)

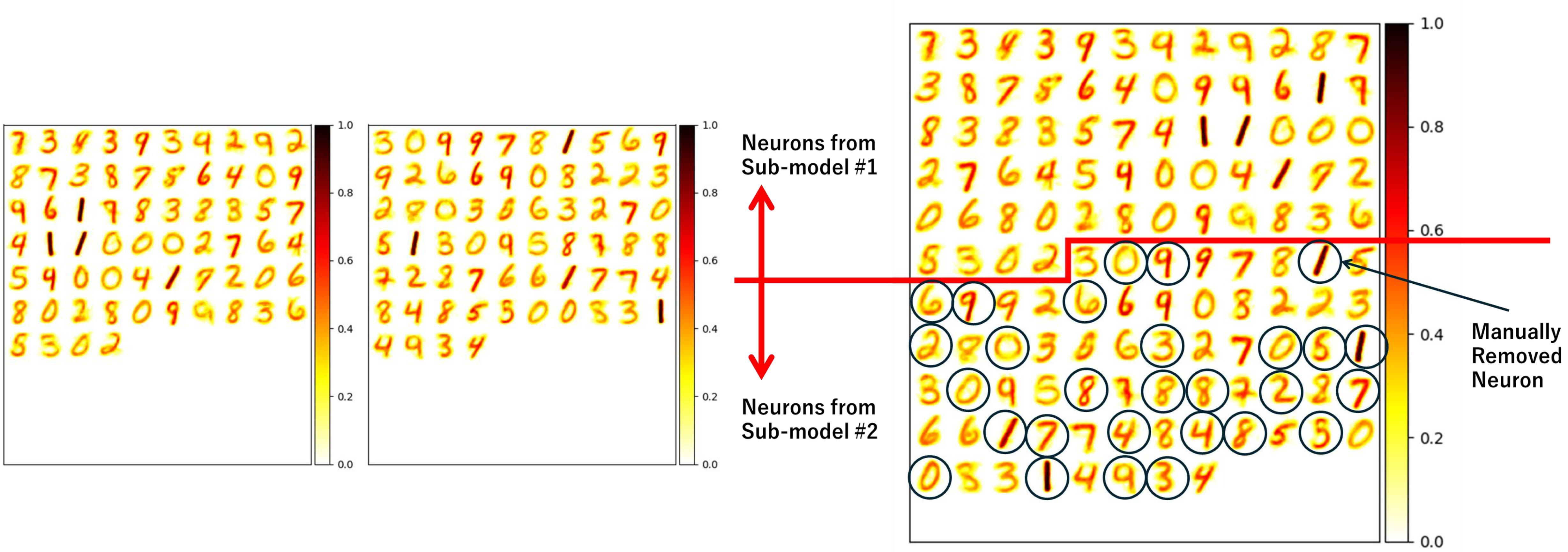
$$\frac{\frac{1}{n}\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i - \bar{A})^2} \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n}(B_i - \bar{B})^2}}$$

Measures how well each pattern correlates
-1 (Complete opposite), 0 (No similarity), 1 (Same)


STDP Theoretical Curve
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
Δ w (mV)
-50
0
50
t_{post} - t_{pre} (ms)
Strength of
Potentiation
LTD time window
LTP time window
Strength of
Depression



$Spike$
Input Spikes
T
Input Neuron
w_i
Excitatory Neuron
Multiply spike with w_i
Integrate -and-Fire
$Spike$
Output Spikes
T
Output Spike
t_1
Input Spike
t_2
If $|t_1 - t_2| > time\ window$:
No weight update
Else if $t_1 > t_2$: Increase weight
Else if $t_2 > t_1$: Reduce weight



Model	sub-model #1	sub-model #2	Merged Model	Manually Compressed Model
No. of neurons	64	64	128	100
Accuracy	75.38%	75.54%	77.60%	76.84%