2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

TactileSGNet: A Spiking Graph Neural Network for Event-based Tactile Object Recognition

Fuqiang Gu, Weicong Sng, Tasbolat Taunyazov, and Harold Soh
Dept. of Computer Science, School of Computing,
National University of Singapore
{gufq, sngweicong, tasbolat, harold}@comp.nus.edu.sg

Abstract— **Tactile perception is crucial for a variety of robot tasks including grasping and in-hand manipulation. New advances in flexible, event-driven, electronic skins may soon endow robots with touch perception capabilities similar to humans. These electronic skins respond asynchronously to changes (e.g., in pressure, temperature), and can be laid out irregularly on the robot's body or end-effector. However, these unique features may render current deep learning approaches such as convolutional feature extractors unsuitable for tactile learning. In this paper, we propose a novel spiking graph neural network for event-based tactile object recognition. To make use of local connectivity of taxels, we present several methods for organizing the tactile data in a graph structure. Based on the constructed graphs, we develop a spiking graph convolutional network. The event-driven nature of spiking neural network makes it arguably more suitable for processing the event-based data. Experimental results on two tactile datasets show that the proposed method outperforms other state-of-the-art spiking methods, achieving high accuracies of approximately 90% when classifying a variety of different household objects.**

I. INTRODUCTION

Object recognition is a basic perceptual skill that underlies many tasks, from driving a car to preparing a meal. Advances in machine vision have provided robots with excellent visual object recognition capabilities (e.g., [1], [2]). But while vision serves as an important visual modality, it can fail to distinguish objects with similar visual features or in less-favorable conditions, e.g., under low-lighting or occlusion. In such cases, tactile sensing can provide important information (e.g., texture, roughness, friction), which has been applied in a variety of tasks including object recognition [3], [4], [5], slip detection [6], and texture recognition [7].

This study focuses on the challenging task of touch-based object recognition with *event-driven tactile* sensors [8], [9]. Prior works (e.g., [7], [4], [3]) have mainly used standard synchronous tactile sensors with conventional machine learning approaches (e.g., convolutional neural networks [10]). However, event-driven sensors are inherently different, both in terms of operation and data provided. Similar to event-based cameras [11], [12], event tactile sensors asynchronously report changes in the environment and thus, provide event-based "spikes" where each taxel fires independently of the rest. Compared to standard synchronous frame-based sensors, event-driven sensing can achieve higher power-efficiency, better scalability, and lower latency. However, learning with these sensors remains in its infancy [13].

In this paper, we present TactileSGNet, a novel spiking graph neural network for object recognition using event-based tactile data. In contrast to convolutional neural networks for grid-structured real-valued data, our model operates on *graph-structured spiking data*. This provides two key advantages: first, the model can better exploit local taxel structure that can be highly irregular, e.g., with biologically-inspired configurations or flexible sensors that are wrapped around end-effectors. Second, spiking neural networks (SNNs) are also event-driven and can directly process the spike-based data provided by the sensors; this bypasses potentially expensive transformations from discrete events to real-valued frames. In addition, SNNs can be run on power-efficient neuromorphic processors such as the IBM TrueNorth [14] and Intel Loihi [15].

To our knowledge, TactileSGNet is the first event-driven graph neural network for tactile data. A related model is the recently proposed TactileGCN [16], which uses a graph convolutional network (GCN) [17] for tactile object recognition. The key differences in this work is that TactileSGNet is event-driven (with spiking neurons) and we utilize a topology adaptive graph convolutional network (TAGCN) [18]; the TAGCN has been previously shown to achieve superior performance, whilst being computationally more efficient compared to standard GCNs. Indeed, our computational experiments on two existing event-based tactile datasets using the NeuTouch sensor [8] show that leveraging the TAGCN with spiking neurons achieves superior performance to other popular architectures. We also experiment with alternative approaches for constructing tactile graphs; results suggest automated methods, specifically nearest-neighbor and minimum spanning tree techniques, can achieve even better performance.

II. BACKGROUND & RELATED WORK

Our work combines the recent advances in graph neural networks (GNNs) and spiking neural networks (SNNs) for event-based tactile object recognition. In the following, we provide a brief overview of background and related work in these areas. Note that these research areas are broad and, due to space constraints, we cover representative work and refer readers wanting more details to more comprehensive survey articles.

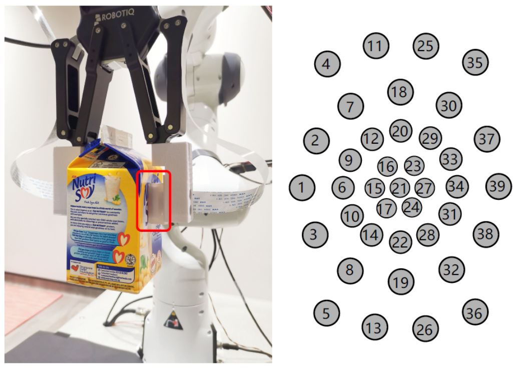


Fig. 1: The NeuTouch sensor mounted on a Robotiq gripper and spatial distribution of 39 NeuTouch taxels.

Tactile Sensing. Tactile sensing provides a modality of information (e.g., roughness, textures, temperature) that is different from visual sensing; incorporating a sense of touch enables robots to better perceive physical environments. Tactile perception has been in many robot tasks such as object recognition [3], [4], [5], slip detection [6], and texture recognition [7].

To date, several types of tactile sensors have been developed (see [19] for a survey); popular sensors include the BioTac[1], PPS[2], and Tekscan[3]. In this paper, we focus on using NeuTouch, an event-based tactile sensor that have been proposed in recent work [8]. Little prior work exists for learning with event-based tactile data. Very recent work [8] proposed a multi-modal spiking network based on SLAYER [20]. Our work is different in that we explore graph spiking neural networks (rather than fully connected layers) with LIF (leaky integrate-and-fire) [21] neurons rather than SRM (spike response model) neurons [22].

Graph Neural Networks (GNNs) are a class of models that combine deep learning models and methods for structured data [23]. GNNs have gained popularity of late due to their applicability in many fields, from social network mining to embedding logic into deep networks [24].

Of particular interest for this work are GCNs where the convolution operation is conducted in the spectral domain via trainable graph filters [25], [26]. To reduce the computational cost of decomposition and projection in the frequency domain, graph filters are usually approximated using finite order polynomials. For example, in [25], [27], the graph filters are approximated using high-degree Chebyshev polynomials of the graph Laplacian matrix. A popular GCN [17] approximates graph filters with the first-order Chebyshev polynomials of graph Laplacian. More recent work [18] has proposed limiting the polynomials of the adjacency matrix (to maximum degree of two) to further reduce the complexity. In this study, we use the TAGCN to perform convolution on the tactile data due to its computational efficiency and demonstrated performance [18].

Spiking Neural Networks (SNNs) form a core approach in neuromorphic computing [28]. SNNs are more biologically plausible than deep neural networks (DNNs), and can be executed on power-efficient neuromorphic hardware (e.g., the Intel Loihi [15]). SNNs can have the similar network topology as DNNs, but use different neuron models. Commonly-used neuron models for SNNs include the LIF [21] and SRM [22]. One issue in SNNs is that the spike function is non-differentiable, making it impossible to use backpropagation to train the network. To address this issue, several solutions have been proposed, such as converting DNNs to SNNs [29], and approximating the derivative of the spike function [30], [20]. In this work, we use SNNs as they are able to directly handle spiking sensor data.

III. Proposed Method: Learning With Tactile Graphs

In this section, we provide a description of our graph-based approach for learning from event-based tactile data. As previously mentioned, unlike visual *pixels*, the *taxels* for touch sensing may be structured in an irregular fashion. Indeed, human touch sensors are distributed unevenly across the body (with correspondingly different neurological demands as illustrated by the popular cortical homunculus).

As artificial e-skins continue to develop in both capabilities and affordability, we anticipate that robots will incorporate flexible skins that provide similar (or possibly superior) touch sensing capabilities to humans. The tactile sensors may be "wrapped" around existing body parts or have taxels organized in irregular configurations. Consider the NeuTouch [8] used in our experiments; the NeuTouch is a biologically-inspired fingertip tactile sensor with 39 taxels that are arranged spatially in radial fashion (Fig. 1). In the following, we will use the NeuTouch as our running example to describe our method, but note that our approach can be utilized with other sensors with different taxel configurations and layouts.

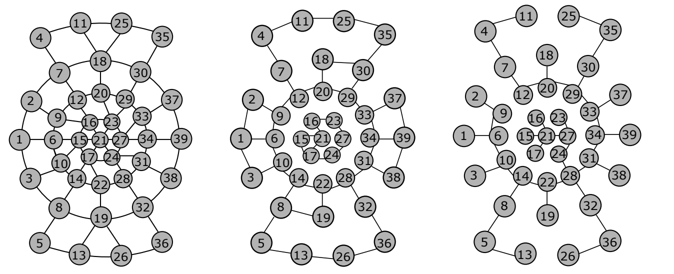
A. Tactile Graph Construction

To process the data from tactile sensors, one could adopt standard convolutional layers used in deep neural networks [31]. However, this would require "forcing" the data into a grid structure, which entails specifying an arbitrary grid size with zero-filled (or interpolated) cell values. Here, we undertake a more *natural* approach by constructing a tactile graph based on the local spatial arrangement of the underlying taxels.

Let $G = (V, E)$ be a tactile graph, where V is a set of N nodes, and E is a set of undirected edges[4]. The nodes are naturally mapped to taxels, but the edges have to be specified. We propose to leverage the spatial/geometric configuration of the points and introduce edges based upon the Euclidean

[1]`https://www.syntouchinc.com/technology/`
[2]`https://pressureprofile.com/`
[3]`https://www.tekscan.com/`

[4]We focus on undirected edges, but our method also accommodates directed edges.



(a) Manual (b) kNN (k=2) (c) MST ($\sigma_d = 0$)

Fig. 2: Tactile graphs constructed by different distance-based methods. (a) Graph constructed manually by-hand. (b) Graph obtained by kNN (k=2). (c) Graph generated using Kruskal's MST algorithm.

distances between nodes $d(v_i, v_j) = \|v_i - v_j\|_2$. Here, we explore three different distance-based methods:

1) **Manual**, where edges are manually connected according to their physical proximity;
2) **k-Nearest Neighbors (kNN)**, where each node is connected to its k closest neighbors;
3) **Minimum Spanning Tree (MST)**, where edges in a MST are added to the edge set of the graph, along with extra edges between any two nodes with distances smaller than a user-specified distance threshold σ_d.

As a concrete example, Fig. 2 illustrates the tactile graphs constructed by using the above methods for the NeuTouch. Our experiments will largely compare methods using the manual approach, but we include additional experiments that show how the graph connectivity affects performance on the object recognition task.

B. TactileSGNet

To process the data from our tactile graph, we propose a spiking neural network architecture we call TactileSGNet (shown in Fig. 3). The network uses LIF neurons, and includes a topology adaptive graph convolutional (TAGConv) layer [18], fully-connected (FC) layers, and a final voting layer for classification. In the following, we describe each of these components:

LIF Activations. In conventional convolutional neural networks, the most common activation functions are the ReLU [32] and its variants (e.g., LReLU [33]). However, the ReLU activation function is unsuitable in SNNs. We use the LIF model, which is a popular model for describing the dynamics of spiking neurons [34], [35], [28]. The dynamics of the LIF neuron is described by

$$\tau \frac{du(t)}{dt} = -u(t) + \sum_i w_i x_i, \tag{1}$$

where $u(t)$ represents the internal membrane potential of a neuron at time t, $\sum_i w_i x_i$ is the weighted summations of the inputs from pre-neurons, and τ is a time constant. Fig. 4 visualizes the dynamics of a LIF spiking neuron.

To better understand the membrane potential update, we can apply the Euler method to approximate the solution of the differential equation (1) shown above [30]. Then, the update of the membrane potential can be written as:

$$u(t+1) = (1 - \frac{dt}{\tau})u(t) + \frac{dt}{\tau}\sum_i w_i x_i, \tag{2}$$

which can be further simplified as:

$$u(t+1) = \beta u(t) + \sum_i w_i' x_i, \tag{3}$$

where $\beta = 1 - \frac{dt}{\tau}$ can be considered as a decay factor, and w_i' is the weight incorporating the scaling effect of $\frac{dt}{\tau}$. Thus, the LIF activation function f_{LIF} is described as:

$$f_{LIF}(u) = \text{fire a spike } \& \ u(t) \leftarrow u_R \text{ , if } u(t) \geq u_T \tag{4}$$

where u_R and u_T are constants, representing the reset value and firing threshold, respectively. The LIF activation function indicates that a neuron will fire (i.e., output a spike) when its membrane potential reaches or surpasses a given threshold u_T. After the firing, its membrane potential will be reset to a value u_R.

TAGConv Layer. Compared to the popular graph convolution [17], the TAG convolution [18] used in our network adapts to the topology of the input graph. A TAG convolution operation is defined as:

$$\mathbf{z}_f = \sum_{c=1}^{C} \mathbf{G}_{c,f} * \mathbf{x}_c + \mathbf{b}_f, \tag{5}$$

where $\mathbf{z}_f$ is the f-th output feature map, $\mathbf{x}_c$ is the c-th input feature of all nodes ($\mathbf{x}_c \in \mathbb{R}^N$, where N is the number of nodes), C is the number of input features of each node. $\mathbf{b}_f$ is a learnable bias vector, $*$ is the convolution operator, and $\mathbf{G}_{c,f}$ is the f-th graph filter. To make the convolution operation work for arbitrary graph topologies, the graph filter needs to be carefully designed. One approach is to define the graph filter with the normalized adjacency matrix of the graph,

$$\mathbf{G}_{c,f} = \sum_{k=0}^{K} g_{c,f,k} \mathbf{A}^k, \tag{6}$$

where $g_{c,f,k}$ is the polynomial coefficient of the graph filter, and $\mathbf{A}$ is the normalized adjacency matrix.

Fully-Connected (FC) Layer. The FC layer is similar to those used in conventional neural networks, i.e.,

$$\mathbf{h} = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{7}$$

where $\mathbf{x}$ is the inputs from previous layer, $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, and $\mathbf{h}$ is the output feature.

Voting Layer. The voting layer is used to decode the network output. We adopt the voting strategy in [36], [30]; briefly, each output label is first associated to one of neurons in the voting layer. The predicted class is the one associated with the neuron with the largest number of votes (corresponding to spikes) averaged over the time window.

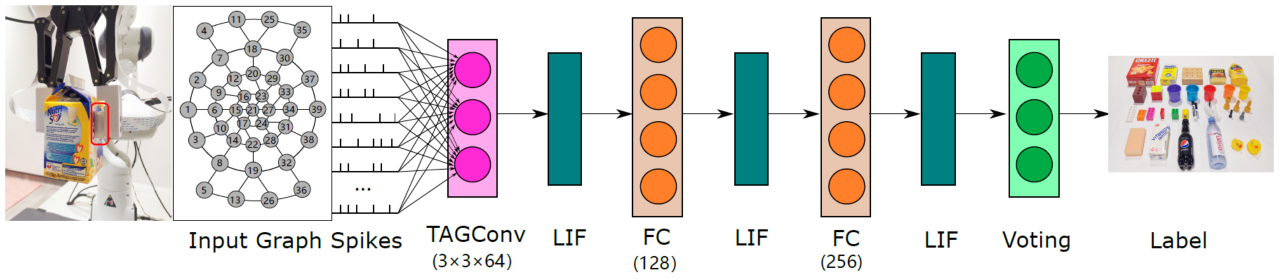


Fig. 3: TactileSGNet Architecture. TactileSGNet is a spiking neural network (SNN) that processes input spikes from taxels with connectivity specified by an input graph. It comprises a graph convolutional layer, two fully-connected (FC) layers, and a voting layer.

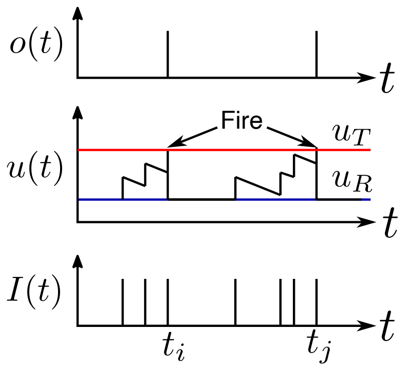


Fig. 4: The dynamics of a LIF neuron. It takes as input binary spikes and outputs binary spikes. $I(t)$ represents the input signal to a neuron, $u(t)$ is the membrane potential of the neuron, and $o(t)$ is the output of the neuron. An output spike will be emitted from the neuron when its membrane potential surpasses the firing threshold u_T, after which the membrane potential will be reset to u_R.

C. Training

To train the network, we define the loss function that captures the mean squared error between the label vector $\mathbf{y}$ and the averaged voting results over a given time window,

$$\mathcal{L} = \left\| \mathbf{y} - \frac{1}{T}\sum_{t=1}^{T} \mathbf{U}\mathbf{o}^t \right\|^2 \tag{8}$$

where $\mathbf{U}$ is the voting matrix, and $\mathbf{o}^t$ is the output feature from the last layer at time t.

In non-spiking neural networks, one can train a network by minimizing the loss function via standard backpropagation. However, spikes are non-differentiable. Fortunately, we can approximate the derivative of the spike function, which has been shown to be effected on various tasks [35], [37], [20]. In this study, we use the rectangular function $f(u)$ to approximate the derivative of the spike function due to its simplicity and reported performance [30], [21],

$$f(u) = \frac{1}{a}\text{sign}\left(|u - u_T| < \frac{a}{2}\right) \tag{9}$$

where a is a width parameter.

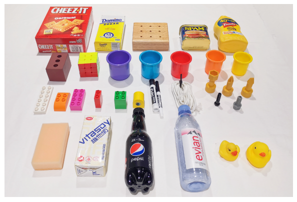
IV. Experimental Results

The primary objective of our experiments was to evaluate different architectures for event-based tactile object recognition. To this end, we compare alternative architectures including multi-layer perceptron (MLP) and convolutional neural networks (CNN) against our model. We also sought to understand the potential benefits of using the adaptive TAGConv layer, rather than a standard GCN. In the above experiments, we focused on the manually-designed input graph. We ran a second set of experiments to compare the three different graph construction methods described in Sec. III-A.

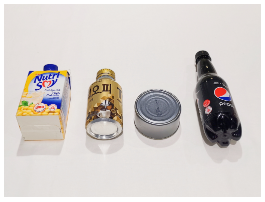
A. Datasets

We compared the methods using the recently developed event-based tactile datasets [8]. In brief, the datasets were collected using a 7-DoF Franka Emika Panda arm equipped with a Robotiq 2F-140 gripper, equipped with a NeuTouch event-based tactile sensor [8] and an ACES decoder [9] to decode the sensor signals into spikes. The Panda picked up a variety of different household objects to generate the two datasets:

- **EvTouch-Objects**: This dataset comprises tactile data from 36 object classes (Fig. 5(a)). Among these objects, 26 are objects from the YCB dataset [38], and the remaining 10 objects are deformable objects chosen to supplement the relatively rigid YCB objects. To collect tactile data, the robot gripper grasped the object, and lifted it off the table by 20 cm before placing it back onto the table. We used the data collected during the time from lifting an object to releasing it ($\approx$ 5 seconds). For each object class, 20 samples were collected, yielding a total of 720 samples.
- **EvTouch-Containers**: This dataset includes tactile data for four containers: a coffee can, a plastic soda bottle, a soymilk carton, and a metal tuna can (Fig. 5 (b)). These containers have a maximum volume of 250g, 400g, 300g, and 140g, respectively. Each container was filled with {0%, 25%, 50%, 75%, 100%} of the respective maximum amount of water (or rice for the open tuna can), resulting in 20 object classes. During the data collection, the robot gripper grasped each container, and



(a) EvTouch-Objects



(b) EvTouch-Containers

Fig. 5: (a) EvTouch-Objects dataset contains 36 object classes, including 20 objects from the YCB benchmarks (the first four rows). (b) EvTouch-Containers contains four types of containers: coffee can, plastic soda bottle, soy milk carton, and metal tuna can. Each container is filled with five amount of the same liquid, resulting in 20 object classes.

then lifted it off the table by 5 cm. We used the data collected during the time grasping an object to lifting and holding it for a while ($\approx$ 6.5 seconds in total). There are a total of 300 samples (15 samples per object class). This dataset may be particularly challenging for tactile sensing since the weights may not be easily distinguishable.

For both datasets, we used a bin duration of 0.02 seconds. Interested readers can find more details about the datasets in [8] and the corresponding website[5].

B. Compared Methods

We compared the proposed TactileSGNet with three baseline spiking architectures (described below). All methods used LIF neurons and were implemented in PyTorch using a LIF-based framework [30]. For fair comparison, all the methods share a similar network structure (number of layers, and number of units for respective layers) and similar hyperparameters. More precisely, the general network structure was `Input-TAGConv-FC1(128)-FC2(256)-Voting`, and we substituted the `TAGConv` layer with one of the following baselines:

- **MLP**, which uses a standard fully connected layer with 64 neurons to replace the TAGConv. This baseline represents the setup where minimal prior structure is introduced;
- **Grid-based CNN** where the tactile data was organized in a grid structure according to the spatial distribution of taxels [31]. We set the size of each grid cell to 1 mm $\times$ 1 mm, yielding a grid of 13$\times$19 cells. The raw readings from each taxel were assigned to a grid cell and the remaining unfilled grid cells were zero-filled.
- **GCN**, where we replaced TAGConv with the GCN [17]. As such, this baseline is similar to the state-of-the-art TactileGCN [16]), except that the network is a SNN.

Source code for our models is available at `https://github.com/clear-nus/TactileSGNet`.

[5]`https://clear-nus.github.io/visuotactile/`

TABLE I: Hyperparameter setting

Parameter	Value
Number of Layers	3
Membrane potential threshold u_T	0.5
Potential reset parameter u_R	0
Batch size	1
Decay factor of membrane potential β	0.2
Learning rate	1×10^{-3}
Width parameter of the approximated derivative a	0.5

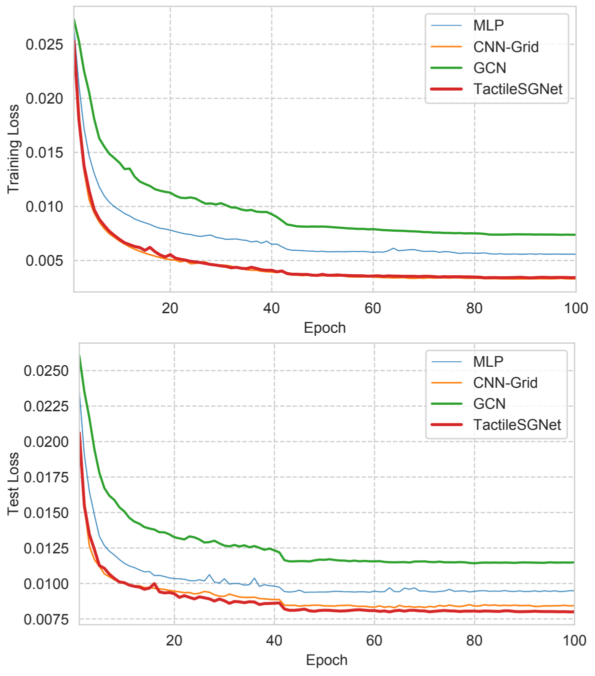


Fig. 6: Training and Test Losses as training progressed on EvTouch-Objects.

C. Training and Evaluation Methodology

The parameters used in our models (and for training) are given in Table I. We split the data into a training set (80%) and a test set (20%) with equal class distribution. We optimized each model on the training dataset for 100 epochs using the Adam optimizer. Our comparison measure was accuracy on the test dataset. We repeated the training and test procedure for 10 rounds (with different initialization).

We manually verified that all models were sufficiently trained by examining their training (testing) loss profiles. Figures 6 show the training (test) losses as the iterations progressed in a representative run; the training loss and test loss for all the methods decreased quickly at the beginning epochs and then gradually converged. The TactileSGNet converged faster than the other methods and has both a lower training loss and test loss.

D. Object Classification Performance

Table II shows the mean accuracy with standard deviation over the 10 rounds. We observe that the TactileSGNet outperforms the other methods, with a mean accuracy of 89.44% (on EvTouch-Objects dataset) and 64.17%

TABLE II: Test Accuracy of the Compared Methods on EvTouch-Objects and EvTouch-Containers.

Method	**EvTouch-Objects**	**EvTouch-Containers**
Grid-based CNN	88.40 (1.14)	60.17 (2.78)
MLP	85.97 (0.85)	58.83 (2.49)
GCN	85.14 (1.51)	58.83 (2.84)
TactileSGNet	**89.44 (0.55)**	**64.17 (2.75)**

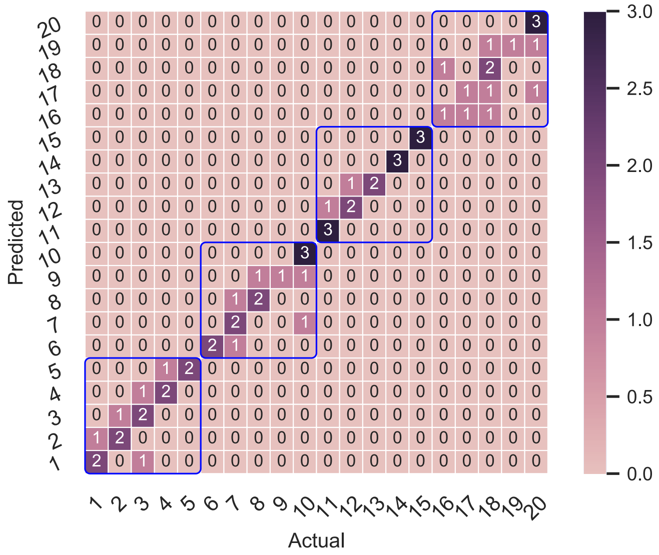


Fig. 7: TactileSGNet Confusion Matrix (on EvTouch-Containers dataset). Each blue rectangle denotes that the object classes within the rectangle are from the same container. Class 1-5 is for plastic soda bottle, class 6-10 is for tuna fish can, class 11-15 is for soy milk carton, and class 16-20 is for coffee can.

(on EvTouch-Containers), respectively. Among the baselines, the Grid-based CNN outperformed MLP and GCN the on both datasets. Compared to Grid-based CNN, our method has an accuracy improvement of about 1% and 4% on EvTouch-Objects dataset and EvTouch-Containers dataset, respectively. The reason for this difference may be the TAGConv layer was better able to process the graph-based representation which encoded the placement of the taxels.

Fig 7 shows the confusion matrix for test set generated from one round on EvTouch-Containers (an illustrative sample). Surprisingly, we found that TactileSGNet was able to perfectly distinguish the different containers (blue rectangles); we had initially expected the model to confuse objects of similar rigidity such as the coffee and tuna cans. TactileSGNet was also able to recognize the container fullness with a relatively high accuracy. The different weight classes in soy milk carton were easier to classify — possibly due to the softness of the container which allowed the sensor and model to detect the pressure exerted — while the coffee can was more difficult (possibly due to container rigidity). We also see that many of the incorrect classifications are reasonable and among similar weights.

TABLE III: Accuracy of TactileSGNet using different tactile graphs on the two datasets. $\langle k \rangle$ denotes the average node degree.

	Parameter	$\langle k \rangle$	EvTouch-Objects	EvTouch-Containers
Manual	-	-	89.44 (0.55)	64.17 (2.75)
kNN	k = 1	1	88.75 (0.64)	62.67 (2.53)
	k = 2	2	88.75 (0.79)	63.33 (1.18)
	k = 3	3	89.24 (1.00)	66.00 (0.91)
	k = 4	4	88.96 (0.69)	**67.00 (1.39)**
	k = 5	5	89.44 (0.44)	65.67 (2.79)
	k = 6	6	89.31 (0.67)	62.00 (0.75)
	k = 7	7	**89.65 (0.83)**	64.67 (4.31)
	k = 8	8	89.51 (0.83)	60.33 (1.39)
MST + σ_d	σ_d = 0.0	1.9	89.03 (0.64)	63.67 (2.17)
	σ_d = 1.5	2.1	88.68 (0.93)	**65.33 (3.80)**
	σ_d = 2.0	3.3	89.44 (0.55)	63.00 (0.75)
	σ_d = 2.5	4.1	89.31 (0.94)	65.00 (2.36)
	σ_d = 3.0	5.1	**89.51 (0.69)**	64.33 (1.49)
	σ_d = 3.5	8.4	89.17 (0.94)	63.00 (3.80)
	σ_d = 4.0	10.4	89.03 (0.85)	62.00 (1.39)

E. Impact of Graph Connectivity

In this section, we compare the three methods introduced in Section III-A as input to TactileSGNet. For kNN, the value of k we considered ranged from 1 to 8. For the MST + σ_d method, the distance threshold was set in the range of (0, 1.5, 2, 2.5, 3, 3.5, 4); note that the shortest distance between taxels on the NeuTouch is about 1.5 mm.

Table III shows the accuracy of the TactileSGNet using different tactile graphs. For the kNN method, the best mean accuracy is achieved when $k = 7$ on EvTouch-Objects, and when $k = 4$ on EvTouch-Containers. For the MST + σ_d method, the best mean accuracy is when $\sigma_d = 3$ on EvTouch-Objects, and when $\sigma_d = 1.5$ on EvTouch-Containers. Overall, the best accuracy achieved by both kNN and MST + σ_d methods is slightly higher on EvTouch-Objects and about 3% higher on EvTouch-Containers than the manual method. Tactile graphs with more edges (a larger k or σ_d) did not necessarily result in a better accuracy. The proposed method appears robust to the number of edge connections in tactile graphs, and the deviation in the classification accuracy achieved with different connections was relatively small (below 1% on EvTouch-Objects and 4% on EvTouch-Containers).

V. Conclusion

In this paper, we present a novel spiking graph neural network for event-based tactile learning. Compared to existing works, our method can exploit local topological structure of the taxels via an input graph. Experiments show that our method achieves higher performance compared to existing methods; using only event-based tactile data, TactileSGNet are able to distinguish various household objects with almost 90% accuracy. More broadly, our results indicate that event-driven tactile perception can be effective and we hope that our findings will spur research in this promising area.

Acknowledgments

This work was supported by the SERC, A*STAR, Singapore, through the National Robotics Program under Grant No. 172 25 00063.

References

[1] T. Gevers and A. W. Smeulders, "Color-based object recognition," *Pattern recognition*, vol. 32, no. 3, pp. 453–464, 1999.

[2] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 681–687.

[3] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands," *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[4] S. E. Navarro, N. Gorges, H. Wörn, J. Schill, T. Asfour, and R. Dillmann, "Haptic object recognition for multi-fingered robot hands," in *2012 IEEE haptics symposium (HAPTICS)*. IEEE, 2012, pp. 497–502.

[5] H. Soh, Y. Su, and Y. Demiris, "Online spatio-temporal Gaussian process experts with application to tactile classification," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4489–4496.

[6] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, "More than a feeling: Learning to grasp and regrasp using vision and touch," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3300–3307, 2018.

[7] T. Taunyazov, H. F. Koh, Y. Wu, C. Cai, and H. Soh, "Towards effective tactile identification of textures using a hybrid touch approach," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4269–4275.

[8] T. Taunyazoz, W. Sng, H. H. See, B. Lim, J. Kuan, A. F. Ansari, B. Tee, and H. Soh, "Event-driven visual-tactile sensing and learning for robots," in *Proceedings of Robotics: Science and Systems*, July 2020.

[9] W. W. Lee, Y. J. Tan, H. Yao, S. Li, H. H. See, M. Hon, K. A. Ng, B. Xiong, J. S. Ho, and B. C. Tee, "A neuro-inspired artificial peripheral nervous system for scalable electronic skins," *Science Robotics*, vol. 4, no. 32, p. eaax2198, 2019.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5633–5643.

[12] A. Mitrokhin, C. Ye, C. Fermuller, Y. Aloimonos, and T. Delbruck, "Ev-imo: Motion segmentation dataset and learning pipeline for event cameras," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 6105–6112.

[13] M. Pfeiffer and T. Pfeil, "Deep Learning With Spiking Neurons: Opportunities and Challenges," *Frontiers in Neuroscience*, vol. 12, no. October, pp. 774: 1–18, 2018.

[14] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[15] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[16] A. Garcia-Garcia, B. S. Zapata-Impata, S. Orts-Escolano, P. Gil, and J. Garcia-Rodriguez, "Tactilegcn: A graph convolutional network for predicting grasp stability with tactile sensors," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *arXiv preprint arXiv:1710.10370*, 2017.

[19] H. Liu, Y. Wu, F. Sun, and D. Guo, "Recent progress on tactile object recognition," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417717056, 2017.

[20] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.

[21] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, pp. 331:1–12, 2018.

[22] W. Gerstner, "Time structure of the activity in neural network models," *Physical review E*, vol. 51, pp. 738–758, Jan 1995.

[23] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[24] X. Yaqi, Z. Xu, K. S. Meel, M. Kankanhalli, and H. Soh, "Embedding symbolic knowledge into deep networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 4235–4245.

[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[26] F. M. Bianchi, D. Grattarola, C. Alippi, and L. Livi, "Graph neural networks with convolutional arma filters," *arXiv preprint arXiv:1901.01343*, 2019.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[28] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[29] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in neural information processing systems*, 2015, pp. 1117–1125.

[30] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1311–1318.

[31] B. S. Zapata-Impata, P. Gil, and F. Torres, "Non-matrix tactile sensors: How can be exploited their local connectivity for predicting grasp stability?" *arXiv preprint arXiv:1809.05551*, 2018.

[32] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[33] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the thirtieth international conference on machine learning (ICML)*, vol. 30, 2013.

[34] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[35] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[36] 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.

[37] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[38] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.